

Towards a framework for intelligent document image enhancement in pursuit of improved OCR performance

by

Ryno Kleinhans



Thesis presented in partial fulfilment of the requirements for the degree of
Master of Engineering (Industrial Engineering)
in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr GS Nel

March 2023

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2023

Abstract

A characteristic trait of the age of digitalisation is the ubiquitous transition from paper-reliant and manual-based business processes to completely digital, computer-assisted and automated versions thereof. Although many industries have already commenced with this transition away from paper documents, several real-world information chains are still intertwined with downstream paperbased systems. Some of these systems might require several decades to transition into a fully digital version thereof. Consequently, in order to fully automate these processes, the paper-based documents ought to be digitised.

Computerised approaches, *e.g.* optical character recognition engines, have achieved notable success in accurately extracting and transforming pixel-based information into machine-encoded information. The algorithmic performance of these engines is, however, reliant on the quality of the captured document images. Although there are a plethora of image enhancement techniques designed to increase image quality, the implementation of some of these techniques involves a large degree of dependency on human cognition as each document image requires a unique set of preprocessing steps. Accordingly, the application of data-driven approaches from the realm of machine learning — more specifically, deep learning — certainly warrants consideration within the presented context.

In this thesis, a generic framework for intelligent document image enhancement for improved optical character recognition is proposed. The focus of the framework is placed on facilitating the text extraction procedure of document images by automating the preprocessing stage by means of intelligently identifying the best combination of document image enhancement techniques to implement in respect of individual (document) images. Powerful approaches from the domain of computer vision, together with the implementation of transfer learning, are considered.

An instantiation of this framework is, first, implemented on a benchmark document analysis data set. Subsequently, the framework is applied to a real-world case study in the South African banking sector in order to illustrate the practical workability of the framework. During both instantiations, the models developed by means of the framework are shown to improve the optical character recognition accuracy of the document images.

Opsomming

'n Kenmerkende eienskap van die era van digitalisering is die alomteenwoordige oorgang van papier-afhanklike en handgebaseerde besigheidsprosesse na volledig digitale, rekenaargesteunde en outomatiese weergawes daarvan. Alhoewel baie nywerhede reeds begin het met hierdie oorgang weg van papierdokumente, is verskeie werklike inligtingskettings steeds verweef met stroomaf papiergebaseerde stelsels. Sommige van hierdie stelsels kan 'n paar dekades nodig om oor te skakel na 'n volledig digitale weergawe daarvan. Gevolglik, om hierdie prosesse ten volle te outomatiseer, behoort die papiergebaseerde dokumente gedigitaliseer te word.

Gerekenariseerde benaderings, *e.g.* optiese karakterherkenningsenjins, het noemenswaardige sukses behaal om pixel-gebaseerde inligting akkuraat in masjien-gekodeerde inligting te onttrek. Die werkverrigting van hierdie enjins is egter afhanklik van die kwaliteit van die vasgelêde dokumentbeelde. Alhoewel daar 'n oorfloed van beeldverbeteringstegnieke is wat ontwerp is om beeldkwaliteit te verhoog, behels die implementering van sommige van hierdie tegnieke 'n groot mate van afhanklikheid van menslike insig aangesien elke dokumentbeeld 'n unieke stel voorverwerkingstappe vereis. Gevolglik regverdig die toepassing van data-gedrewe benaderings uit die gebied van masjienleer — meer spesifiek, diep leer — beslis oorweging binne die voorgestelde konteks.

In hierdie tesis word 'n generiese raamwerk vir intelligente dokumentbeeldverbetering vir verbeterde optiese karakterherkenning voorgestel. Die fokus van die raamwerk word geplaas op die fasilitering van die teksonttrekkingsprosedure van dokumentbeelde deur die voorverwerkingsstadium te outomatiseer deur middel van intelligente identifisering van watter kombinasie van dokumentbeeldverbeteringstegnieke om ten opsigte van individuele beelde te implementeer. Kragtige benaderings vanuit die rigting van rekenaarvisie, tesame met die implementering van oordragleer, word oorweeg.

'n Instansiasie van hierdie raamwerk word eerstens geïmplementeer op 'n maatstafdokumentanalise datastel. Daarna word die raamwerk toegepas op 'n werklike gevallestudie in die Suid-Afrikaanse banksektor om die praktiese werkbaarheid van die raamwerk te illustreer. Tydens beide instansiasies word die modelle wat deur middel van die raamwerk ontwikkel is, gewys om die optiese karakterherkenning akkuraatheid van die dokumentbeelde te verbeter.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- To my supervisor, Dr GS Nel, for the monumental effort and extraordinary guidance provided to me over the past few years. I am grateful for his meticulous feedback and instilling a constant pursuit of excellence through my academic journey. Most of all, I sincerely appreciate his availability and willingness to (if fitting) just lend an ear, a notable character trait for any academic mentor. This enabled me the opportunity to articulate my own thoughts, further forming me as an independent researcher.
- To my family, Dr Rudi Kleinhans and Annemarie Kleinhans, for your unconditional love and support. Thank you, not only for your financial support, but for truly taking an interest in my work. I acknowledge your unwavering strength which served as my emotional foundation.
- To the numerous Capitec Bank employees (Marius Kuhn, Johan Olivier, PW Janse Van Renseburg, David Gouvias, Dean Matter, Carla Albertyn, and Jaco Moolman) who not only supported me in my academic endeavours, but also took me in as one of your own. Thank you for your stimulating conversations where we pondered about modern philosophy and data ethics.
- To the *Stellenbosch Unit for Operations Research in Engineering* (SUnORE) members, challenging and allowing me to grow me in ways more than merely in the realm of academics.
- To SUnORE for the generous financial assistance provided in the form of a bursary.
- To the Department of Industrial Engineering of Stellenbosch University for the availability of the academic facilities, enabling me to conduct my research in a professional and stimulating environment.
- To the One who provides me with the opportunity, intellectual capabilities, work ethic, and never-ending guidance.

Table of Contents

Abstract	iii
Opsomming	v
Acknowledgements	vii
List of Acronyms	xiii
List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Background	1
1.2 Problem statement	8
1.3 Research scope	8
1.4 Thesis objectives	9
1.5 Thesis organisation	10
1.6 Data collection and management	11
1.7 Ethical considerations	11
2 Deep learning	13
2.1 Artificial neural networks	13
2.1.1 Fundamentals of artificial neural networks	15
2.1.2 Activation functions	17
2.1.3 Network learning and training	19
2.2 Feedforward neural networks	22
2.3 Recurrent neural networks	22
2.3.1 Fundamentals of recurrent neural networks	23
2.3.2 The vanishing gradient problem	25

2.4	Long short-term memory	25
2.4.1	Fundamentals of long short-term memory	26
2.4.2	Relevant applications of long short-term memory	28
2.4.3	Connectionist temporal classification	28
2.5	Convolutional neural networks	29
2.5.1	Fundamentals of convolutional neural networks	31
2.5.2	Prominent convolutional neural network architectures	36
2.6	Transfer learning from pre-trained models	40
2.7	Chapter summary	43
3	Optical character recognition	45
3.1	OCR engines	45
3.1.1	Origins of OCR engines	46
3.1.2	OCR challenges when digitising printed text	47
3.2	Major phases of optical character recognition engines	48
3.2.1	Image acquisition	49
3.2.2	Preprocessing	49
3.2.3	Segmentation	50
3.2.4	Feature extraction	51
3.2.5	Character classification	51
3.2.6	Post-processing	52
3.2.7	Output distribution	52
3.3	Optical character recognition evaluation metrics	52
3.4	Prominent optical character recognition engines	55
3.4.1	Tesseract	55
3.4.2	EasyOCR	58
3.5	Chapter summary	60
4	Document image enhancement	61
4.1	Significance of image enhancement for text recognition	61
4.2	Geometric transformations	63
4.2.1	Skew correction	63
4.2.2	Text orientation correction	66
4.2.3	Cropping	67
4.3	Pixel transformations	68
4.3.1	Line removal	68

4.3.2 Binarisation	69
4.3.3 Noise removal	72
4.3.4 Image sharpening	74
4.4 Chapter summary	76
5 Intelligent document image enhancement framework	77
5.1 A generic data mining process	77
5.2 A high-level modular data mining framework	80
5.3 Document image enhancement framework	81
5.3.1 The Configure subcomponent	83
5.3.2 The Labelling subcomponent	88
5.3.3 The Modelling subcomponent	91
5.3.4 The Analysis subcomponent	94
5.4 Chapter summary	95
6 Proof-of-concept implementation	97
6.1 Data set background	97
6.2 Implementation of InDIE framework	101
6.2.1 Implementation of the Configure subcomponent	102
6.2.2 Implementation of the Labelling subcomponent	108
6.2.3 Implementation of the Modelling subcomponent	114
6.2.4 Implementation of the Analysis subcomponent	117
6.3 Results produced by InDIE framework implementation	117
6.4 Chapter summary	128
7 Case study implementation	129
7.1 Data set background	129
7.2 Implementation of the InDIE framework	133
7.3 Results produced by InDIE framework implementation	139
7.4 Subject-matter expert validation	148
7.5 Chapter summary	150
8 Conclusion	151
8.1 Thesis summary	151
8.2 Thesis contributions	153
8.3 Suggestions for future work	154

References

159

List of Acronyms

- AI:** Artificial Intelligence
- ANN:** Artificial Neural Networks
- AUC:** Area Under the Curve
- BGD:** Batch Gradient Descent
- BPTT:** Backpropagation Through Time
- CEC:** Constant Error Carousel
- CER:** Character Error Rate
- CNN:** Convolutional Neural Networks
- CRAFT:** Character Region Awareness for Text detection
- CRISP-DM:** Cross-industry standard process for data mining
- CRNN:** Convolutional recurrent neural network
- CSV:** Comma Separated Value
- CTC:** Connectionist Temporal Classification
- DFD:** Data Flow Diagrams
- DPI:** Dots Per Inch
- FNN:** Feed-forward Neural Networks
- FSP:** Financial Service Provider
- GD:** Gradient Descent
- GPS:** General Problem Solver
- GT:** Ground Truth
- ICDAR:** International Conference on Document Analysis and Recognition
- IDP:** Intelligent Document Processing
- InDIE:** Intelligent Document Image Enhancement Framework
- JPEG:** Joint Photographic Experts Group

JSON: Java Script Object Notation

LReLU: Leaky Rectified Linear Unit

LSTM: Long Short-term Memory Neural Networks

MAE: Mean Absolute Error

MLP: Multi-layer Perceptron

MSE: Mean Squared Error

NLP: Natural Language Processing

OCR: Optical Character Recognition

PAGE: Page Analysis and Ground-truth Elements

PReLU: Parametric Rectified Linear Unit

PRImA: Pattern Recognition and Image Analysis

RCL-CDIP: Ryerson Vision Lab Complex Document Information Processing

ReLU: Rectified Linear Unit

RMSE: Root Mean Squared Error

RMSProp: Root Mean Squared Propagation

RNN: Recurrent Neural Networks

ROC: Receiver Operating Curve

RPA: Robotic Process Automation

SGD: Stochastic Gradient Decent

SROIE: Scanned Receipts OCR and Information Extraction

SVM: Support Vector Machine

TBPTT: Truncated Backpropagation Through Time

WER: Word Error Rate

List of Figures

1.1 A depiction of the Turing Test	2
1.2 Basic model of machine learning	3
1.3 Relationship between different paradigms of artificial intelligence	4
1.4 Examples of two death certificate document images	7
2.1 A graphical representation of a generic biological neuron	14
2.2 A simple mathematical model of an artificial neuron	15
2.3 A representation of the general architecture of an FNN	16
2.4 The ANN learning process in the supervised learning paradigm	21
2.5 Varying inputs/outputs combinations	23
2.6 The basic RNN architecture	24
2.7 The vanishing gradient problem	25
2.8 The architecture of a typical LSTM memory cell block	26
2.9 A greyscale image as interpreted by a computer	30
2.10 A matrix flattened into a vector	30
2.11 The comparison of dense connectivity and sparse connectivity	31
2.12 The effect of vertical and horizontal convolution kernels	32
2.13 The convolution operation	33
2.14 Same padding applied to a 5×5 input tensor	34
2.15 Max pooling and average pooling	35
2.16 Max pooling of three different feature maps	35
2.17 Evolution of CNN architectures	36
2.18 The architecture of AlexNet	37
2.19 The architecture of VGG-16	38
2.20 Model scaling dimensions of the EfficientNet architecture	39
2.21 The architecture of EfficientNet-B0	40
2.22 Strategic implementation of pre-trained models	42

2.23 The size-similarity matrix and the corresponding decision map	43
3.1 Visualisation of the word “cat” in five alphabetic systems	49
3.2 A typical OCR engine	50
3.3 Visualisation of different feature extraction techniques	51
3.4 A typical OCR accuracy analysis workflow	53
3.5 A chronological depiction of OCR engine releases	55
3.6 A fixed-pitch word	56
3.7 Various scenarios faced by OCR engines	57
3.8 A segment of the word “arm”	57
3.9 Classification of the character “h”	58
3.10 The Easy OCR framework	58
3.11 Extracted feature sequences and frames	59
4.1 An example of a complex captured image with numeric text	62
4.2 An example of a preprocessed image with numeric text	62
4.3 An example of an incorrectly enhanced image with numeric text	63
4.4 Deskewing a document image	64
4.5 Horizontal projection lines of a skew and deskewed image	64
4.6 Detected lines based on the Hough transformer-based approach	65
4.7 Document image with four different orientations	66
4.8 Cropping procedure for a receipt document image	67
4.9 Document image line removal	68
4.10 A sample histogram with a global threshold	70
4.11 Examples of well-known binarisation algorithms	72
4.12 Noise removal on a document image	73
4.13 Image sharpening principle	75
4.14 Edge acutance before and after image sharpening	75
5.1 Six phases of the CRISP-DM reference model	78
5.2 Generic data mining framework	80
5.3 A populated generic data mining framework with specific modules	82
5.4 Level-one DFD of the Configure subcomponent	84
5.5 Intersection between two lists	88
5.6 Level-one DFD of the Labelling subcomponent	89
5.7 Possible enhancement procedure categories	90
5.8 Level-one DFD of the Modelling subcomponent	92

5.9 Level-two DFD of the Modelling subcomponent	93
5.10 Level-one DFD of the Analysis subcomponent	94
6.1 Five degraded SROIE receipt document images	100
6.2 Four ICDAR 2019 SROIE JSON files	101
6.3 Receipt document image deskewed and correctly orientated	104
6.4 Three ICDAR 2019 SROIE receipt document images	105
6.5 A receipt document image and its corresponding JSON file	107
6.6 Receipt GT feature list and its corresponding OCR output list	108
6.7 A cropped receipt document image	109
6.8 A binarised receipt document image	110
6.9 A denoised receipt document image	111
6.10 A sharpened receipt document image	111
6.11 An over-sharpened receipt document image	112
6.12 The top performing methods for document image classification	115
6.13 The base OCR accuracy results for the ICDAR 2019 SROIE data set	119
6.14 Graphically comparing the base OCR results with the enhanced OCR results	121
6.15 Graphically comparing the base OCR results with the sharpened OCR results	123
6.16 Training and validation AUC curves	124
6.17 Training and validation loss curves	125
6.18 Training and validation loss curves with tuned transfer learning	126
6.19 Test set confusion matrix results for the binary classification model	126
7.1 Examples of six generic payslip document images	132
7.2 The base OCR accuracy results for the payslip data set	140
7.3 Comparison of base OCR results with best OCR results	142
7.4 Base OCR accuracy and new best OCR accuracy distributions	145
7.5 Training and validation curves for the payslip model	145
7.6 Confusion matrix for the payslip test set	146

List of Tables

2.1	Summary of prevalent activation functions.	18
2.2	Commonly applied output layer activation functions for CNNs	36
6.1	Summary of average base OCR accuracy for the ICDAR 2019 SROIE data set	118
6.2	Summary of the receipt data set labelling distribution	120
6.3	Summary of assigned enhancement procedures of the receipt data set	120
6.4	Summary of the average full OCR accuracies for the receipt data set	122
6.5	Summary of the average best OCR accuracies for the receipt data set	122
6.6	Average OCR accuracy scenarios for different test set conditions	127
6.7	Comparing improved, unchanged, and deteriorated receipt results	128
7.1	Total missing values for each GT feature for the payslip data set	131
7.2	GT features ranked according to the guidance by subject-matter expert	134
7.3	Constructed enhancement procedures for the payslip data set case study	137
7.4	Summary of the payslip data set labelling distribution	141
7.5	Summary of assigned enhancement procedures of the payslip data set	141
7.6	Summary of the average full OCR accuracies for the payslip data set	143
7.7	Summary of the average best OCR accuracies for the payslip data set	144
7.8	Newly attained enhancement procedure label results	144
7.9	Average OCR accuracy scenarios for different test set conditions	147
7.10	Comparing the number of improved, unchanged, and deteriorated payslips	148

CHAPTER 1

Introduction

Contents

1.1 Background	1
1.2 Problem statement	8
1.3 Research scope	8
1.4 Thesis objectives	9
1.5 Thesis organisation	10
1.6 Data collection and management	11
1.7 Ethical considerations	11

1.1 Background

In 1946, the English mathematician Alan Turing produced a detailed design of a code breaking machine called *The Bombe* [110]. The machine was designed with the purpose to decipher the Enigma code which was utilised by the German army during the Second World War. *The Bombe* is considered as the first working electro-mechanical computer. The success achieved by this machine led to the 1950 publication of Turing’s seminal article “Computing Machinery and Intelligence” [286], an article that is well-known and highly regarded in the scientific community. It is within this landmark article that the question “Can machines think?” was considered for the first time. Turing devised a useful test to ascertain whether a machine exhibits intelligence. Known as the Turing Test (depicted in Figure 1.1), it comprises a human interrogator (entity A) communicating with two other unknown entities, namely: Another human being (entity B) and a machine (entity C). The human interrogator can only communicate with entity B and entity C with written natural language and can only receive answers from entity B and C in written natural language. If the human interrogator is unable to reliably distinguish the human from the machine when examining the written natural language received by these two entities, then the machine is said to exhibit intelligence — albeit a narrow form thereof. Today, the Turing Test is still considered as a relevant benchmark in order to identify whether a machine exhibits intelligence [247].

Roughly six years after the publication of Turing’s article the term *Artificial intelligence* (AI) was officially coined in 1956 by Marvin Minsky (a mathematician and computer scientist) and John McCarthy (a computer scientist and cognitive scientist) [110]. McCarthy defined AI as “the science and engineering of making intelligent machines, especially intelligent computer

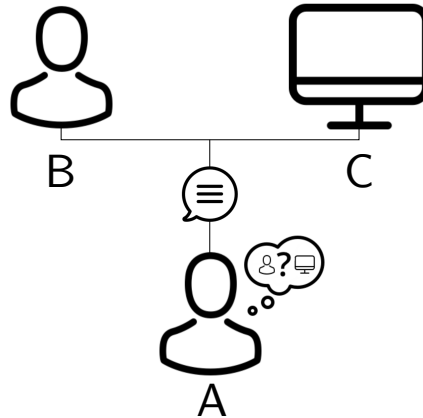


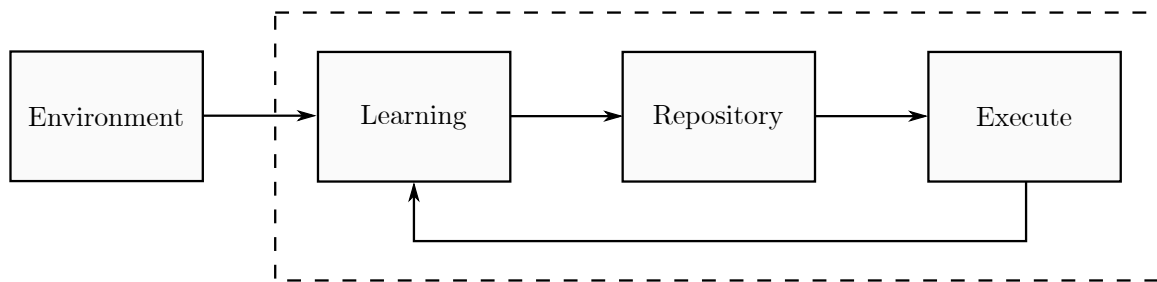
FIGURE 1.1: *The Turing Test depiction. A human interrogator (entity A) communicating with another human (entity B) and a machine (entity C) via written natural language.*

programs” [46]. Furthermore, he affirmed that AI is related to the notion of computers being able to “understand human intelligence,” however, he placed emphasis on AI not only being limited to “methods that are biologically observable.”

The following two decades (*i.e.* 1956–1973) was characterised by significant development and success in the field of AI. In 1959, the *General Problem Solver* (GPS) program was developed [201]. GPS was able to solve simple mathematical problems by separating a problem into smaller sub-problems and attempting to solve each sub-problem individually. Devised in 1966, the well-known ELIZA computer program was the first *natural language processing* (NLP) tool capable of simulating a conversation with a human [301]. A few decades later, in 1997, a chess program developed by IBM, called Deep Blue, was able to beat the world chess champion [39]. In order to determine the next optimal step in respect of 20 moves ahead, Deep Blue could process 200 million possible moves per second. Developed in 2015, the computer program *AlphaGo* showcased the potential of AI when the program beat the world champion in the immensely complicated board game *Go* [261]. Being significantly more complex than chess, it was long believed that computers will never have the computational capacity to beat a human in *Go* (*e.g.* at opening there are 361 possible moves in *Go*, whereas there are only 20 in chess). By utilising modern and efficient algorithms, AlphaGo was able to achieve a superior level of performance in order to beat its human opponent.

Consequently, it can be inferred that AI has been researched actively in the science and academic fields since the 1950s. Today, AI is employed in various industries, namely: Logistics, healthcare, retail, banking and finance, real-estate, and manufacturing, just to mention a few [277]. It is apparent that AI plays a significant role in day-to-day activities and will continue to fundamentally impact lives as the AI research field expands.

Machine learning is regarded as a subfield of AI and is employed towards simulating human learning activities. In 1959, Arthur Samuel (a pioneer in the field of computer gaming and AI) was the first academic to define machine learning as “the field of study that gives computers the ability to learn without being explicitly programmed” [140]. One of the primary aims of machine learning is to continually improve in an automated manner when exposed to new data (*i.e.* experience). A basic model of the machine learning process is depicted in Figure 1.2, where the system receives new information from the external environment. In order to learn, the system processes the new external information and identifies patterns, thereby converting the information into knowledge. The newly attained knowledge is stored within a repository. The repository comprises many general principles and design rules which are used to guide

FIGURE 1.2: *The basic model of machine learning* [298].

the system towards performing an action. After a certain action is completed, the information attained by performing the task is fed back into the learning stage of the system in order to facilitate self-adjustment. Machine learning algorithms can learn patterns markedly faster and adapt its strategy accordingly when compared with the capability of a human to learn a pattern and execute an action. Consequently, manual labour can be replaced by machine learning algorithms in order to execute certain tasks traditionally performed by humans. Image recognition, NLP, online fraud detection, medical diagnosis, and spam detection are all examples of industry problems which can be approached by means of machine learning algorithms [50, 56, 159, 260, 312].

Deep learning is a subfield of machine learning which attempts to learn high-level abstractions embedded within data by imitating the human brain [106]. One of the foundational underpinnings of deep learning is *artificial neural networks* (ANNs). Modelled on the human brain, an ANN comprises artificial neurons connected so as to form a network-like architecture — the aim of which is to achieve expedient learning from experience. Compared with more traditional machine learning approaches, ANNs can model linear and/or non-linear relationships [316]. Prior to the 1980s, the programmatic implementation of ANNs was too computationally expensive, however, algorithmic breakthroughs and the marked decrease in computing hardware cost, together with the increased capabilities of computer chips (*e.g.* GPUs), facilitated the increase in popularity of deep learning techniques over the past few decades [106]. The subfield of machine learning comprises many statistical learning algorithms which are capable of extracting patterns within data, however, when dealing with *unstructured data*¹ ANNs tend to outperform most other machine learning techniques [64].

During the last few decades, many state-of-the-art machine learning algorithms have been outperformed by deep learning approaches [296]. *Computer vision*, which is a subfield of AI, is one of the most prominent fields dominated by deep learning approaches. The relationships between AI, machine learning, deep learning, and computer vision are depicted in Figure 1.3. Computer vision aims to imitate the complexity underlying the mechanisms of human vision in order to enable computers to analyse visual information algorithmically. In 1998, Yann LeCun *et al.* [169] introduced their work on so-called *convolutional neural networks* (CNNs) in the paper titled “Gradient-based learning applied to document recognition”. In this paper Yann LeCun *et al.* reported on how a CNN is capable of creating complicated abstractions by progressively aggregating basic features and synthesising them. In their paper, the capabilities of various statistical algorithms towards identifying handwritten digit recognition tasks are reviewed and it was concluded that the novel CNN algorithm outperforms all other techniques. The potential and utility of CNNs were recognised, however, to effectively utilise a CNN, two requirements ought to be met, they are (1) an abundance of data and (2) sufficient computational resources.

¹Data that do not reside in a fixed field within a file (*e.g.* images, audio, or text).

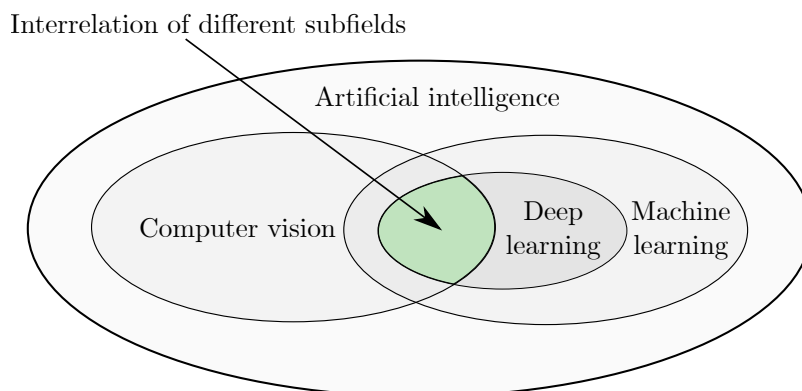


FIGURE 1.3: Venn diagram showing the relationship between artificial intelligence, machine learning, deep learning, and computer vision.

In the 1990s, computers did not meet the computational requirements in order to facilitate the effective application of CNNs.

Roughly two decades later, in 2012, a group of Canadian researchers developed AlexNet (named after the main creator, Alex Krizhevsky), which is a CNN-based approach applied in the context of image recognition [162]. AlexNet outperformed the current best-practice image recognition algorithms with ease. The main factor that distinguished AlexNet from the other algorithms is the application of CNNs. This was achieved by training the CNN on a markedly large data set called ImageNet [62]. Released in 2009, ImageNet comprises millions of labelled images. The continuous improvement of computer capabilities (in respect of hardware especially) also contributed to the success of AlexNet. The results obtained by AlexNet showcased that the improvement of modern technology made it practically feasible to utilise deep learning algorithms in the domain of computer vision. Today, CNNs form a pivotal part of most computer vision applications. Examples of these computer vision applications, include: Enabling autonomous cars to track objects in their surroundings, mapping high vulnerable areas during natural disasters, informing farmers of crop health and quality, and detecting fatal illnesses of medical patients [31, 204, 242, 284].

A characteristic trait of the proliferating age of digitalisation is the phenomenon that companies within every industry are rapidly transitioning and digitising their business processes from manual and paper-reliant to fully digital and computer-automated versions thereof. By being less dependent on manual labour, companies are attempting to utilise the benefits that accompany this digitised paradigm. There are various practical and financial incentives for companies to increase the computer automation of manual processes, namely: Improved human resource utilisation, improved information security, reduced potential human errors, simplified information transfer, and savings with respect to valuable company time (and money). Although the *financial service provider* (FSP) industry is certainly not exempt from this transition, they are, arguably, not transitioning at the same rate as other industries. In-branch loan applications, in-branch account applications, cheques, and monthly bank statements, to name but a few, are examples of processes that hinder and inhibit most FSPs from adopting a digital-only process workflow. These examples are especially prevalent for FSPs located in developing countries as the deficiency and access to new technology impedes innovation.

Although there remain processes that are traditionally executed manually, the developments in computer vision, including *optical character recognition* (OCR) technology, represent a possible solution for the FSP industry towards comprehensively digitising their processes. OCR technology, commonly implemented by employing CNNs, is defined as the electronic conversion of

images of typed, handwritten or printed text into machine-encoded text [131]. Before employing OCR on an image containing textual information, it is imperative to implement appropriate image enhancement techniques as the performance of an OCR engine is largely dependent on the quality of the image [163]. There are various well-researched image enhancement techniques designed and developed for reducing the imperfections of *degraded* images (*i.e.* suboptimal for OCR purposes). Although the implementation of these techniques are aimed at improving the quality of the image, it is important to note that an incorrect transformation might be detrimental to the image quality [63]. Some of these techniques include: Binarisation, de-skewing, sharpening, noise removal, cropping, orientation correction, and line removal. When there is a need to digitalise fixed structure documents (*i.e.* documents where specific information is located on the exact same place on every document), OCR is often employed in combination with *robotic process automation* (RPA). Traditionally, after an OCR method is applied to a structured document image to recognise the characters in an image, the RPA software employs rule-based methods to extract relevant information from the new structured machine-encoded data [291]. These rules may include the coordinates of text boxes on a page, and what type of text should be located in each text box (*e.g.* identification numbers may only consist of number characters).

There are several examples in the current FSP industry in which OCR and RPA technology are employed to automate and digitise some process. One of the most popular applications of OCR and RPA technology is for customers to capture credit card information using a smartphone camera. Online credit card purchases constitute a significant part of the total revenue for companies selling their products and services by means of online platforms. Requiring customers to manually enter credit card information on an online software program when registering their account which can be a tedious and error-prone activity. The integration of OCR technology enables customers to utilise their smartphone camera to capture their credit card information without having to manually type it in.

In 2012, Uber implemented this principle in their smartphone application [213]. With the use of OCR and RPA technology, customers are no longer required to manually enter their credit card information. When signing up, customers can select the option to scan their credit card instead of typing the information in. The software program activates the smartphone camera and displays the view of the camera together with guidelines aiding the customer towards positioning their credit card correctly. The camera then captures a photo of the credit card and the program employs appropriate image enhancement techniques together with the OCR technology in order to recognise the characters on the credit card. After the characters are identified, the program can easily employ RPA and extract the required credit card information as the format of most credit cards are similar (*i.e.* the information is located roughly on the same area on the credit card). Finally, the captured information is then displayed to the customer to ensure that the process has been executed correctly. By employing the appropriate image enhancement techniques, together with OCR and RPA technology, the time spent signing up has been reduced and the customer experiences a more user-friendly process.

Alfa Bank is one of the largest private commercial banks in Russia. In 2020, Alfa bank partnered with an OCR technology specialist company, called Smart Engines [266]. By integrating OCR and RPA technology into the mortgage application process, employees could simply present the passport document of an applicant to a camera in order to capture and extract the relevant information. The format of passport documents is fixed, therefore the information can be easily exported into the relevant fields of the digital application form automatically. Before the OCR and RPA system's implementation, employees were required to manually enter the applicant's passport information. This frequently resulted in human errors and subsequent loss

in productivity. Consequently, the introduction of OCR and RPA technology yielded significant value for the Russian FSP.

Considering the two aforementioned examples, the combination of OCR and RPA systems has proved to be a noteworthy method for digitalisation when converting fixed structured document images into usable information. In both examples, traditionally manual and laborious tasks are successfully automated by the digitalisation of fixed structured document images. Due to the fixed layout templates used to structure the textual information on these type of documents it is considered to be far more elementary to locate the sought after textual strings and identify which document image enhancement techniques to implement for the document image data set at hand, since the enhancement procedure can be designed specifically for the data set circumstances and structured document type. This is, however, not always sufficient when it is required to digitise a specific domain of image data that comprise various unique document formats (*e.g.* various company payslips, different journal articles, and distinct business memos) and that differs significantly in capturing quality or composition (*e.g.* watermarks, noise, folds, smudges, shadows, light ink, and over-exposure).

Consider the two visualised death certificate document images in Figure 1.4. Although both document images stems from the same document domain, they differ significantly in layout and composition. The death certificate document image shown in Figure 1.4(a) is considered to be relatively clear with notable open space between different entities on the document while being printed on a yellow-shaded paper. Printed lines are present, but does not protrude too close to printed textual information. Conversely, the death certificate document image visualised in Figure 1.4(b) is considered to be markedly complex with several watermarks, small and closely printed textual information, and an abundance of printed lines located notably close to the printed textual information. Consequently, the document image enhancement techniques best suited for improving the quality of the first death certificate document image (Figure 1.4(a)) is expected to differ significantly from the document image enhancement techniques best suited for improving the quality of the second death certificate document image (Figure 1.4(b)). Moreover, although both document images contain the same pertinent textual information required on a typical death certificate, corresponding textual strings are far from being located at similar coordinates on the two individual document images. Therefore, the rule-based approach of RPA cannot be considered as a reliable method when non-standardised document templates are present.

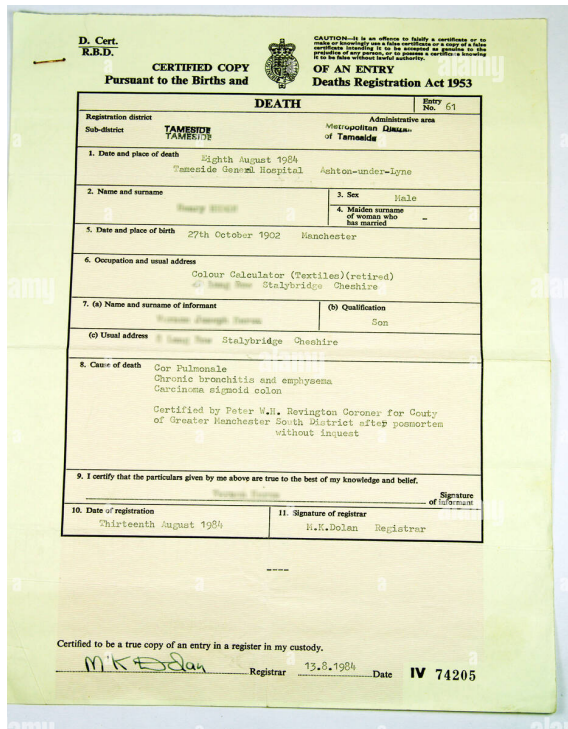
Considering the additional complexities alluded to during the aforementioned death certificate example, a need is identified for the design and development of a computerised solution when it is desired to automate the extraction of printed textual information from a data set comprising non-standardised and uniquely degraded document images. A short overview of a contextual case study now follows which showcases the aforementioned need practically.

The South African FSP industry may be regarded as an oligopoly² market structure. Naturally, it is notably difficult for a new bank to enter the market. In 2001, Capitec entered the market and within two decades became the largest South African retail bank in terms of active clients [37]. Capitec's differentiating factor is represented by their aim of catering to the lower income groups of South Africa. The small banking costs of Capitec have also attracted higher income individuals, who primarily bank at other FSPs, towards opening secondary³ accounts at Capitec in order to reap some of the tangible benefits. Another reason for the success

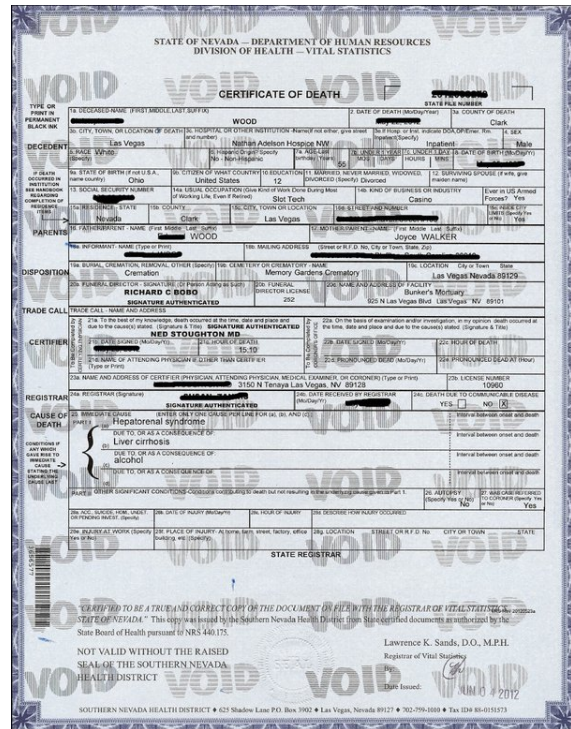
²Market shared with a small number of entities with limited competition.

³For the purpose of this thesis, the so-called primary bank of a client refers to the bank into which the main income (or salary) of the client is paid.

1.1. Background



(a)



(b)

FIGURE 1.4: A visualisation of the two different death certificate document images [5].

achieved by Capitec stems from their unique unsecured loans⁴ offering (which can amount up to R250 000) [42]. In order to approve an unsecured loan, the bank needs to determine the creditworthiness of the loan applicant. Currently, Capitec requires an applicant to visit a branch and provide a consultant with three documents for this process, namely: A South African ID (of the applicant), the applicant’s latest payslip, and the most recent bank statements of the applicant which is used to determine if the applicant has a stable income. The consultant then makes copies of these documents and manually enters (*i.e.* annotates) the required information (*e.g.* identification number, client name, client surname, occupation, employer name, age, net pay, and employer address) into a computer system.

Capitec identified this manual and timely process as a waste of valuable resources and desires to eliminate the need to manually enter the applicant’s information into their system. In order to automate this process, OCR is considered as a method for converting the pixel-based information into machine-encoded information, however, due to the non-standard variety of payslip templates and payslip degradations, some scanned payslip document images might require the implementation of unique combinations of document enhancement techniques (*i.e.* a unique enhancement procedure). This multi-faceted computer vision problem faced by Capitec forms the basis of work carried out in this thesis. Although Capitec stands to benefit greatly from a solution to the main problem addressed in this thesis, the general methodology adopted may be applicable to many incarnations of this problem in different industries and domains.

⁴Loans that do not require collateral.

1.2 Problem statement

There is an abundance of research dedicated to the development of computerised approaches for the purpose of enhancing, digitising, and processing document image data. Modern OCR engines have achieved notable success in extracting pixel-based information from document images and transforming it into machine-encoded information. The performance of these OCR engines is, however, dependent on the quality of the captured document images.

Although there are a plethora of image enhancement techniques designed to increase document image quality, the implementation of some of these techniques may either be beneficial or detrimental to digitalisation of the captured document image — depending on the possible degradations and composition. Consequently, the selection of which document enhancement technique(s) to implement on which document image involves a large degree of dependency on human rationale as each document image requires the implementation of a unique set of document enhancement techniques. Accordingly, the application of data-driven approaches from the realm of machine learning — more specifically, deep learning — certainly warrants consideration within the presented context in order to automate the selection and implementation of the appropriate document enhancement techniques. The underlying intuition is that a machine learning approach can be adopted towards learning the relationship between document images (more specifically, their visual features) and the applicability of enhancement techniques.

The principal aim in this thesis is to design, develop and demonstrate the practical workability of a generic framework that introduces machine intelligence to the digitalisation of document images in order to address the aforementioned shortcomings. The employment of the generic framework ought to be executed in pursuit of improving the potential digitalisation accuracy of a document image data set. The framework should facilitate (1) the process of preparing previously annotated data and its corresponding document images for analysis, (2) the engineering of various enhancement procedures appropriate to the considered document image data, (3) the utilisation of machine intelligence in assigning and implementing the best enhancement procedure for each unique document image, (4) as well as the analysis and synthesis of the obtained results. Powerful approaches from the domain of computer vision together with the implementation of transfer learning are considered.

1.3 Research scope

In order to narrow down the scope of the problem considered in this thesis, the following delimitations are adopted:

OCR engines. Multiple OCR engines have been developed throughout the past few decades, with some freely available to the public while others requiring the purchase of a software licence. Only two OCR engines are considered within this thesis, both of which are open-source software, namely the Tesseract OCR engine [92] and the EasyOCR engine [141].

Document image enhancement techniques. There are a plethora of problem specific document enhancement techniques, some of which comprise multiple hyperparameters. The document image enhancement techniques considered within this thesis are limited to the following: *Hough transformer-based approaches* [135] for skew correction, *morphological operators* [120] for line removal, *histogram equalisation* [77] and *local adaptive thresholding* [281] for binarisation, *median filtering* [126] and *Gaussian filtering* [275] for noise removal, and *high-pass filtering* [149] for image sharpening.

Input variables. The training, testing, and validation performed in this thesis are limited to two components of data, namely: The open-source annotation and document image data which are regularly used to evaluate and compare the performance of popular OCR models, and annotation and document images provided by the industry partner of this thesis.

Learning paradigm. Supervised learning in combination with transfer learning are the machine learning paradigms that forms the basis of the discourse considered within the modelling phase of this thesis.

Deep learning techniques. CNNs are selected to form the basis of discourse in this thesis. The CNN algorithm is a reputable architecture employed in the field of computer vision as it has the capability to convert extracted image features into lower dimensions while retaining its characteristics.

1.4 Thesis objectives

The following five objectives are pursued in this thesis:

- I To *conduct* a review of the pertinent literature related to:
 - (a) *Feedforward neural networks* (FNNs), *recurrent neural networks* (RNNs), *long short-term neural networks* (LSTMs), CNNs, and the application of transfer learning in the field of computer vision,
 - (b) the working of a typical OCR system, different OCR evaluation metrics, and prominent OCR engines, and
 - (c) the document image enhancement techniques considered within this thesis.
- II To *design* and *develop*, based on the literature review of Objective I, a modular and generic framework comprising the following four subcomponents:
 - (a) A subcomponent capable of *guiding* a user in processing the input data for analysis,
 - (b) a subcomponent which *facilitates* the engineering and comparison of enhancement procedure categories and the appropriate assignment thereof,
 - (c) a subcomponent aimed at providing *guidance* in the development and evaluation of a prediction model, and
 - (d) a subcomponent for *extracting* insights from the attained results.
- III To *verify* the potential utility of the framework of Objective II through a proof-of-concept implementation on a benchmark data set.
- IV To *validate* the practical workability of the framework of Objective II by execution of a case study on a real-world data set provided by the industry partner of this thesis.
- V To *recommend* sensible follow-up work related to the contributions of this thesis, which may be pursued in future.

1.5 Thesis organisation

Including this introductory chapter, the research underlying this thesis is executed across eight chapters aimed at addressing the problem statement and fulfilling the stated objectives.

Chapter 2 comprise a review of the material pertaining to ANNs, in fulfilment of Objective I(a). More specifically, the fundamentals of FNNs, RNNs, LSTMs, and CNNs are discussed in detail. The chapter concludes by exploring the concept and implementation of transfer learning — specifically in the realm of computer vision.

An in-depth review of literature pertinent to the understanding of OCR is conducted in Chapter 3, in fulfilment of Objective I(b). The chapter opens with an exploration of the typical OCR process, followed by the major steps involved in executing an OCR operation. The theoretical foundation of deep learning architectures, acquired in Chapter 2, expedites the discussion of two prominent OCR engines, *i.e.* Tesseract OCR and EasyOCR.

Chapter 4, in fulfilment of Objective I(c), is devoted to the consideration of document image enhancement techniques. First, the utility of implementing the correct document enhancement techniques are supported by means of an example. This is followed by a discourse on the most popular document enhancement techniques used for improving the performance of an OCR engine.

Following the review of the literature is the design and development of a generic framework which is capable of guiding a user in incorporating machine intelligence into the assignment and implementation of document image enhancement techniques in pursuit of an improved OCR performance. This chapter is executed in fulfilment of Objective II. Chapter 5 opens with a brief discussion on the architecture of a computer software and of the general data mining process, the inspiration for the framework architecture. The framework is then presented, followed by a detailed top-down discussion of the various subcomponents constituting the framework.

Chapter 6 comprises a discussion on the verification of the preposed framework through a proof-of-concept instantiation, in fulfilment of Objective III. First, an open-source and prominent benchmark document image data set is discussed in detail. Thereafter, the generic modules within the proposed framework is populated with specific algorithms and user-defined settings in order to illustrate the utility of the framework. The results generated throughout the employment of the framework are then evaluated and analysed.

In Chapter 7, a case study is performed in order to validate whether the framework can produce the desired outcome when implemented on a real-world data set provided by the industry partner of this thesis, in fulfilment of Objective IV. The chapter opens with an exploration of the industry partner data set. Thereafter, the framework is populated with specific algorithms and user-defined settings in order to illustrate the working of the framework. The implementation of the framework is briefly discussed, followed by an in-depth analysis of the obtained results. In concluding the chapter, the proposed framework and the obtained results are subjected to face validation by two industry partner subject-matter experts.

Chapter 8, the final chapter, serves as the conclusion of the thesis. A summary and critical appraisal of the contributions of the thesis are provided, followed by recommendations with respect to suitable future research endeavours, in fulfilment of Objective V.

1.6 Data collection and management

As mentioned above, the training of machine learning techniques, and specifically CNNs, require a large amount of data in order to be trained, validated, and tested. Consequently, in order to achieve the objectives of this thesis, it is pertinent to acquire real-world image data. A partnership has been established with the industry partner for this thesis (*i.e.* Capitec Bank) whereby they provide appropriate image data (and its corresponding annotations), in order for the author to successfully design and develop an effective and generic framework. Access to the data of the industry partner is afforded exclusively to the author through a company laptop. It must be noted that all data used in this thesis are electronic data — the acquisition, storage, or analysis of paper data are not required in order to complete this thesis. The data and identities of the individuals are stored on a secure Capitec Bank cloud based network which can only be accessed through a laptop provided by the industry partner. The data on this cloud based network are encrypted according to the standards and guidelines of the data governance department of Capitec Bank. The author is provided with a unique passkey which is required to utilise the laptop.

1.7 Ethical considerations

The image data sets employed in this thesis are notably sensitive. The data comprise personal, financial, and background information of the clients of the industry partner. Accordingly, the following principles of ethical research are considered in this thesis:

Be aware of the influence and role of a researcher. As a researcher, it is important to remember the potential influence that the results may have on all the stakeholders, and which roll to fulfil in the transfer of newly attained knowledge. When reporting results, the researcher ought to report the true results attained by the models and experiments, thereby contributing true knowledge and insight in the specific research paradigm.

Uphold confidentiality and privacy rights. When employing data in a research thesis, it is pertinent to uphold the privacy rights of the individuals who provided the data. It is therefore in the researcher's best interest to be familiar with the privacy laws pertaining to this specific research. In the case of this thesis, the data are exceptionally sensitive and ought to be kept confidential at all costs, not only to protect the individuals, but also the industry partner. Accordingly, it is the responsibility of the author to ensure that the sufficient measures are in place so as to uphold confidence in the confidentiality and privacy of the data.

Know and use ethical resources. It is the responsibility of the researcher to have several resources available and the willingness to employ these resources when deliberating about an ethical dilemma. Relevant documentation and field experts ought to be consulted when ethical dilemmas are considered in order to provide guidance and knowledgeable insight of the scenario.

CHAPTER 2

Deep learning

Contents

2.1 Artificial neural networks	13
<i>2.1.1 Fundamentals of artificial neural networks</i>	15
<i>2.1.2 Activation functions</i>	17
<i>2.1.3 Network learning and training</i>	19
2.2 Feedforward neural networks	22
2.3 Recurrent neural networks	22
<i>2.3.1 Fundamentals of recurrent neural networks</i>	23
<i>2.3.2 The vanishing gradient problem</i>	25
2.4 Long short-term memory	25
<i>2.4.1 Fundamentals of long short-term memory</i>	26
<i>2.4.2 Relevant applications of long short-term memory</i>	28
<i>2.4.3 Connectionist temporal classification</i>	28
2.5 Convolutional neural networks	29
<i>2.5.1 Fundamentals of convolutional neural networks</i>	31
<i>2.5.2 Prominent convolutional neural network architectures</i>	36
2.6 Transfer learning from pre-trained models	40
2.7 Chapter summary	43

The aim in this chapter is to review the pertinent literature related to deep learning — more specifically, FNNs, RNNs, LSTMs and CNNs. The chapter introduces the reader to principle concepts and terminology found in the deep learning literature so as to facilitate an understanding of the work presented in the remainder of this thesis. The chapter opens with a discussion on ANN fundamentals, activation functions, and network learning and training. This is followed by in-depth discussions of four prominent ANN architectures — *i.e.* FNNs, RNNs, LSTMs and CNNs — during which architecture fundamentals and applications are explored. Additionally, an intuitive explanation of the concept and application of transfer learning — specifically in the realm of computer vision — is considered. A concise summary serves as the chapter’s conclusion.

2.1 Artificial neural networks

Utilising computational power to automate everyday tasks that involve some form of intelligence and/or pattern recognition may be considered markedly difficult [182]. Humans, however, appear

to perform similar tasks without applying significant effort at all. Consider the case where a human brain receives a vast amount of visual information of a surrounding area (including a variety of objects) through their eyes. The human brain is able to transform this visual information into knowledge rapidly, without substantial effort, in order to make an informed decision on the next best action. Accordingly, understanding the approach adopted by humans to effortlessly perform such complex tasks and then simulating these processes on computer systems may lead to a significant increase in automation. Therefore, the emulation of *biological neural networks* by computerised models as a research domain is well warranted.

An ANN may be regarded as a mathematical or computational machine learning algorithm inspired by the substantial parallel computation of a biological neural network and its ability to learn and respond appropriately to presented stimuli [116]. Similar to other machine learning algorithms, ANNs are employed to solve problems through repeated exposure to data and learn without any explicit rule-based programming [251]. A biological neural network forms part of the human nervous system. The term “neural” is the adjective for neuron and the term “network” denotes a graph-like (*i.e.* network) structure [182]. A biological neural network is a collection of more than 10^{10} interconnected biological neurons through sub-networks (*i.e.* nerve cells¹) [1]. A depiction of a biological neuron is illustrated in Figure 2.1.

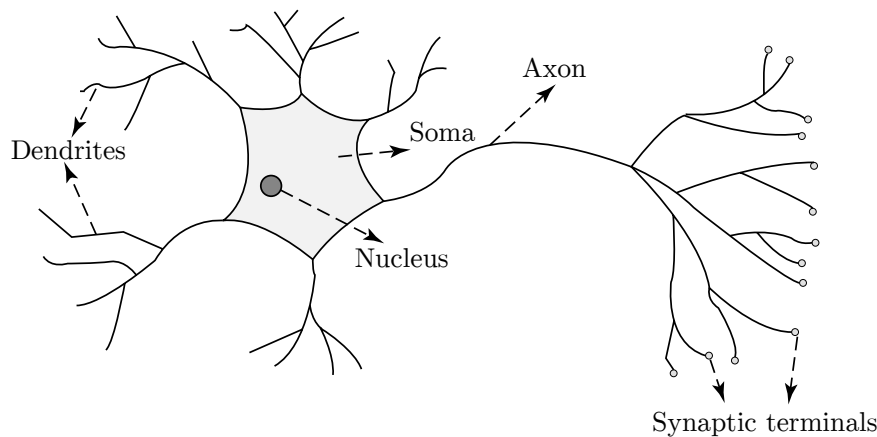


FIGURE 2.1: A graphical representation of a generic biological neuron [1].

Each biological neuron comprises a soma (*i.e.* the cell body), one axon, and a multitude of dendrites [116]. Dendrites are tree-shaped networks of nerve fibres connected to the soma of the biological neuron which receive signals from other biological neurons. Within the soma is the nucleus of the cell. The axon can be considered as a single long fibre which extends outwards from the soma. Eventually, the axon divides into numerous sub-strands which terminates into small end-bulbs called synaptic terminals, as indicated in Figure 2.1 [1]. The gaps between a synaptic terminal and a neighbouring biological neuron’s dendrites are called synapses, and are responsible for facilitating the propagation of signals (*i.e.* information). The number of synaptic connections from one biological neuron to a neighbouring neuron can range from a few hundred to 10^4 [116].

Biological neurons receive, process, and transmit information through biochemical reactions. The magnitude of the incoming signals from the dendrites vary in terms of strength due to the difference in efficiency of the synaptic transmissions [182]. All input signals are aggregated by the soma to form a corresponding output signal. If a sufficient aggregated input signal is received, according to an inherent threshold level, the biological neuron would be stimulated and

¹The fundamental units of the brain and nervous system.

an impulse would be *fired* through its axon. If the threshold level is not met, however, the input signal will quickly decay and result in no further action. This operation is thought to underpin the manner according to which the human brain can store information and react to new input data in order to make cognitive decisions [251].

2.1.1 Fundamentals of artificial neural networks

Mathematically, a biological neural network may be loosely caricatured as a weighted, directed graph of massively parallel interconnected nodes called *artificial neurons* — the basic processing elements of ANNs [116]. Figure 2.2 contains a graphical illustration of a simple mathematical model of the artificial neuron (henceforth merely referred to as neurons). This simple model comprises three rules: Multiplication, summation, and activation [160].

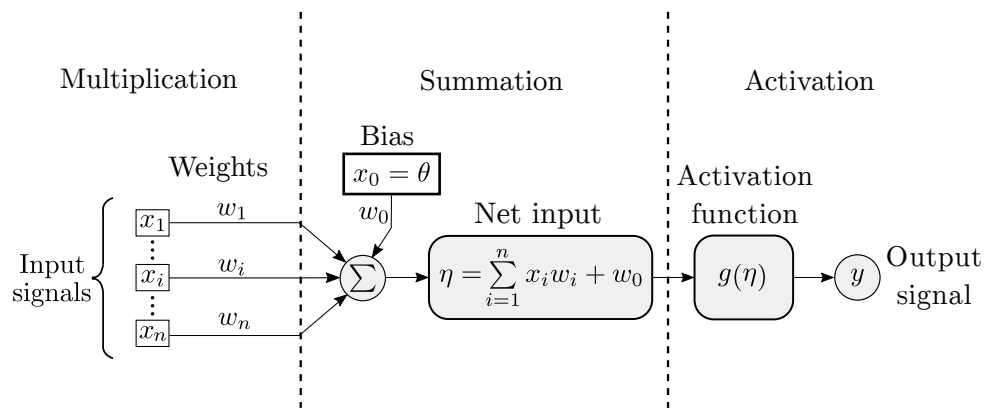


FIGURE 2.2: A simple mathematical model of an artificial neuron.

Within an ANN, neurons are directly connected *via* communication links. The effects of the synapses (as discussed earlier) are represented by the input signals $[x_1 \cdots x_i \cdots x_n]^T$ and the strength of the input signal is scaled by the connection weights $[w_1 \cdots w_i \cdots w_n]^T$, where weight w_i is multiplied by the i -th communication link. The weights are considered to be adjustable and trainable parameters of the network. Given an ANN with randomly assigned weight values, training data within which intrinsic relationships and patterns are embedded, and a task to be accomplished by the ANN, an appropriate *learning algorithm* may be employed in order to *learn* the underlying mathematical representation of the training data, thereby adjusting the weights accordingly. This learning process is discussed in greater detail in §2.1.3. Different weight values will therefore result in different input signals. These scaled input signals are then summed to form an aggregated input signal (denoted by η) [182]. Accompanying η is a bias value ($x_0 = \theta$) and a connection weight w_0 . A bias value is introduced so as to offset the activation function to the left or the right along the input axis, without alternating the shape of the function. Typically, one would set either $w_0 = 1$ or $\theta = 1$, and then adjust the value of the other. If η is positive, the input signal tends to stimulate the neuron output, whereas for a negative aggregated input signal it tends to restrain the neuron output — depending on the selected *activation function* (denoted by g) which mathematically models this firing process. Accordingly, the computation of the neuron's output signal, denoted by y , is

$$y = g(\eta) = g\left(\sum_{i=1}^n x_i w_i + w_0\right). \quad (2.1)$$

The computation of y might seem simple and trivial, however, the true value of this mathematical model is realised when multiple neurons are interconnected to form an ANN. In order to

fully realise the benefits of this interconnected model, the neurons are generally not connected randomly, but are instead structured in a pattern [160]. The pattern according to which the interconnections are arranged within an ANN is referred to as the *architecture* of the network. Several topographies of ANNs have been “standardised” through years of research. Different types of predefined ANN topographies are suited for solving different problems. These different types of ANNs are discussed in more detail later in this chapter. An FNN, which was the first ANN architecture, is illustrated graphically in Figure 2.3 and depicts the general working of most ANNs. An FNN comprises the two elementary elements of any ANN, namely connected neurons and their corresponding weights. Accordingly, the graphical illustration in Figure 2.3 is deemed as a sufficient representation of ANNs and their basic working. The graphical representation comprises neurons represented by the circles (or nodes), and weighted connections represented by the directional arrow lines. The basic architecture of an ANN comprises three types of neuron layers, namely: *Input*, *hidden*, and *output layers* [1].

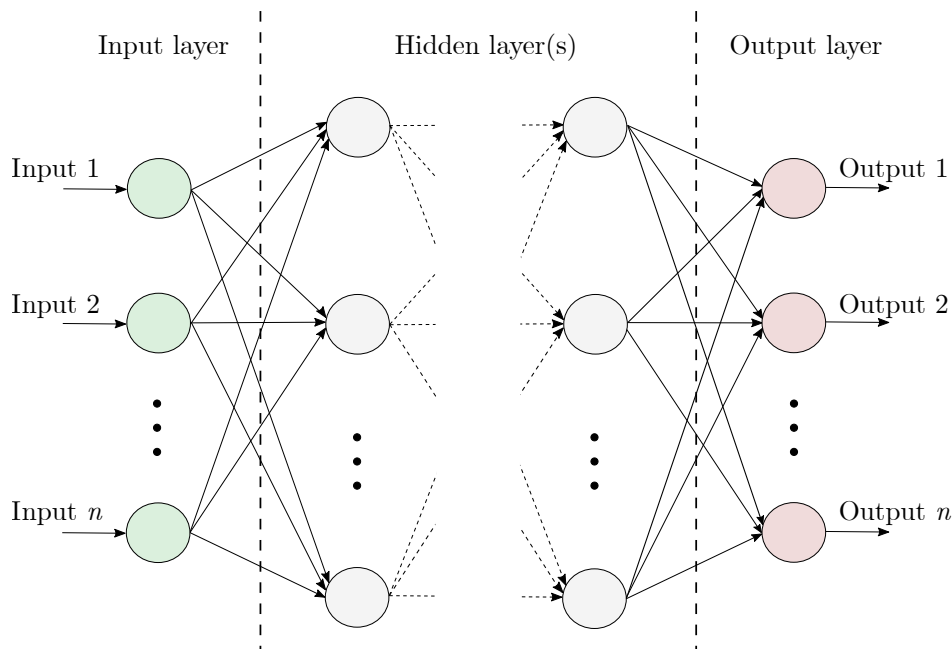


FIGURE 2.3: A graphical representation of the general architecture of an FNN [29].

The signal flow, specifically in FNNs, is strictly in a forward direction, *i.e.* from the neurons in the input layer to the neurons in the output layer (as shown by the directional arrow lines). The input layer receives the selected input variables (*i.e.* an input vector), while the output layer returns a predefined output vector. All the layers between the input and output layers are referred to as hidden layers, comprising *hidden neurons*. ANNs include hidden layers which enable these networks to approximate non-linear and complex functional mappings [297]. Although it is possible for an ANN to have no hidden layers, it would have limited use as it would only be able to classify input data that are linearly separable. In order to determine an appropriate number of hidden layers in an architecture, a trade-off between the closeness-of-fit and the extrapolation² capabilities ought to be considered. A general heuristic for determining a suitable number of hidden layers in an ANN is as follows: As the dimensionality of the problem increases, the corresponding number of hidden layers should also be increased. Furthermore, in order to determine an appropriate number of hidden neurons in each hidden layer, a trade-off between training time and the accuracy of training ought to be considered. If the number of hidden

²A prediction of a value based on extending a known sequence of values beyond the area that is certainly known.

neurons is insufficient, the ANN might have a need for more feature detectors, thereby sacrificing model accuracy. Conversely, if the quantity exceeds the number required by the network, the excessive number of hidden neurons might result in the ANN memorising the training data, thereby producing poor generalisation.

2.1.2 Activation functions

As mentioned in §2.1.1, each neuron in an ANN may be characterised by a weight, bias, and activation function. The purpose of the activation function is to derive a non-linear output for each neuron from the provided input values, thereby modelling the firing process of a neuron mathematically. Consider the case where an ANN is employed without any activation functions. Each neuron simply performs linear transformations by multiplying the input values and bias by their corresponding weight values, and propagate the sum of these values onto the next neuron. Accordingly, the number of hidden layers (and neurons within those layers) would be irrelevant as the composition of multiple linear functions is a linear function itself [21]. Therefore, without any activation functions present, the ANN would simply mimic the learning process of a linear regression model. Incorporating activation functions increases model complexity, however, it enables the model to perform highly complex tasks with greater success. A summary of the most prevalent activation functions is provided in Table 2.1.

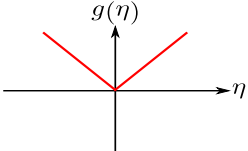
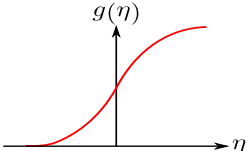
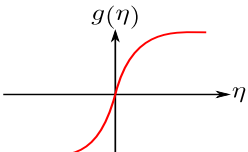
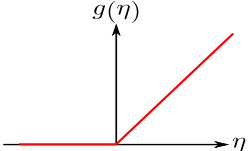
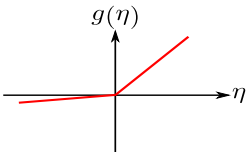
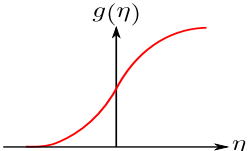
The most basic activation function is the *linear* activation function given by (2.2) in Table 2.1 where λ denotes the slope of the linear function. The function produces an activation that is proportional to the input received. In the special case where $\lambda = 1$, the function is known as the *identity* function. Typically, this function is employed by neurons in the input or output layers of an ANN where the input signal is not altered, but instead simply transmitted to the next layer [116]. The *sigmoid* activation function, given by (2.3), is a non-linear activation function which transforms an input range from $(-\infty, +\infty)$ to the range $[0, 1]$. The function has a smooth S-shape and is continuously differentiable. The sigmoid activation function is commonly (but not exclusively) used in the neurons of an ANN's output layer as this function is employed for predicting probability-based outputs [205]. According to Neal [199], the main advantages of the sigmoid function include the ease with which they may be interpreted, as well as their ability to be successfully applied in shallow network architectures³. The *hyperbolic tangent* activation function, most commonly referred to as the *tanh* activation function, is expressed in (2.4) and shares a similar S-shape as the sigmoid activation function. Unlike the sigmoid activation function, however, the S-shape curve of the tanh function is a zero-centred function with an output range of $[-1, 1]$ [205]. Accordingly, the sign of the output produced by the function might be different to the sign of the input [21]. In some cases, the tanh activation function is preferred over the sigmoid activation function as it may result in increased training performances [12]. Despite being widely adopted in industry, the tanh activation function has a few limitations. If $\eta = 0$, the function attains a gradient of one, resulting in *dead* neurons. If a neuron is dead, its accompanying weight is rarely used or updated [205]. The occurrence of dead neurons is known as the *vanishing gradient* problem. Extensive research has been conducted in order to resolve this significant issue which subsequently led to the development of the *rectified linear unit* (ReLU) activation function.

Nair and Hinton [195] proposed the ReLU activation function in 2010. The usage of the ReLU activation function showcased state-of-the-art results for deep learning applications and is therefore one of the most widely utilised activation functions [205]. The ReLU activation function is given by (2.5). The function performs a threshold operation according to which negative

³Networks that consist of only one or two hidden layers.

values are set to zero, thereby eliminating the vanishing gradient problem. ReLU has been successfully employed in numerous deep learning architectures, which includes CNN architectures. A noteworthy limitation of the ReLU activation function is its relatively poor generalisation performance in certain cases when compared with some of the aforementioned functions [205]. Moreover, the most significant limitation of ReLU is that in the case where a neuron only receives negative input values, the activation function transforms all these input values to zero, thereby transforming the neuron's output to zero. This is a special case of the vanishing gradient problem [219]. Various analyses have been conducted in order to improve upon ReLU, and variants of ReLU have been developed. The *leaky ReLU* (LReLU) activation function was proposed in 2013 and is expressed in (2.6). LReLU is markedly similar to ReLU, but with the addition of introducing an α parameter which provides a small slope for negative η values. The

TABLE 2.1: Summary of prevalent activation functions.

Name	Plot	Equation	Range
Linear		$g(\eta) = \lambda\eta$ (2.2)	$(-\infty, +\infty)$
Sigmoid		$g(\eta) = \frac{1}{1 + e^{-\eta}}$ (2.3)	$[0, 1]$
Tanh		$g(\eta) = \frac{e^{\eta} - e^{-\eta}}{e^{\eta} + e^{-\eta}}$ (2.4)	$[-1, 1]$
ReLU		$g(\eta) = \begin{cases} 0, & \text{if } \eta < 0 \\ \eta, & \text{if } \eta \geq 0 \end{cases}$ (2.5)	$[0, \infty)$
LReLU		$g(\eta) = \begin{cases} \alpha\eta, & \text{if } \eta < 0 \\ \eta, & \text{if } \eta \geq 0 \end{cases}$ (2.6)	$(-\infty, +\infty)$
Softmax		$g(\eta) = \frac{e^{\eta_i}}{\sum_{j=1}^K e^{\eta_j}}$ (2.7)	$[0, 1]$

purpose of this slope is to ensure that the gradients are never equal to zero during training. Another prevalent variant of ReLU is called the *parametric ReLU* (PReLU), proposed by He *et al.* [307] in 2015. PReLU is almost identical to LReLU with the only point of difference being that the α variable in the PReLU activation function is a learnable parameter which is adaptively learnt during the training of the network. In the case where $\alpha = 0$, PReLU becomes ReLU. According to He *et al.*, performance on large scale image recognition improved when PReLU was employed over ReLU. The final noteworthy activation function is the *softmax* activation function, given by (2.7). The softmax activation function is employed to compute a probability distribution from a vector of real numbers. Accordingly, the output range is $[0, 1]$ [90]. The softmax is commonly employed in the output layer of a network and when the network is tasked with computing probabilities for classification problems — *i.e.* the main differentiating factor between the softmax and sigmoid activation functions [205].

2.1.3 Network learning and training

The ability to learn is perhaps the fundamental trait of intelligence. Unlike rote learning⁴, the ability to learn focusses on the difficulty of extracting inferences from accumulated information in order to generalise a behaviour when a novel situation arises [166]. Specifically for deep learning networks, the notion of network learning and training is defined as the process of adjusting weight values between neurons so as to enable the network to learn intrinsic patterns embedded within the input data in order to produce the desired set of outputs according to some update rule [1]. When training a model, it is important to know the nature of the model environment in which the network must carry out some task — *i.e.* what information is available to the network in order to perform the given task. ANNs may be classified into four main learning paradigms, namely *supervised learning*, *unsupervised learning*, *semi-supervised learning*, and *reinforcement learning*.

Supervised learning aims to map a set of input variables to an output variable, and thereafter apply this mapping to predict outputs in respect of unseen data [57]. The term *supervised* originates from the concept of an external teacher that provides the desired outputs that correspond to presented inputs. Specifically in the realm of ANN learning, an input vector (*i.e.* features) is presented at the input layer of the network, while a set of desired responses (*i.e.* corresponding labels or outcomes), one for each neuron, is provided to the output layer of the network. After a forward pass is performed by the network, the discrepancies between the output of the network and the actual label is determined, and thereafter the network weights are updated according to some learning rule [1]. Unsupervised learning simply receives an input vector, but does not obtain corresponding labels for the neurons in the output layer (*i.e.* unlabelled data) [86]. The primary objective of unsupervised learning is to train the network to discover statistically salient features so as to identify clusters or patterns within the input vector. Moreover, in contrast to supervised learning, there is no *a priori* set of categories for the input vector to be classified as. Therefore, the network ought to develop its own representation of the input vector. In the semi-supervised learning paradigm, the network aims to combine supervised and unsupervised learning by utilising labelled data as well as unlabelled data when performing a learning task. Typically, semi-supervised learning algorithms would perform one of these learning methods as its primary method, and then attempt to improve the results by utilising information associated with the other method. This paradigm is particularly relevant when the input is large, and the corresponding labels are scarce [292]. Lastly, within the reinforcement learning paradigm, the network is taught by trial-and-error to perform an action in order to be *rewarded*. This paradigm

⁴Simply memorising presented information without drawing inferences from the information [166].

of learning is related to a problem set where the best action is not known *a priori*, and therefore the network ought to develop an action policy in order to maximise the received reward.

Supervised learning algorithms are widely used in the industry for making predictions in the realms of *regression* and *classification* problems. Regression algorithms are employed in order to predict a continuous value based on the provided input data [108]. Accordingly, the primary goal of a regression problem is to approximate a function that best maps a set of inputs to a continuous (quantitative) output. Examples of regression target variables are weight, costs, height, income, or inventory level. Classification algorithms, on the other hand, are employed in order to approximate a mapping function that best classifies input data in respect of a discrete output variable, such as categories or labels [32]. Examples of classification target variables are dog breeds, plant species, and sports. It is important to consider the class distribution when training a classification model [255]. A data set is considered to be *imbalanced* when there is skewness in the output data, *i.e.* when the minority class is notably outweighed by the majority class. The bias in the training data might influence the training of the machine learning algorithm, resulting in the model disregarding the minority class entirely. Accordingly, it might be beneficial to balance the data set. The implementation of *random sampling* may be considered in order to balance a data set [310]. The two main approaches of random sample are *oversampling*, whereby instances of the minority class are randomly duplicated in order to increase the minority class, and *undersampling*, whereby instances of the majority class are randomly removed in order to reduce the majority class. Although rebalancing the data set may be beneficial for model training, it is important to note that oversampling tends to result in overfitting, while undersampling may result in a loss of valuable information [310].

In terms of training an ANN in the learning paradigm of supervised learning, an *epoch* passes when the entire training data set has been evaluated by the model once [180]. It is common for the training data set to be too big for all the data to be presented to the model simultaneously [253]. Accordingly, the data set is then divided into *batches*. The *batch size* refers to the number of training examples within a batch. The number of batches required to complete a single epoch is called an *iteration*.

Training a network comprises updating the network weights by comparing the network output with a respective desired label. A gradient-based technique is typically employed and is rendered computationally feasible by the method of *backpropagation* [58]. Simply put, backpropagation is a derivative-based approach towards adjusting the network weights based on some *error* (*i.e.* loss) obtained after evaluating a batch of the training data. In order to elucidate the workings of backpropagation, consider the case where a supervised learning task is presented to an FNN (graphically represented in Figure 2.4).

In order to initialise the model before training commences, the network weights are assigned random values. The input data are then passed from the input layer, to the hidden layers, and then finally on to the output layer. This is known as *forward propagation*. At each neuron, the weighted sums of the input signals (and bias) are passed through the activation function. The activation value is passed to the next neuron (*via* the next set of weighted connections) until all the signals reach the output layer. The output produced by the neurons in the final layer represents the prediction of the network. This prediction may then be compared with the corresponding label (or target value) of the given input vector, and a loss/cost may be calculated according to some preselected *loss function* — the aim is to minimise the computed loss.

In the paradigm of supervised learning, there are two main types of loss functions, namely regression loss functions and classification loss functions. Examples of prominent regression loss functions include *mean absolute error* (MAE) [44] which aims to compute the absolute differences between the target and the predicted output, *mean squared error* (MSE) [241] which aims to

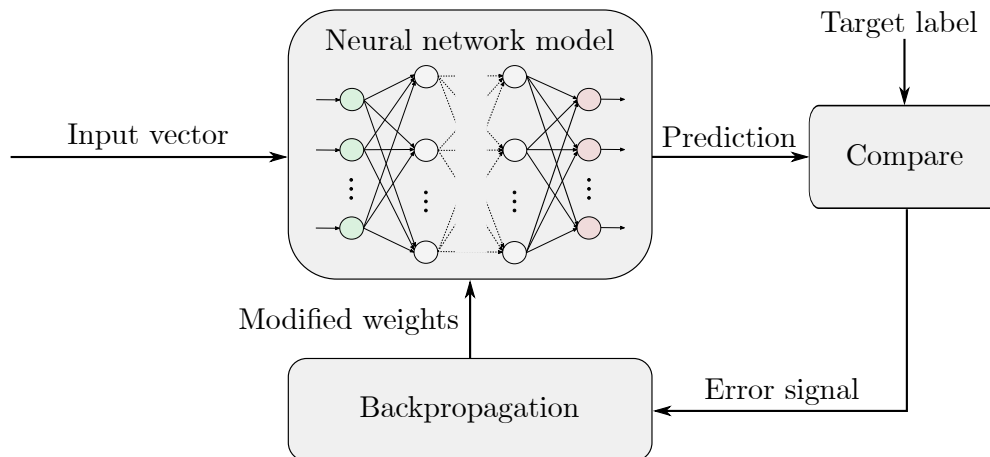


FIGURE 2.4: A graphical representation of the ANN learning process in the supervised learning paradigm.

compute the average of the squared differences between the target and the predicted output, and *root mean squared error* (RMSE) [44] which aims to compute the average difference between the target and predicted output using the *Euclidean distance* [59]. Examples of prominent classification loss functions include *binary cross-entropy* [122] where only two pre-set categories are considered and the actual value (0 or 1) is compared with the probability that the input aligns with that category, and *categorical cross-entropy* [158] where the number of classes considered is greater than two, following a similar process as binary cross-entropy. After computing the loss through the employment of a selected loss function, the value is then fed backwards in the network in order to fine-tune the network weights.

A network may comprise millions of weight parameters. A so-called optimiser (*i.e.* training algorithm) is employed in order to govern the weight adjustment process computationally and (in certain cases the *learning rate* of the model) in pursuit of minimising the calculated loss function. The learning rate is a configurable hyperparameter that can be used to control to what extent and rate the model weights are updated [33]. Several optimiser algorithms have been developed for different applications. The *gradient descent* [11] optimisation algorithm is one of the most commonly employed optimisation algorithms to train deep learning models. The algorithm is initialised with some random weights, partial derivatives are then calculated of the loss in respect of each network weight for the entire training data set, and then the weights are adjusted accordingly (*i.e.* in the negative direction of the gradient) in order to yield a smaller loss value.

There are three primary types of gradient descent optimisers, namely *batch gradient descent* (BGD) [142], *stochastic gradient descent* (SGD) [232], and *mini-batch gradient descent* [152]. The BGD optimiser (also called *vanilla gradient descent*) is employed to compute the gradient for each point in the training data set, updating the weights only after the evaluation of the entire data set. BGD is known to produce a stable error gradient and a stable convergence. By contrast, the SGD optimiser, referred to as a probabilistic approximation of gradient descent [23], stochastically selects (sampled without replacement) and computes the gradient for each data point and updates the weights for each training example one-by-one. SGD has been empirically shown to speed up convergence, however, some instability is introduced [66]. The mini-batch gradient descent optimiser combines the principles of the aforementioned optimisers, partitioning the training set into small batches. The gradient is then computed for a batch, and the weights updated according to the smaller loss value. This process is then repeated for each batch.

The *Adam* [155] optimiser (the name of which is derived from *adaptive moment estimation*) is considered to be a further extension of the SGD optimiser. The key difference between the SGD and the Adam optimisers is that while the SGD optimiser maintains a single learning rate throughout training, the Adam optimiser updates the learning rate for each network weight individually. The Adam optimiser is considered to be a benchmark optimiser as it is easy to implement, it is computationally less demanding, and requires less fine-tuning than other optimisers. Although the Adam optimiser performs well in most cases, algorithms such as the SGD optimiser may sometimes generalise better based on empirical findings [107].

After all training iterations have been evaluated and the weights adjusted according to the adopted optimiser, the next epoch may commence, thereby repeating this pattern typically until either the loss is below a prespecified threshold, or a maximum number of epochs have been reached [187].

2.2 Feedforward neural networks

There are several types of ANN architectures, each utilising different principles in order to determine a set of learning rules. These network types are implemented based on the availability of data and the mathematical operations required to perform a task. Each type of network has unique strengths which enable them to operate in particular domains and perform specific tasks. The following few sections are devoted to a discussion on the fundamental differences between some of the most prominent ANN types, namely FNNs, RNNs, LSTM, and CNNs.

In §2.1.1, specific reference was made to FNNs (graphically illustrated in Figure 2.3) and several concepts of this particular network were discussed. FNNs are arguably the most basic type of ANN as it solely comprises standard interconnected neurons, categorised into the three aforementioned layers. The defining and main differentiating feature of FNNs is that the cyclic flow of information (*i.e.* the presence of *feedback* connections or loops) is strictly forbidden — data move in a forward direction from the first layer onwards until it reaches the output layer [174]. Due to the absence of any feedback connections, all the neurons in the network may be arranged into the three different types of configured layers, which renders the network easily understandable.

There are multiple variants of FNNs, each identifiable by their unique structural differences. An FNN comprising only a single hidden layer is referred to as a *single-layer feedforward neural network*, whereas an FNN with more than one hidden layer is referred to as a *multi-layer feedforward neural network*. In the remainder of this thesis, multi-layer feedforward neural networks are referred to as *multi-layer perceptrons* (MLPs). Due to its layered structure, an MLP showcases *hierarchical* processing, passing information on from one layer to another, until the last layer provides an output. MLPs may also be described as fully connected networks as each neuron is connected to every neuron in both the previous and following layer [82]. The MLP is the most widely utilised FNN variant due to its universal approximation capabilities [116].

2.3 Recurrent neural networks

Several ANN structures have been proposed in the literature to perform tasks related to temporal patterns. Examples of tasks where these temporal sequences occur are speech recognition, dynamic control systems, and time series predictions. In some cases, MLPs cannot be utilised in time-varying tasks as they are limited to only receiving static data patterns (*i.e.* an input

vector with a predefined dimensionality); therefore, MLPs require a fixed input/output window size in advance [249]. Examples of different variable input/output combinations are graphically illustrated in Figure 2.5, where the bottom green rectangles represent the input vectors, the middle grey rectangles represent the network, and the top red rectangles represent the output vectors.

Note that the term *many* is not a fixed number for each input or output of the networks. From left to right, the first example is a one-to-one scenario where a fixed-size input vector is received and a fixed size output vector is predicted, as is the case for MLPs. Examples two to five, however, all have a varying input/output vector present, which is difficult for MLPs to process. Examples of real-world tasks of these scenarios are:

- One to many: Text generation, image captioning,
- many to one: Text classification, sentiment analysis, and
- many to many: Voice recognition, machine translation.

Consequently, an alternative ANN architecture that can incorporate temporal dynamics of time varying-patterns is necessary.

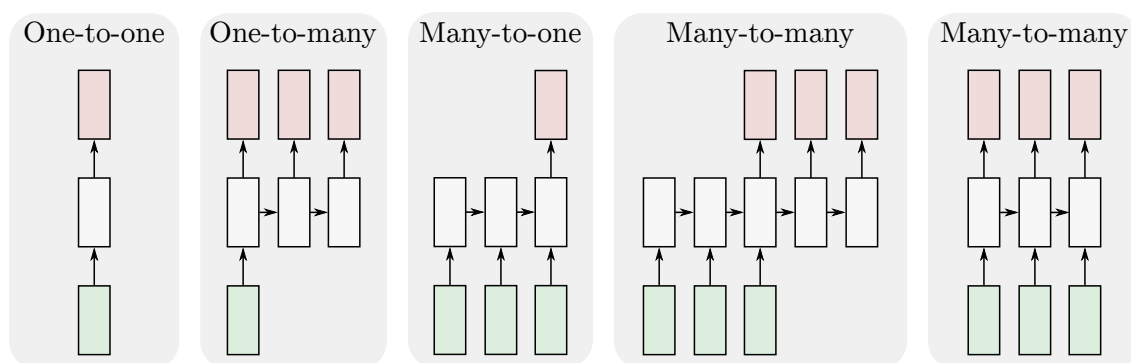


FIGURE 2.5: Examples of varying inputs/outputs combinations [148].

2.3.1 Fundamentals of recurrent neural networks

In 1986, Rumelhart *et al.* [233] proposed the RNN architecture, designed specifically for performing time-varying tasks. An RNN introduces the notion of time by augmenting an FNN such that it includes a *recursive* processing unit with a hidden state derived from the past [174]. While the primary function of an FNN’s layers is hierarchical processing, the layers of an RNN introduce *sequential memory* processing. Each generation of the network attempts to *remember* as much information from the past as possible in order to produce more accurate predictions of the future. The network achieves this by containing at least one *feed-back connection* — the fundamental feature of an RNN — enabling activation signals to be fed back [36].

In order to explain how forward propagation in an RNN is performed, consider the basic RNN architecture (with a recursive pointer to itself), as shown in Figure 2.6, where x^t , y^t , and h^t denote the input, output, and hidden states, respectively, at a time step t [309]. Applied to the hidden state are the weight parameters U , W , and V .

In order to elucidate the workings of a basic RNN, consider the case where an RNN is tasked with predicting a letter in the word “car”. The input vector is the first two letters *i.e.* “c” and

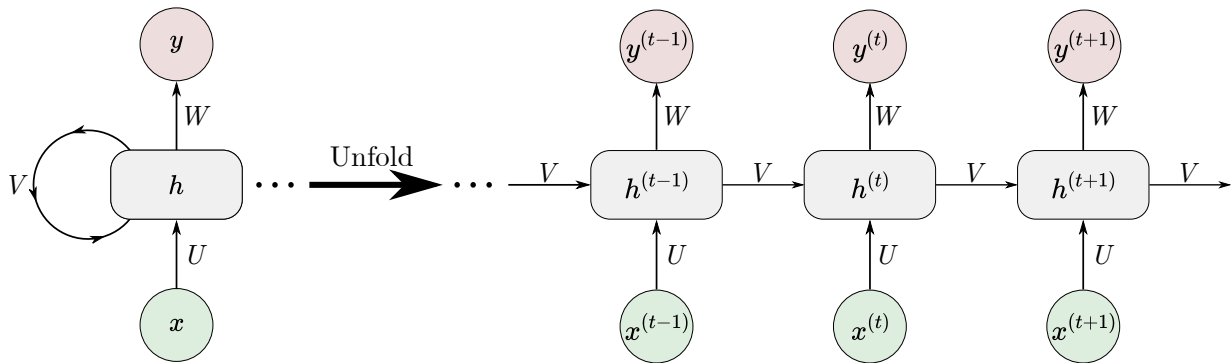


FIGURE 2.6: The basic RNN architecture with a self-looping or recursive pointer to itself [309].

“a”, and the network is tasked with predicting the last letter *i.e.* “r”. In order to predict the third letter in the target word, the hidden state applies a recurrence formula to the input vector. When the letter “a” is supplied to the network, the recurrence formula is applied to the letter and the previous state, *i.e.* “c”. The states are referred to as time steps. At time step t , the input letter is “a”, and at time step $(t - 1)$ the input letter was “c”. The recurrence function f for the current state may be written as

$$h^{(t)} = f\left(h^{(t-1)}, x^{(t)}\right), \quad (2.8)$$

where $h^{(t)}$ denotes the new state, $h^{(t-1)}$ denotes the previous state, and $x^{(t)}$ denotes the current input. Assuming the tanh activation function is selected for the hidden state, a weight value V for the hidden state connection, and a weight value U for the input neuron connection, the expression for the state at time t may be expressed as

$$h^{(t)} = \tanh\left(Vh^{(t-1)} + Ux^{(t)}\right). \quad (2.9)$$

By incorporating the hidden state connection and weight into the expression, the value of the previous state is taken into consideration when calculating the new state. For sequences comprising a large number of characters, multiple states can be calculated and *remembered* throughout the network. Once the current state is calculated, the output state may be calculated as

$$y^{(t)} = Wh^{(t)}, \quad (2.10)$$

where W is the connection weight between the hidden state and the output neuron. By applying the softmax activation function in the final output neuron, the classwise probabilities for the next letter can be calculated.

If the hidden state is *folded up*, it has a similar structure to that of a regular MLP, and the backpropagation algorithm performs a similar operation to when it is applied to an MLP. When performing forwardpropagation, the information flows forward through the network and at each time step. The algorithm, *backpropagation through time* (BPTT), was introduced by Werbos [302] in 1990 and is utilised to modify an RNNs connection weights, thereby training the network. The algorithm adopts its name due to its movement back in time through the time steps to modify the hidden state connection weights. The BPTT learning algorithm extends naturally from the standard backpropagation algorithm utilised in the training of MLPs [36]. Consider the case where $y^{(t)}$ is the predicted value and $\bar{y}^{(t)}$ is the true label. The error signal can be calculated as a cross-entropy loss E given by the expression

$$E(\bar{y}, y) = -\sum \bar{y}^{(t)} \log(y^{(t)}), \quad (2.11)$$

and by treating the entire sequence (or in this case, the entire word) as a single training example. Accordingly, the total error for the training example is the sum of all the errors at each time step (*i.e.* character). This error signal is then propagated backwards through the network, visiting each time step in sequence, combining the backpropagation result and modifying the connection weights accordingly.

2.3.2 The vanishing gradient problem

It is important to note that the training of RNNs has a major limitation. According to Bengio *et al.* [22], and later expanded upon by Hochreiter *et al.* [123], training RNNs has long been considered to be a markedly challenging task due to the difficulty of learning long-range dependencies. When performing BPTT across many time steps, the gradients of the early time steps might begin to *vanish* or *explode*. This phenomenon is known as the *vanishing gradient problem* [174].

Consider a network that comprises only a single input node, a single recurrent hidden node, and a single output node. The unfolded recurrent hidden node is visualised in Figure 2.7. An input value β is presented to the network at time step $(t-5)$ and an error is calculated at time step t , assuming that the input of all the intervening time steps is zero. The recurrent connection weight V is the same between all time steps. Consequently, as the network steps through the time steps, the contribution of the input β at time step $(t-5)$ either exponentially vanishes or explodes as it approaches time step t . Whether the gradient vanishes or explodes depends on which activation function is selected for the hidden node and whether the connection weight value is $|V| > 1$ or $|V| < 1$. In Figure 2.7, the specific phenomenon of a vanishing gradient is illustrated where the contribution of the input β is represented by the intensity of the green shading in the nodes. Note how the contribution of the input β declines for each time step as the network traverses to time step t . Williams and Zipser [304] proposed the *truncated backpropagation through time* (TBPTT) learning algorithm as a possible solution to the vanishing gradient problem. The TBPTT limits the number of time steps the error signal may be propagated back to, according to a predefined threshold. By truncating the learning algorithm, however, the network's ability to learn long-range dependencies is compromised [174].

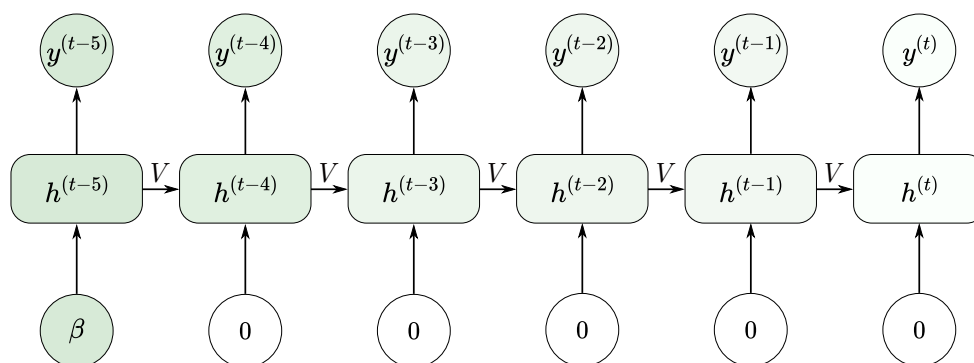


FIGURE 2.7: The vanishing gradient problem visualised in the hidden state of a basic RNN.

2.4 Long short-term memory

In order to overcome the shortcomings of RNNs (alluded to in §2.3.2), Hochreiter and Schmidhuber [124] introduced the LSTM architecture. The network is named according to the following

intuition: The basic RNN has *long-term memory* through the modification of the connection weights, and *short-term memory* in the form of *ephemeral*⁵ activations, which are passed between successive nodes. LSTMs introduce a third type of memory, an intermediate type of storage, through the employment of recurrently connected sub-networks known as *memory cells* [174]. The purpose of a memory cell is to remember values over arbitrary time intervals by utilising a *constant error carousel* (CEC), which holds the error signal within each cell. The network resembles the basic RNN architecture which comprises a hidden state, however, each hidden state is replaced by a memory cell.

2.4.1 Fundamentals of long short-term memory

A basic LSTM memory cell *block* (depicted in Figure 2.8) comprises an input node z , an input gate i , an output gate o , a forget gate f , and an output node k . The flow of information associated with the cell is regulated by the three gates. The input signal is represented by $x^{(t)}$ and the output signal by $h^{(t)}$, while $h^{(t-1)}$ is the output of the previous LSTM layer. The σ represents the employment of a sigmoid activation whereas τ represents the employment of a tanh activation function. The sigmoid activation function is always used as a gating function at the input, forget, and output gates as it outputs a value between 0 and 1, determining whether information may flow through the gate. It is customary to use the tanh activation function for the input and output nodes as its second derivative has so-called long dynamic range, however, in some cases the ReLU activation function may also be used [174]. The dotted blue connections are *peephole* connections, which allow the block to inspect its current internal state, while \odot denotes point-wise multiplication [84]. The current cell state is represented by $c^{(t)}$ (*i.e.* the CEC component), while the previous LSTM layer's cell state is represented by $c^{(t-1)}$.

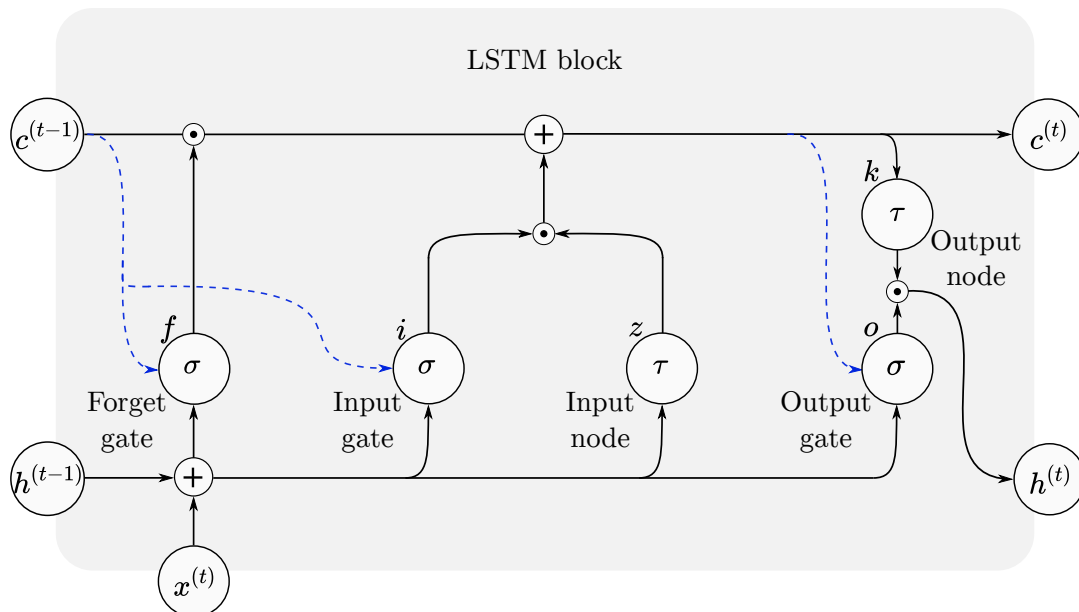


FIGURE 2.8: The architecture of a typical LSTM memory cell block [293].

In order to delineate the workings of an LSTM network, consider a network that comprises N LSTM memory cell blocks and M inputs. Forward propagation begins at the input node z where

⁵The concept of objects being transitory, existing only briefly.

the current input, $x^{(t)}$, and the output of the previous LSTM layer, $h^{(t-1)}$, is combined, yielding

$$z^{(t)} = \tau \left(U_z x^{(t)} + V_z h^{(t-1)} + b_z \right), \quad (2.12)$$

where U_z and V_z are the connection weights associated with $x^{(t)}$ and $h^{(t-1)}$, respectively, while b_z is a bias value associated with the input node. Thereafter, the input gate i is updated with the combination of $x^{(t)}$, $h^{(t-1)}$, and the previous cell state $c^{(t-1)}$ with their respective connection weights of U_i , V_i , and p_i , and the bias b_i of the component, through the expression

$$i^{(t)} = \sigma \left(U_i x^{(t)} + V_i h^{(t-1)} + p_i \odot c^{(t-1)} + b_i \right). \quad (2.13)$$

Through the process of updating the input node z and the input gate i , the LSTM layer determines which information should be retained in the cell state $c^{(t)}$. Subsequently, the LSTM needs to determine which information should be removed from the previous cell state $c^{(t-1)}$. This operation is performed at the forget gate.

The forget gate was proposed by Gers *et al.* [85] in order to enable the network to reset its state by disregarding information which is not required by the network for learning. The forget gate f , is calculated based on the current input $x^{(t)}$, the previous LSTM layer's output $h^{(t-1)}$, and the cell state $c^{(t-1)}$, their respective accompanying weight values U_f , V_f , and p_f , and the associated biased value b_f of the forget gate f . This calculation may be expressed as

$$f^{(t)} = \sigma \left(U_f x^{(t)} + V_f h^{(t-1)} + p_f \odot c^{(t-1)} + b_f \right). \quad (2.14)$$

Thereafter, the new cell state $c^{(t)}$ may be calculated by combining the input node z , the input gate i , and the forget gate f , yielding

$$c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)}. \quad (2.15)$$

The output gate o may then be calculated by combining the current input $x^{(t)}$, the previous output $h^{(t-1)}$, and the previous cell state $c^{(t-1)}$. This may be expressed as

$$o^{(t)} = \sigma \left(U_o x^{(t)} + V_o h^{(t-1)} + p_o \odot c^{(t-1)} + b_o \right), \quad (2.16)$$

where U_o , V_o , and p_o are the connection weights associated with $x^{(t)}$, $h^{(t-1)}$, and $c^{(t-1)}$, respectively, while b_o denotes the bias for the output gate. In order to calculate the output signal $h^{(t)}$, the output node k and the output gate o is used which yields

$$h^{(t)} = \tau \left(c^{(t)} \right) \odot o^{(t)}. \quad (2.17)$$

Comprising three different logical units of memory, an LSTM layer processes different dimensions of the information during the forward propagation stage, thereby improving the long and short term memory capabilities of the network [309]. Intuitively, the LSTM network may decide when activation is permitted into the cell state $c^{(t)}$ (*i.e.* through the input gate) and, similarly, decide when to let this activation out (*i.e.* through the output gate). In the case where both gates are closed, the activation is trapped within the block, neither growing nor shrinking. Consequently, if this case occurs, the input $x^{(t)}$ has no effect on the output at intermediate time steps.

Similar to an RNN, the different layers of an LSTM have shared connection weights. Graves and Schmidhuber [103] proposed using BPTT in order to train the weights that connect the different network components. During backpropagation, an accumulation of gradients, denoted by $\Delta^{(t)}$, is received by the cell state $c^{(t)}$ which comprise gradients from both the next cell state $c^{(t+1)}$ and the output signal $h^{(t)}$. In the case of the error signal being denoted by E , $\Delta^{(t)}$ corresponds to $\partial E / \partial \mathbf{y}^{(t)}$. The $\Delta^{(t)}$ accumulation is then propagated back throughout the LSTM layers, modifying the weights accordingly.

2.4.2 Relevant applications of long short-term memory

The LSTM architecture is incorporated in a wide array of modern studies and industry domains, propelled by its ability to effectively capture long-term temporal dependencies [105, 124]. LSTM is markedly well suited for time-series predictions. Fischer and Krauss [78] applied LSTMs in order to make financial market predictions. Due to its non-linearity and non-stationarity, forecasting financial data presents a significant challenge to researchers and practitioners. From their study, Fischer and Krauss showcased that LSTMs outperform other traditional machine learning algorithms. In the context of predicting petroleum production, Sagheer and Kotb [237] confirmed the superiority of LSTMs when they stacked multiple LSTM blocks in a hierarchical fashion and increased their model's ability to process temporal tasks when compared with traditional methods. Another example of LSTMs in time series predictions is predicting the remaining useful life of physical systems. Elsheikh *et al.* [72] proposed a new *bidirectional* LSTM (BLSTM) architecture in order to predict the remaining life of production resources. BLSTM enables a layer to receive information from the past and future, simultaneously.

In the field of speech recognition, a subfield of NLP, Graves *et al.* [100] were the first to utilise LSTMs, as the network can handle long-term lags markedly well. In another NLP example, Ryu *et al.* [235] developed a binary classifier that was trained with *in-domain* data, and could detect *out-of-domain* sentences. Former state-of-the-art models could not reach the high accuracy achieved by the LSTM model. Moreover, LSTMs may also be used to detect dialogue breakdowns, as LSTMs possess the ability to remember long-term context. Takayama *et al.* [276] considered various network architectures in pursuit of detecting dialogue breakdowns and concluded that unidirectional LSTMs (BLSTMs that are trained twice through forward and backward directions) produced the best results.

LSTMs also add significant value to the field of computer vision applications. Kafle and Kanan [145] as well as Gao *et al.* [80] applied LSTMs, in combination with computer vision techniques, to visual answering tasks. The intuition was that if the network can understand questions and answers, then a system can be developed to receive text-based questions about an image in order to infer an answer. Moreover, in the NLP space, Kanjo *et al.* [147] were the first to utilise LSTMs in combination with CNNs in sentiment analysis in order to extract information from multi-modal data (*i.e.* using physiological, environmental and location data) in order to recognise emotions. According to the authors, the accuracy level increased by 20% by employing the hybrid network.

Another field in which LSTMs proved to be useful, is in OCR applications [17, 30, 264]. Naz *et al.* [198] conducted a text recognition study in which a sliding window is employed over text line images for feature extraction. The resulting vector is then fed into a multi-dimensional LSTM-CNN hybrid network. The authors concluded that utilising LSTMs, specifically in hybrid deep learning architectures, produced significantly improved accuracy on benchmark problems when compared with traditional methods.

2.4.3 Connectionist temporal classification

RNNs and LSTMs may be used for sequence labelling tasks such as speech recognition or text recognition [248]. Several methods have been developed for the training of sequence labelling models, whereby *connectionist temporal classification* (CTC), introduced by Graves *et al.* [101], has proven to be the most widely adopted method [175]. The steps employed by the CTC method facilitates end-to-end model training while not requiring any pre-defined alignment information.

The expected output is a matrix containing character probabilities for each time-step, mapped to the final text.

Consider the input sequence $X_{1:T}$ of length T , provided to a model. The model then predicts a sequence $y^{1:T}$ with y^t denoting the probability vector of observing labels over the fixed-length alphabet L' . The fixed-length alphabet, $L' = L \cup \emptyset$, comprises each of the pre-defined labels, whereby a blank label \emptyset is included at the time-step t . All observed time-steps may be concatenated into a single path, denoted by π . In order to achieve the desired sequence labelling training, the relationship between π and the target sequence l is sought after. The CTC framework attains access to this relationship by defining a many-to-one mapping operation, denoted by β . First, the mapping operation removes non-unique labels, followed by the removal of blanks within the provided path. Feasible paths may then be defined as π mapped onto l through the implementation of β . The sum of probabilities of all the feasible paths is known as the *conditional probability* of a given l , defined as

$$p(l | X_{1:T}) = \sum_{\pi \in \beta^{-1}(l)} p(\pi | X_{1:T}), \quad (2.18)$$

and where the probability of π is given as

$$p(\pi | X_{1:T}) = \prod_{t=1}^T y_{\pi_t}^t, \forall \pi \in (L')^T. \quad (2.19)$$

In pursuit of model training, the CTC method attempts to optimise the loss function utilising dynamic programming to efficiently sum all the feasible paths. The time complexity of this method is $\mathcal{O}(TW^2)$, where T denotes the sequence length and W the size of the provided dictionary [99].

2.5 Convolutional neural networks

The aim of computer vision is to derive information from images, videos, and other visual inputs by means of computational methods. A computer cannot see visually as humans do, therefore visual information ought to be converted into an appropriate format readable by computers. Mathematically, an image may be expressed as a matrix of pixel values, where each value represents the characteristics of the pixel. Consider the greyscale⁶ image of the number eight, graphically illustrated in Figure 2.9. Each pixel on the original image (Figure 2.9(a)) may be converted into a number between 0 and 255, where 0 corresponds to pure black and 255 corresponds to pure white. The overlay of the image pixels and the pixel values is showcased in Figure 2.9(a). The pixels may then be removed, leaving only an array of numbers (Figure 2.9(a)) which can be processed by a computer.

In order to utilise this pixel value information in machine learning tasks, one may consider transforming the matrix into a one-dimensional array or vector, by simply “flattening” it. This process is demonstrated in Figure 2.10, which depicts the transformation of a 3×3 matrix into a 1×9 vector. This vector is then subsequently fed as an input into a MLP. This method, however, only yields an average accuracy score when applied to basic binary images and furthermore yields little to no accuracy when applied to large and complex images having thousands of pixel dependencies throughout [238]. Therefore, an alternative ANN architecture ought to be considered, specifically designed to handle spatial and pixel dependencies.

⁶A range of grey shades from white to black.

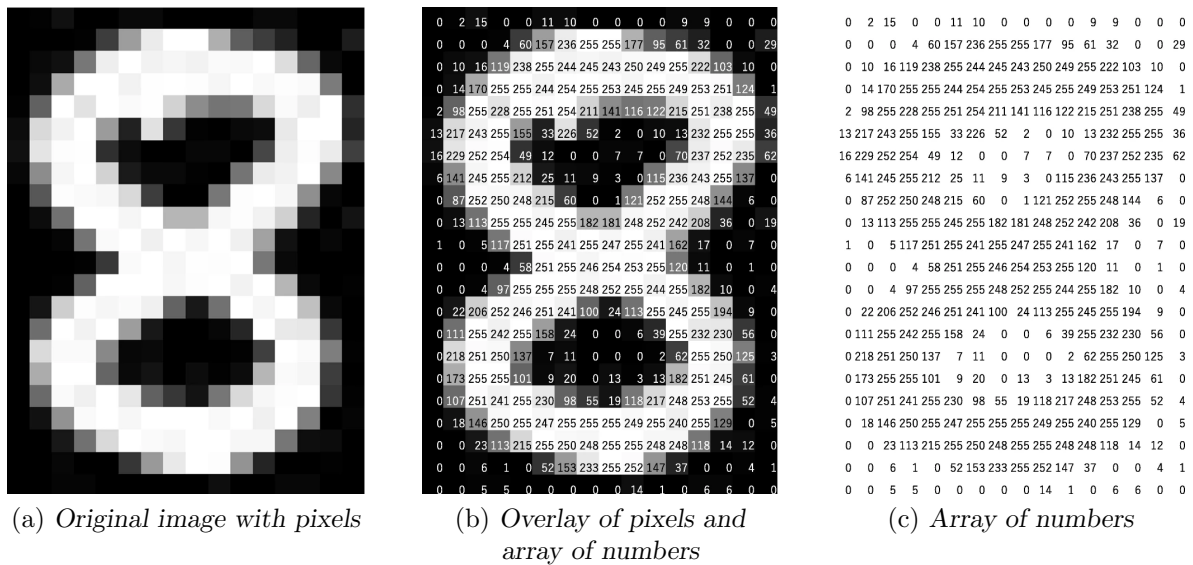


FIGURE 2.9: A graphical illustration of a greyscale image of the number eight where (a) represents the image in the form of a pixels, (b) represents the image in the form of an overlay of pixels and an array of numbers, and (c) represents the image solely in the form of an array of numbers [293].

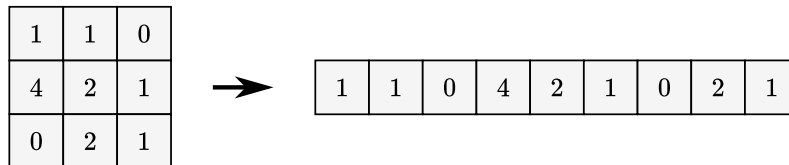


FIGURE 2.10: A simple 3×3 matrix flattened into a 1×9 vector [238].

The CNN architecture was first introduced by LeCun and Bengio [168] in the 1980s. Analogous to the connectivity pattern of neurons in the human visual cortex, the CNN architecture is designed to *mimic* the collection of overlapping subfields in the human brain. Each of these subfields only respond to stimuli belonging to a specific area, called the *receptive field*. In terms of CNNs, this is achieved by utilising *local connectivity* between neurons, and transforming the input in a style that is hierarchically organised [185]. A CNN layer comprises a set of neurons and several *filters* (also referred to as *convolutional kernels*⁷) within the layer. Each filter comprises multiple weight values which are the trainable parameters of a CNN. The nodes produce outputs in the form of groups of d -dimensional arrays, known as feature maps, and are used as inputs for the next layer. A node solely receives the input from a specific *window area* of the previous layer, thereby performing the role of the receptive field. CNNs enable the processing of data that comprise some spacial dependencies (*e.g.* images with a two-dimensional grid-like structure) which has resulted in a plethora of new computer vision applications [9, 35, 117, 317].

CNN architectures possess three characteristics that enable them to outperform traditional ANN architectures when processing image data, namely: *Sparse connectivity*, *parameter sharing*, and *translational equivariance*. In a typical ANN, it is common for layers to be densely connected — each neuron in a layer is directly connected to all the neurons in the preceding and following layer. By contrast, the connections found between neurons in a CNN are referred to as *sparse* — since each neuron is only connected to a limited number of other neurons in the preceding and

⁷An array of weights.

following layer. The intuition is that a sparse network requires less computational power as there are fewer connection weights to be trained, thereby making CNNs more applicable for computer vision tasks [240]. The difference between dense connections (also known as fully connected) and sparse connections is highlighted in Figure 2.11. If dense connectivity is utilised (Figure 2.11(a)), node x_3 has connections to all the nodes $[s_1, s_2, s_3, s_4, s_5]$, whereas if sparse connectivity is utilised (Figure 2.11(b)), node x_3 only has connections to nodes $[s_2, s_3, s_4]$.

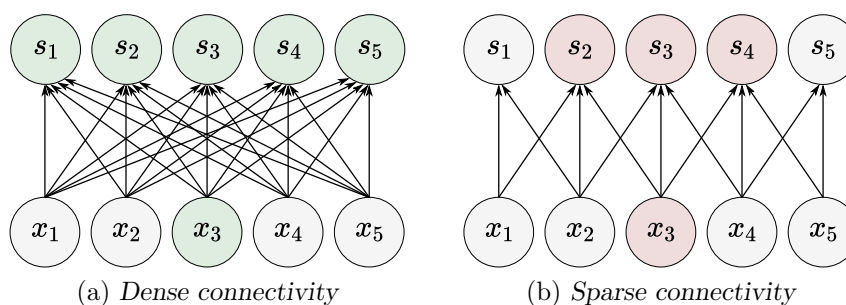


FIGURE 2.11: Graphical comparison of dense connectivity and sparse connectivity [171].

The second characteristic that enables CNNs to outperform conventional ANNs is parameter sharing between CNN output neurons. Within an ANN, each connection weight is used only once when computing the output of the network, whereas convolutional kernel weights are shared across the CNN. This is a result of employing the same sets of convolutional kernel weights across all the window areas of the input data. The sharing of these parameters reduces the computational expenditure of the network. The third characteristic, *translational equivariance*, refers to the notion that if the input changes, the output also changes correspondingly. This is showcased by a CNN when subsampling is employed in order to only propagate prominent data, thereby reducing the data to be processed [186]. With the reduced input data, less computational power is required to train the network. These three characteristics all reduce the computational burden of a CNN in order to perform machine learning tasks, thereby making it the current dominant architecture in the computer vision realm.

2.5.1 Fundamentals of convolutional neural networks

CNN architectures comprise multiple layers, however, unlike traditional ANNs, CNNs comprise three types of layers (or building blocks), namely: Convolutional, pooling, and fully connected layers. The convolutional and pooling layers are employed to perform feature extraction and dimensionality reduction, whereas the fully connected layer maps the extracted features into a final output according to some provided task (*e.g.* classification). The purpose and mathematical operations of the three layer types are discussed in detail in the following sub-subsections.

Convolutional layer

A fundamental component of a CNN architecture is the convolutional layer. The purpose of this layer is to automatically perform feature extraction without the need of human intervention. The convolutional layer comprises both linear and non-linear operations, namely the convolution operation and activation functions, respectively [308]. The linear convolution operation performs the feature extraction of the input (a multi-dimensional array of numbers, commonly referred to as a tensor) with the utilisation of a kernel that is applied across the input.

The linear convolution generally receives a third-order tensor as input, denoted by \mathbf{x} , comprising H rows, W columns, and D channels, with $0 \leq i < H$, $0 \leq j < W$, and $0 \leq d < D$. A tensor of an image usually comprises three channels, *i.e.* the R, G, and B colour channels. If a grey-scaled image is used as input, the tensor would only have a single channel (*i.e.* $D = 1$), and the third-order tensor is reduced to a simple two-dimensional matrix. In order to extract features from the image, different kernels may then be applied across the whole input tensor. Depending on the set of weights for a particular kernel, different features may be extracted from the input tensor, *e.g.* the detection of vertical and horizontal edges of an image [6]. The extraction of these two particular features are graphically illustrated in Figure 2.12, where Figure 2.12(a) represents a grey-scaled image used as the input tensor for this example. Two different kernels are then selected for the extraction of the two different features. Two convolutions are performed on the image, the first to extract vertical edges (visualised by the white stripes in Figure 2.12(b)), and a second to extract horizontal edges (visualised by the white stripes in Figure 2.12(c)).

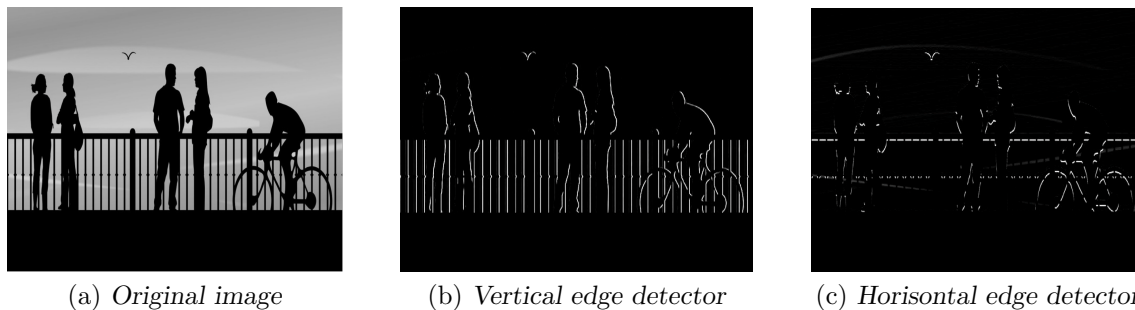


FIGURE 2.12: The effect of different convolution kernels on (a) a greyscale image, where (b) visualises the vertical features extracted, and (c) visualises the horizontal edges extracted [210].

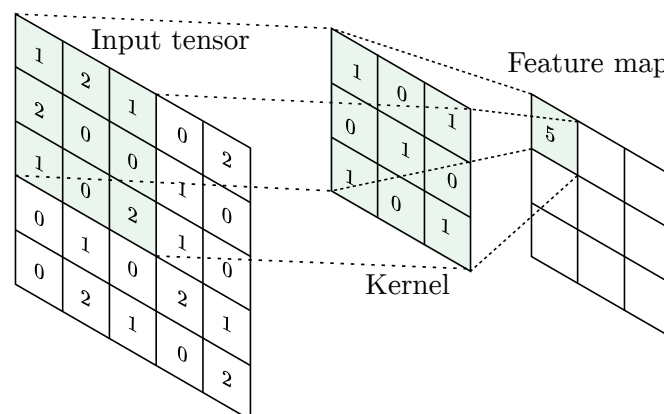
In order to elucidate the mathematical workings of a full convolution, consider the l -th layer of a CNN with a third-order tensor \mathbf{x}^l , where $\mathbf{x}^l \in \mathbb{R}^{H^l \times W^l \times D^l}$, as input to be transformed into an output \mathbf{y} , which is the input to the next layer. Assume $D = 1$, thereby making it a simple matrix of size 5×5 with one single convolutional kernel of size 3×3 , as illustrated in Figure 2.13. If a kernel is placed on the top of the input matrix, the product between the numbers at the same location in the kernel and input matrix may be computed and then summed together to obtain a single number [306]. For example, if a kernel is placed on the top left-hand corner, shown in Figure 2.13(a), the convolution result for the first feature map cell value would be equal to 5. The kernel may then move a preselected number of cells to the right-hand side and compute the next feature map cell value. The number of cell values that the kernel traverses between positions is referred to as the *stride* value, denoted by S . It is common to select a stride of $S = 1$, however, $S > 1$ may be selected in order to downsample the feature map [308]. For this example, $S = 1$ is utilised. The kernel continues to move from left to right and from top to bottom and repeat the convolution until all the cells within the feature map is computed, as shown in Figure 2.13(b), producing a full feature map. In the case where an image has more than one channel, the kernel has the same depth as that of the number of channels. When the kernel is placed on top of the input tensor at the spatial location $(0,0,0)$, the convolution operation is repeated for each channel and the sum of all the HWD^l products is assigned as the convolutional result of the specific spatial location [154].

Convolutional layers generally comprise multiple convolution kernels. Assuming D kernels are used, each with a spatial span of $H \times W$, all the kernels in the l -th layer are denoted as \mathbf{f} , a fourth-order tensor in $\mathbb{R}^{H \times W \times D^l \times D}$ with index variables of $0 \leq i < H, 0 \leq j < W, 0 \leq d^l < D^l$

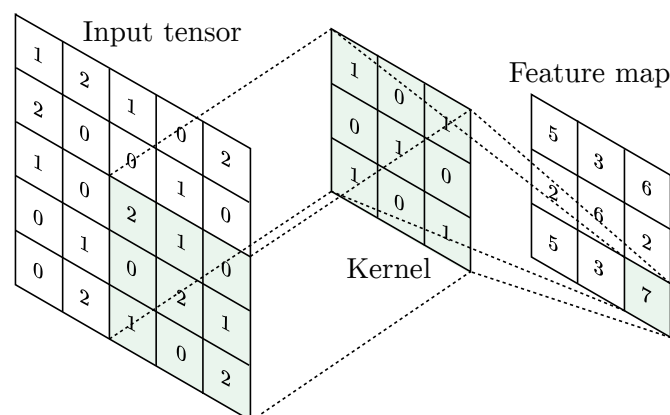
and $0 \leq d < D$. Therefore, in mathematical terms, the convolution procedure can be expressed as

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^H \sum_{j=0}^W \sum_{d^l=0}^{D^l} f_{i,j,d^l,d} \times x_{i^{l+1}+i,j^{l+1}+j,d^l}^l. \quad (2.20)$$

Two drawbacks from this operation include the loss of information found on the border of the input tensor and the reduction in size of the feature map in comparison with the input tensor [6]. This occurs when no augmentation is performed to the input tensor, as in the convolution performed in Figure 2.13. This is known as *valid padding* as the feature map has the same dimensions as the kernel itself. A technique known as *same padding* or *zero-padding*, graphically visualised in Figure 2.14, may be employed to avoid these two drawbacks. The same padding technique results in the feature map to have the same dimensions as the input tensor. This is achieved by augmenting the input tensor by adding a border of zeroes around it, and thereafter applying the convolution procedure as discussed. The implementation of this technique leads to border information having increased influence on the resulting feature map, and also provides control over the output size of the feature map.



(a) Calculation for feature map output cell 1



(b) Calculation for feature map output cell 9

FIGURE 2.13: Graphical illustration of the convolution operation when calculating (a) feature map cell 1, and (b) feature map cell 9 [308].

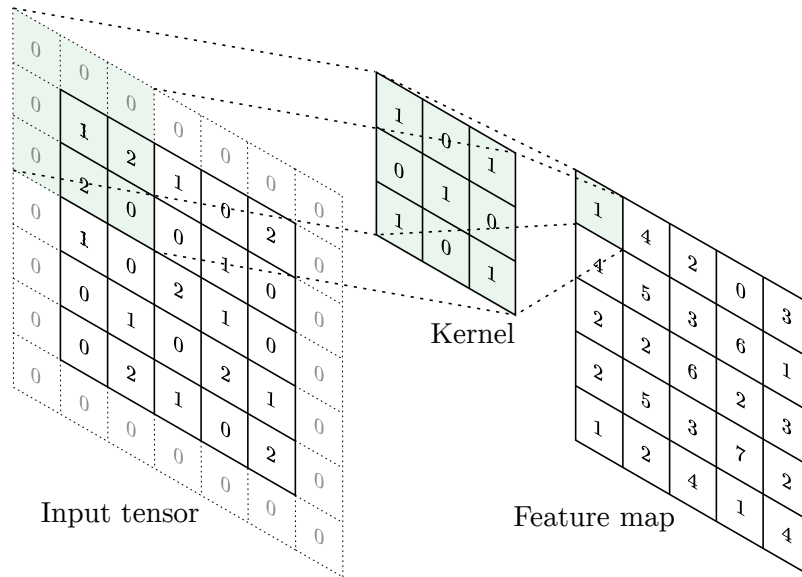


FIGURE 2.14: Same padding applied to a 5×5 input tensor, thereby augmenting it into a 6×6 input tensor [308].

Pooling layer

In order to reduce the spatial size of the convolved feature map, it is common for a convolutional layer to be followed by a pooling layer. The purpose of this layer is to decrease the required computational power for data processing by reducing the dimensions of the output from the convolutional layer while retaining useful information [172]. The pooling layer utilises a principle called *image local correlation* to downsample an image. The pooling operation is performed by mapping subregions, of size $p \times p$, of the feature map into single numbers. The pooling layer may downsample the height H and width W of a feature map, however, the depth D remains the same. By downsampling the feature map dimensions, the number of parameters is reduced, thereby yielding a decrease in network training time.

There are several pooling types available, the most prevalent of which are *max pooling* and *average pooling*. Max pooling is performed by mapping each subregion on the feature map to its maximum value. In mathematical terms, max pooling may be expressed as

$$y_{i^{l+1}, j^{l+1}, d} = \max_{0 \leq i < H, 0 \leq j < W} x_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}^l, \quad (2.21)$$

where $0 \leq i^{l+1} < H^{l+1}$, $0 \leq j^{l+1} < W^{l+1}$, and $0 \leq d < D^{l+1} = D^l$ [306]. Average pooling, on the other hand, is performed by mapping each subregion on the feature map to its average value. More specifically, average pooling may be expressed as

$$y_{i^{l+1}, j^{l+1}, d} = \frac{1}{HW} \sum_{0 \leq i < H, 0 \leq j < W} x_{i^{l+1} \times H + i, j^{l+1} \times W + j, d}^l. \quad (2.22)$$

Similar to the traversal of kernels in the convolutional layer, pooling layers also utilise stride in order to traverse between positions and control the size of the layer output. In practice, it is common to perform pooling with a filter size of 2×2 and a stride of $S = 2$ [308]. A corresponding graphical illustration of the two pooling methods applied to a 4×4 feature map, with $p = 2$ and $S = 2$, is provided in Figure 2.15, where Figure 2.15(a) visualises the max pooling method, and Figure 2.15(b) visualises the average pooling method. Figure 2.16

showcases examples of how the max pooling method was used for downsampling on three different feature maps of the same image. The feature maps were downsized from 26×26 to 13×13 . It is important to notice how the characteristics of the feature maps remained prominent after the pooling procedure, showcasing how pooling retains the important feature map information while reducing the number of parameters.

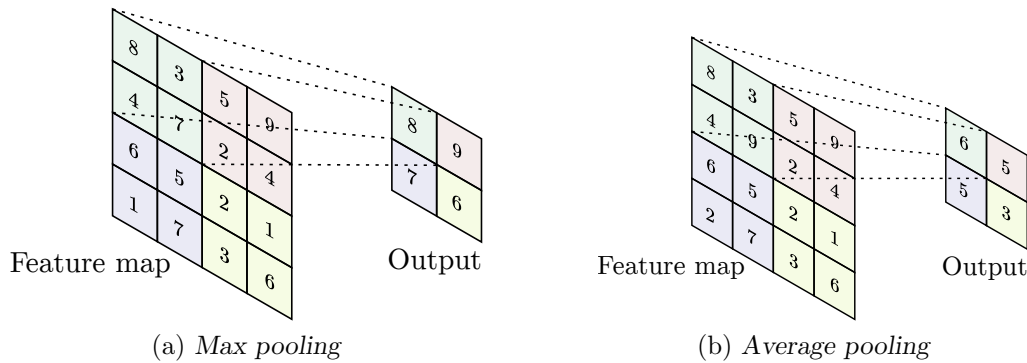


FIGURE 2.15: Two pooling methods applied to a 4×4 feature map, where (a) visualises the max pooling operator and (b) visualises the average pooling operator.

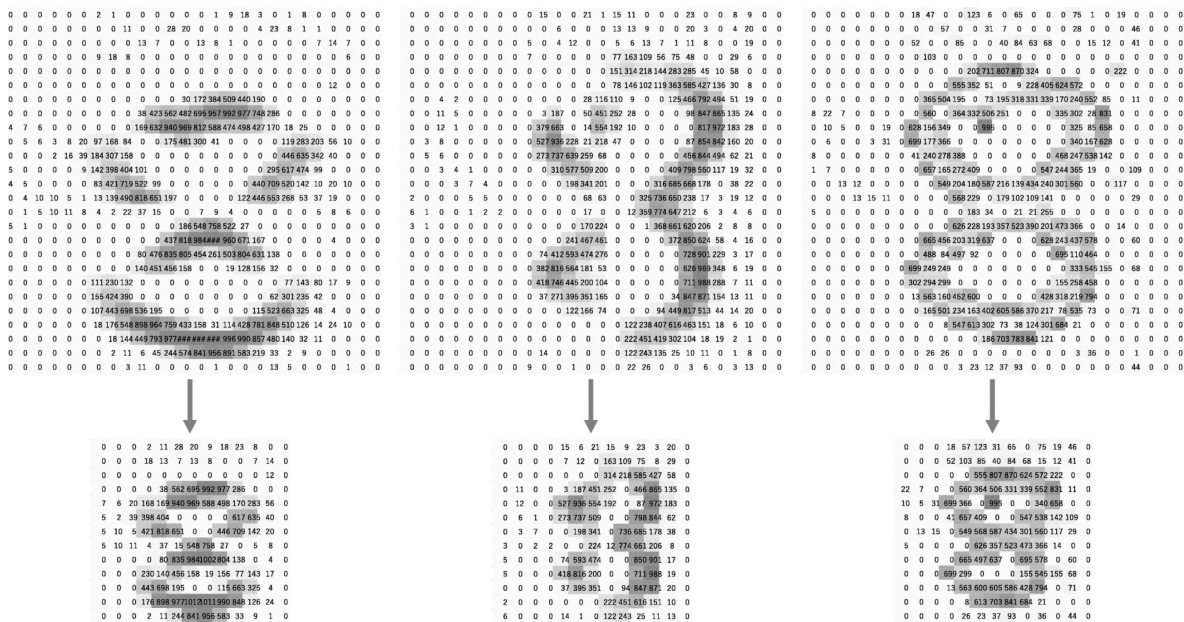


FIGURE 2.16: Max pooling of three different feature maps extracted from the image of the number eight 308.

Fully connected layer

The fully connected layer comprises simply one or more fully connected MLPs, also known as dense layers (*i.e.* each neuron receives input from all neurons of the previous layer). In the case of image classification, the purpose of the final fully connected layers is to utilise the condensed feature mappings and mathematical operations in order to classify the original image. The input to the fully connected layer is a flattened one-dimensional output (as visualised in Figure 2.10) of the *final* convolutional or pooling layer. This flattened array is then fed to

the fully connected layers. Each fully connected layer employs non-linear activation functions, where ReLU is commonly used. The final fully connected layer maps the received input to the network output. In the case of classification, the final layer typically has the same number of output nodes as dependent variables (*e.g.* classes). It is important to note that the final fully connected layer of the architecture might implement a different activation function than the other fully connected layers, as the final activation function is selected according to the task that the network must perform. Commonly used activation functions include softmax, tanh, and the sigmoid functions. Table 2.2 lists several output layer activation functions which may be employed for various prediction tasks.

TABLE 2.2: Commonly applied output layer activation functions for various tasks [308].

Task	Output layer activation function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Sigmoid
Regression to continuous values	Identity

Overfitting tends to be a regular problem when training a CNN. In order to overcome these undesired results, the *dropout* method may be utilised [305]. Dropout is a *regulariser* method employed to prevent overfitting by stochastically setting activations of certain hidden units to zero. This is performed for each training case during the network's training time, resulting in a reduction of co-adaptation of feature detectors, since the neurons with a zero activation cannot influence other retained units.

2.5.2 Prominent convolutional neural network architectures

Various technological and methodological improvements in the early 2000s led to the publication of the AlexNet architecture by Krizhevsky *et al.* [162], which won the ImageNet Large-Scale Visual Recognition Challenge [62] in 2012 convincingly. The performance of AlexNet showcased that CNN models were capable of performing object-detection tasks with considerable efficacy. Consequently, the success of AlexNet inspired the exploration and development of new CNN-based models, resulting in the various CNN architectures found in the literature. A timeline of the evolution of CNN architectures between 1989 and 2019 is graphically visualised in Figure 2.17, showcasing the sudden influx of new architectures from 2012 onwards.

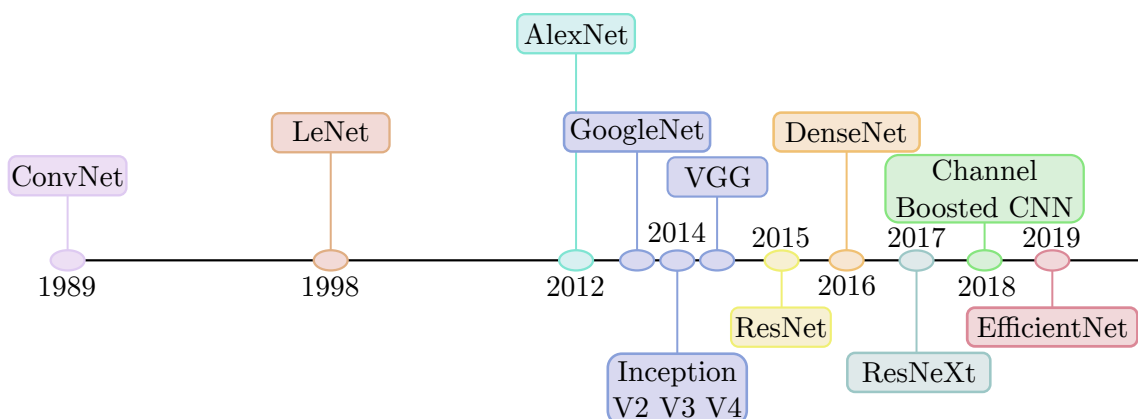


FIGURE 2.17: Evolution of CNN architectures on 1998–2019 timeline [95].

The distinctions between architectures relate to differences in regularisation methods (*i.e.* to implement dropout, data augmentation, and/or early stopping), hyperparameter optimisation (*i.e.* selection of filter size, activation functions, stride, learning rate, weights, biases), and, most notably, structural reformation (*i.e.* number en sequence of layers). It was observed that the primary reason for CNN performance improvements stemmed from the restructuring of processing units (*i.e.* combination of convolutional, pooling, and fully connected layers) and the designing of new types of processing blocks [151]. Accordingly, based on the type of restructuring, CNNs may be categorised into seven classes, namely: Feature-map exploitation, attention-based, spatial exploitation, width, depth, channel boosting, and multi-path CNNs. Most of the recent innovations in CNN architectures were made in respect of spatial and depth exploitation. The remainder of this subsection is dedicated to a more in-depth discussion of the AlexNet [162], VGG [262], and EfficientNet [278] architectures.

AlexNet architecture

Despite inspiring several new and modern CNN architectures, AlexNet remains one of the most efficient and effective architectures when performing image classification [252]. The overall architecture of AlexNet is illustrated in Figure 2.18. The AlexNet architecture has eight weight layers comprising five convolutional layers and three fully connected layers. Max pooling is performed after the first, second, and fifth convolutional layers. The pooling layers perform simple linear operations and are therefore not counted as weighted layers. The first convolutional layer has 96 filters which utilise a large filter size of 11×11 (with an S of four), where the second convolutional layer has 256 filters with a filter size of 5×5 (with an S of one), and the third, fourth and fifth convolutional layers have 384, 384, and 256 filters, respectively, all with a smaller 3×3 filter size and an S value equating to one [314]. The AlexNet architecture has 60 million parameters, which is considered to be a small network when compared with modern CNN architectures. In order to reduce overfitting, the authors also implement data augmentation techniques and dropout in the first two fully connected layers [162]. If dropout is not implemented in the classification layers, AlexNet exhibits substantial overfitting.

The AlexNet architecture is adopted for solving a wide variety of complex tasks, such as: Chest X-ray Covid-19 recognition [55], pathological brain detection [176], and time-series classification, to name but a few. [139].

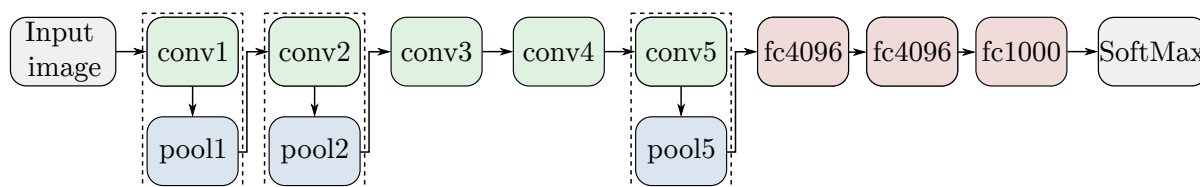


FIGURE 2.18: Overall architecture of AlexNet, where the convolutional, pooling, and fully connected layers are visualised in green, blue, and red blocks, respectively [314].

VGG architecture

In 2014, only two years after the success of AlexNet, Simonyan and Zisserman [262] proposed a simple CNN architecture that is similar, but much deeper (*i.e.* comprises more layers) than AlexNet, called the VGG architecture. According to Khan [151], the reasoning behind the increase of layers (in comparison with AlexNet) was to simulate the relation of depth with respect

to the representational capacity of the network. There are three variants of the VGG architecture, namely VGG-13, VGG-16, and VGG-19. The architecture of VGG-16 is illustrated in Figure 2.19. The VGG-16 architecture comprises 16 weight layers, hence the naming convention, which consist of 13 convolutional layers each employing a convolutional filter size of 3×3 , and three fully connected layers [314]. The S and padding of all convolutional layers in the VGG-16 architecture is fixed to one pixel. Max pooling is implemented in pooling layers with a 2×2 window and an S value equating to two. Simonyan and Zisserman experimentally showcased that by placing small 3×3 filters in a concurrent manner induced the same effect of the larger filter sizes in AlexNet, while reducing the number of parameters [151]. A major downside of the VGG architectures is that they are relatively large when compared with other CNN architectures. VGG-16, for example, comprises 140 million parameters, the majority of which are located in the first fully connected layer. Consequently, the three different variants, *i.e.* VGG-13, VGG-16, and VGG-19, are all employed based on the computational resources available. VGG-16 is, however, by far the most popular variant in the VGG family, as it showcases similar accuracy performance to the VGG-19 architecture on most tasks, whilst comprising a significantly smaller number of trainable parameters.

The VGG architecture outperformed many other architectures in the ImageNet Large-Scale Visual Recognition Challenge in 2014. VGG also outperforms baseline accuracies on various other detection tasks outside the realm of ImageNet. Krishnaswamy and Purushothaman [161] showcased how a pre-trained VGG-16 model was used to detect diseased eggplants using digital images of the samples. Zhong *et al.* [315] utilised two parameter sharing VGG-16 networks in order to build a Siamese network so as to facilitate the accurate recognition of human palm prints.

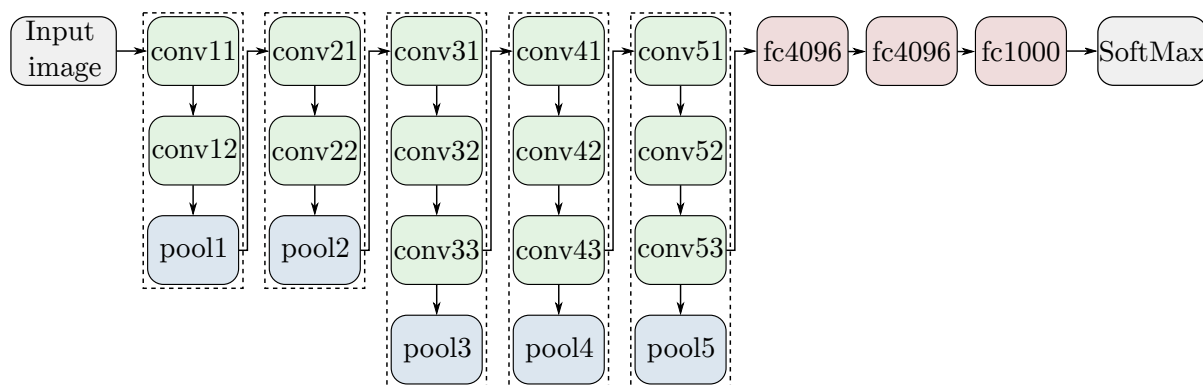


FIGURE 2.19: Overall architecture of VGG-16, where the convolutional, pooling, and fully connected layers are visualised in green, blue, and red blocks, respectively [314].

EfficientNet architecture

In order to restructure a conventional CNN architecture, the scale of either the depth, width, or image input resolution of the architecture is altered. According to He *et al.* [119], increasing the network depth, as in the case of the VGG architecture, is regarded as the most common approach towards scaling an architecture. Deeper networks tend to capture more complex features, resulting in improved generalisation performance with respect to new tasks [278]. Due to the vanishing gradient problem, however, deeper networks can be challenging to train. Attempts toward alleviating the problem by means of *skip connections* [118] and *batch normalisation* [137] have been made, however, the accuracy gain still diminishes if the network is excessively deep. As an example, even though ResNet-1 000 (comprising a depth of 1 000 layers) is a much deeper

network than ResNet-101 (comprising a depth of 101 layers), both networks showcase similar accuracies [278].

When scaling smaller sized networks, it is common to scale the network in terms of its width (*i.e.* number of channels in a layer) [244]. Wider networks tend to be easier to train and capture more *fine-grained* features. The trade-off, however, is that networks that are both wide and shallow tend to not capture higher level features (or representations) [278].

Scaling the resolution of the input images may potentially lead to the capturing of finer-grained patterns. It is common for early CNN networks to be trained on (low resolution) 224×224 images. For improved accuracy, modern CNN networks tend to use higher resolution images. In 2018, GPipe [127] achieved state-of-the-art ImageNet results with 480×480 images, more than double the resolution previously used in the literature. For notably high resolutions, however, the accuracy diminishes, showcasing that scaling up the resolution excessively can result in no additional accuracy gain, but increases the model complexity.

Through empirical observation, Tan and Le [278] found that scaling the dimensions of depth, width, or resolution is not independent. Consider the case where the resolution of the input images are scaled up. In order to capture similar features, the network depth needs to increase as the increased resolution image comprises more pixels. Correspondingly, the network width also has to be scaled up to capture the fine-grained patterns in the higher resolution images. Accordingly, the scale of the three dimensions must be coordinated. In order to facilitate this coordinated scaling, Tan and Le proposed the *compound scaling* method which scales each dimension with a constant ratio, denoted by ϕ . The compound method is graphically illustrated in Figure 2.20, where Figure 2.20(a) is a baseline architecture, Figure 2.20(b) represents scaling by width, Figure 2.20(c) represents scaling by depth, Figure 2.20(d) represents scaling by image resolution, and Figure 2.20(e) represents the compound scaling method.

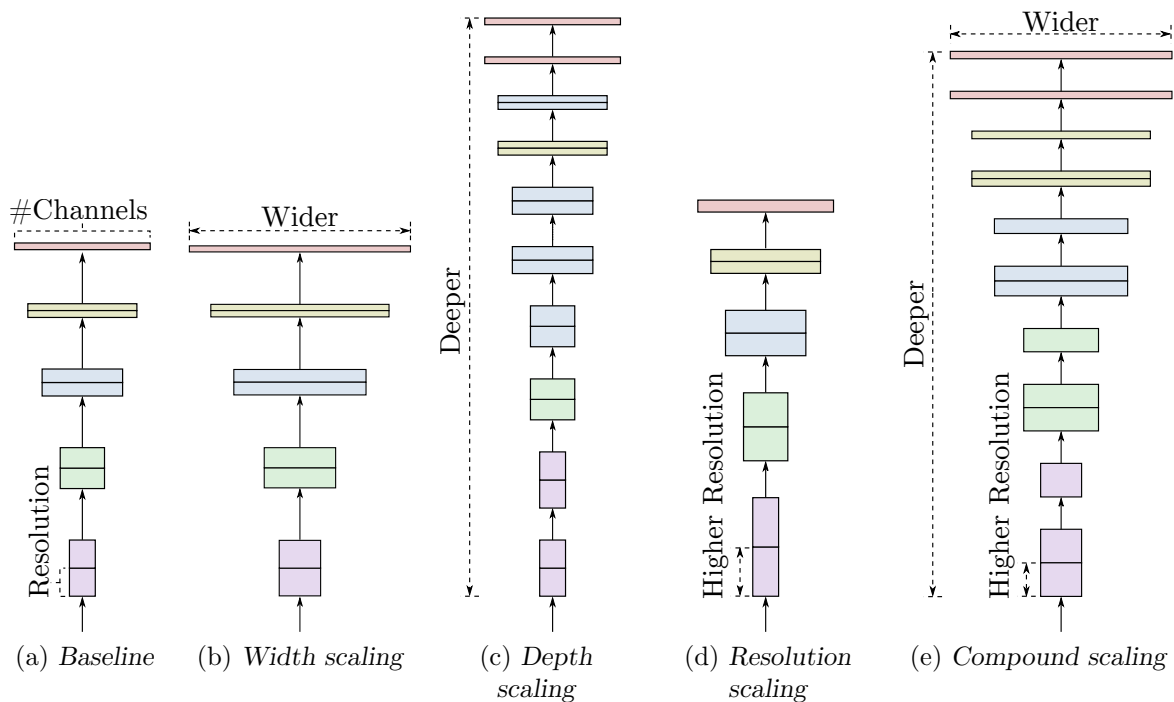


FIGURE 2.20: Model scaling dimensions, where (a) is a baseline network example, (b)–(d) are conventional scaling that only increases one dimension of network width, depth, or resolution, and (e) represents the proposed compound scaling method [278].

The compound scaling method can be expressed mathematically as follows: Depth $d = \alpha^\phi$, width $w = \beta^\phi$, and resolution $r = \gamma^\phi$, such that $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, $\alpha \geq 1$, and $\beta \geq 1, \gamma \geq 1$, where the values of α , β , and γ are determined by means of grid search. Since scaling a model does not influence the layer operations, it is important to have an acceptable baseline architecture. The EfficientNet baseline architecture, EfficientNet-B0, was developed utilising multi-objective *neural architecture search* [73]. The main building block of the EfficientNet-B0 is *mobile inverted bottleneck blocks* (MBconv) [244], a type of *residual block* [118] that employ an inverted structure for efficiency reasons. The overall architecture of EfficientNet-B0 is illustrated in Figure 2.21, where the yellow blocks represent MBconvs. Computationally enabled by the EfficientNet baseline architecture, together with the compound scaling method, Tan and Le demonstrated that the model can be scaled up effectively, so much so that state-of-the-art ImageNet performance is achieved (at the time of publication).

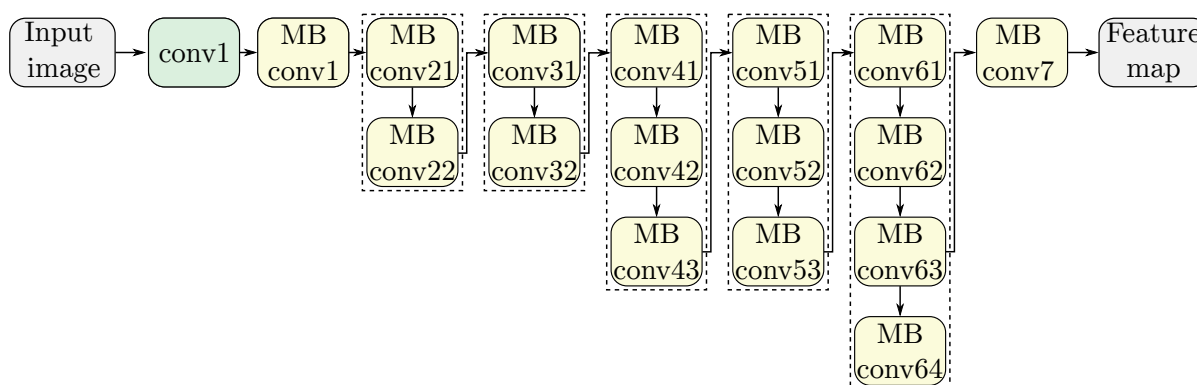


FIGURE 2.21: Overall architecture of EfficientNet-B0, where the MBconv layers are visualised by the yellow blocks [278].

2.6 Transfer learning from pre-trained models

Consider the following hypothetical and analogous scenario: A professional pianist (*i.e.* participant A) and an individual with no musical experience or knowledge (*i.e.* participant B) are tasked with learning how to play the violin. The expectation is that the skills and knowledge that participant A obtained through years of studying and practising how to play the piano would enable a faster learning pace when attempting to play the violin, when compared with the pace of participant B, who is attempting to play an instrument for the first time in their life. The reason for this expectation is that although the act of playing a piano and playing the violin differs notably, there are many conceptual similarities as both are musical instruments requiring a certain set of skills, *e.g.* motor skills, sensory skills, and a sense of rhythm, to name but a few. Participant A is proficient in these skills as a result of his/her experience in playing piano, and may therefore *transfer* these skills over in order to learn to play the piano more effectively.

State-of-the-art deep learning models are known for requiring significantly large training and testing sets comprising well-annotated data [132]. Furthermore, in order to process these large data sets, a significant amount of computational resources and time is required. There are many situations where annotated data are too scarce and computational resources are too limited in order to train new models for computer vision specific tasks. This is where a technique called *transfer learning* [300] may be implemented as a method of accelerating development. Transfer learning is a machine learning technique where parts of, or entire existing models (*i.e.* models that are already trained for a specific tasks), are reused to solve a new task without retraining

on new data. These models are referred to as pre-trained models. The intuition is that previous weights and learnings from a pre-trained model may be utilised in order to avoid starting the entire learning process from scratch. Similar to how the professional piano player can learn to play the violin more effectively, due to their previously attained knowledge and experience with another musical instrument, a pre-trained model may be utilised in order to perform a new task, even though the model might have not been initially trained for that purpose.

A key part of the transfer learning technique is *generalisation*. Models utilised in the transfer learning process must be able to generalise in order to be used in different scenarios. Many pre-trained computer vision models were initially trained on ImageNet, comprising over a million images classified in respect of a thousand categories [114]. The diverse set of images and categories of ImageNet renders it an appropriate data set to train state-of-the-art CNN models capable of good generalisation.

A pre-trained CNN model comprises two parts, namely the convolutional base, which transforms the input image into feature maps, and the classifier, which processes the feature maps as input and output predictions. One of the reasons why transfer learning performs well in the realm of computer vision, is due to the capabilities of the convolutional base to automatically learn hierarchical feature representations [259]. This results in the first few layers of the convolutional base extracting general features, while the last few layers in the convolutional base extract more features of a lower abstraction. Yosinski *et al.* [313] stated that if the first few layers extract general features proficiently, and the last few layers extract specific low-level features proficiently, then there must be a point in the convolutional base where the extracted features transition from general to specific. Accordingly, the features extracted by the layers closer to the input of the architecture are referred to as general features, whereas the features extracted by layers closer to the classifier layers are referred to as specialised features. This is a key concept to take into account when deciding on an appropriate transfer learning strategy.

A transfer learning strategy refers to the combination of pre-trained layers (also referred to as *frozen* layers) and trainable layers (*i.e.* layers with weights that are retrained on the new problem-specific data). It is common for the classifier part of a pre-trained model to be trainable in order to produce accurate problem specific predictions, however, the convolutional base may comprise various combinations of trainable and frozen layers [179]. There are four primary strategies for incorporating transfer learning into a new model. These four strategies are graphically illustrated in Figure 2.22, where the green shading refers to trainable layers and the white shading to the frozen pre-trained layers. Strategy 1, *i.e.* Figure 2.22(a), relates to the case in which the entire model is trainable on the new problem-specific data. Strategy 2, *i.e.* Figure 2.22(b), represents the case in which the majority of the convolutional base is trainable on the new problem specific data, while only the most general feature layers of the pre-trained model are kept frozen. Strategy 3, *i.e.* Figure 2.22(c), refers to the case when only the specialised feature layers are trainable, while most of the convolutional base is kept frozen. Strategy 4, *i.e.* Figure 2.22(d), represents the case where the entire convolutional base is kept frozen.

In order to decide which training strategy to implement, the size and similarity of the new problem-specific data set must be compared with the data on which the pre-trained model was trained. This decision may be guided by a size-similarity matrix, graphically illustrated in Figure 2.23(a), with a corresponding strategy selection matrix visualised in Figure 2.23(b). The selection process is performed as follows:

- Quadrant 1: If the problem-specific data set is considered a large data set, but the type of data differs significantly from the data on which the pre-trained model was trained, Strategy 1 ought to be followed. Since a large well-annotated data set of the specific

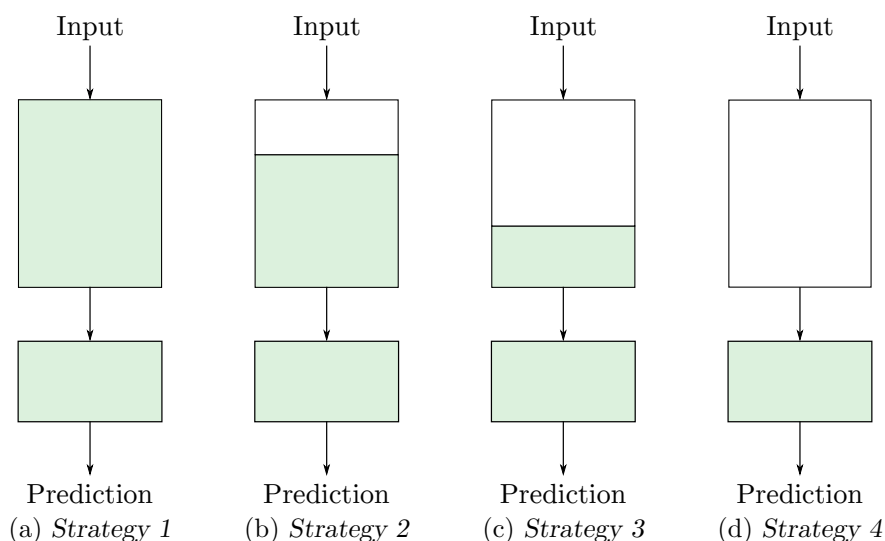


FIGURE 2.22: Strategic implementation of pre-trained models with regards to (a) training the entire model, (b) and (c) training some layers and leaving others frozen, and (d) freezing the convolutional base [179].

problem is available (and if the computational resources are available), the convolutional base of the model may be fully retrained. This yields the most accurate results, as the pre-trained data differ significantly from the problem specific data.

- Quadrant 2: If the problem-specific data set is small and the data differ significantly, Strategy 2 ought to be implemented. This is a challenging situation, as the data might not be sufficient to retrain the entire model, and the two data sets differ significantly from the specialised layers of the convolutional base to improve the accuracy. In this case, it is best to attempt to freeze the most general feature layers, and retrain the remainder of the convolutional base with the data that are available.
- Quadrant 3: If the problem-specific data set is large and the data are also similar to the data on which the pre-trained model was trained, the user has the option to retrain some of the more specialised layers in order to maximise accuracy, which corresponds to Strategy 3. Even though it might not be required to retrain most of the layers, due to the similarity of the two data sets, the user does have the flexibility with the large problem specific data set to attempt to improve the weights of the last few specialised layers in the convolutional base.
- Quadrant 4: If the problem-specific data set is small, yet similar to the data on which the pre-trained model was trained, Strategy 4 ought to be followed. The small size of the problem specific data set would make it challenging to retrain the weights of any layer to an acceptable level when compared with the fully pre-trained model.

After an appropriate strategy for training the convolutional base is selected, the second part of the model, *i.e.* the classifier, must be selected and built. There are a few different approaches in order to build a correct classifier. The most popular approach, especially for image classification, is to use a combination of a few fully connected layers, followed by a final Softmax activated layer, as discussed in §2.5.1. Another approach, proposed by Lin *et al.* [173], is to simply add a global pooling layer in order to reduce the spatial size of the convolved feature maps, followed

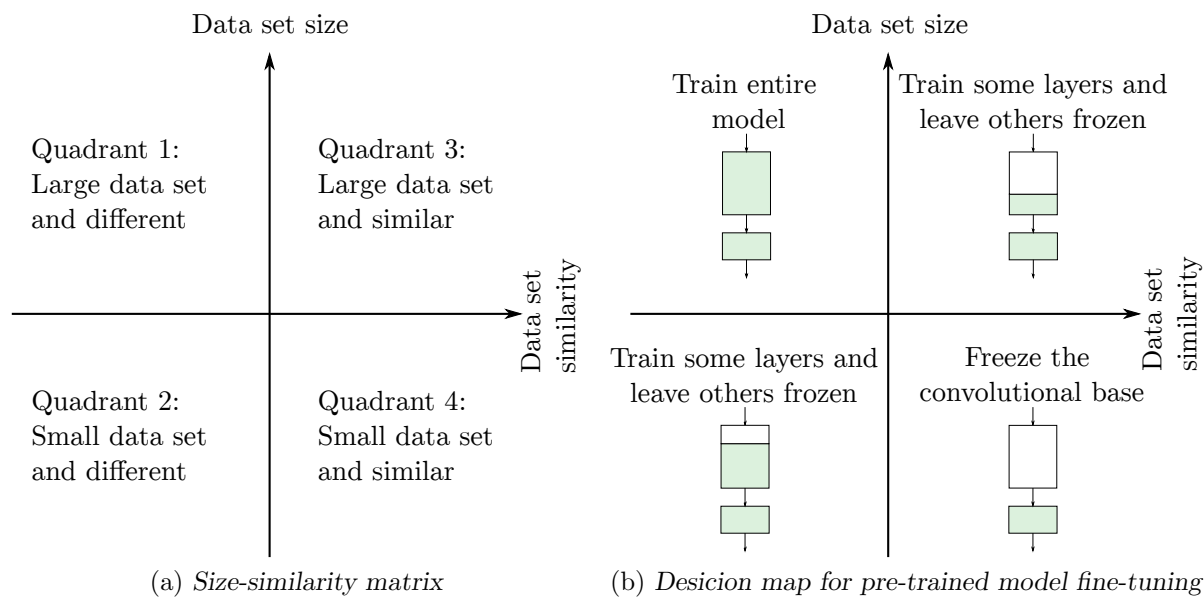


FIGURE 2.23: The size-similarity matrix and the corresponding decision map for pre-trained models [179].

by a final Softmax activated layer. Finally, according to Tang [279], the classifier can be trained with a linear *support vector machine* (SVM) classifier in order to improve accuracy.

Regardless of the approach adopted, the high-level overview of the transfer learning process is as follows: Identify and obtain a pre-trained model that is most appropriate for the specific problem at hand. Thereafter, the size-similarity matrix may be utilised in order to decide on a training strategy for the convolutional base of the pre-trained model, based on the data and resources available. A classifier that receives the feature maps as input and produces the predictions then follows. As a final step, attempts towards improving the model performance may be made with fine-tuning, which may include freezing and/or unfreezing some layers in the convolutional base of the pre-trained model and retraining the model.

Ultimately, utilising a pre-trained model in order to make new problem specific predictions may result in a markedly faster experimentation and prototyping process, while simultaneously requiring less resources and potentially yielding state-of-the-art performance.

2.7 Chapter summary

This chapter contained a review of the most relevant and pertinent literature pertaining to deep learning (and more specifically, FNNs, RNNs, LSTMs, and CNNs). The reader was presented with the necessary background information so as to facilitate an understanding of the remainder of the research reported in this thesis.

In §2.1 the fundamentals of ANNs were explained by addressing the inspiration for ANNs — the biological neural network — activation functions, and network learning and training. This was followed by discussions on the main classes of ANNs, where FNNs were explored in §2.2. The focus of this thesis — *i.e.* RNNs, LSTMs, and CNNs — were elaborated upon in more detail. A discourse on RNNs was conducted in §2.3 during which the reader was introduced to the vanishing gradient problem, a problem that receives considerable attention in the realm of ANNs. This was followed by an exploration of LSTMs in §2.4, where relevant applications of the

network were discussed. CNNs, the most relevant ANN type for this thesis, were discussed in detail in §2.5. Moreover, the most prominent CNN architectures were discussed. As a technique regularly employed in computer vision tasks, transfer learning was finally discussed in §2.6, with an emphasis on the implementation strategy of this technique.

CHAPTER 3

Optical character recognition

Contents

3.1 OCR engines	45
3.1.1 Origins of OCR engines	46
3.1.2 OCR challenges when digitising printed text	47
3.2 Major phases of optical character recognition engines	48
3.2.1 Image acquisition	49
3.2.2 Preprocessing	49
3.2.3 Segmentation	50
3.2.4 Feature extraction	51
3.2.5 Character classification	51
3.2.6 Post-processing	52
3.2.7 Output distribution	52
3.3 Optical character recognition evaluation metrics	52
3.4 Prominent optical character recognition engines	55
3.4.1 Tesseract	55
3.4.2 EasyOCR	58
3.5 Chapter summary	60

The aim in this chapter is to review the pertinent literature pertaining to OCR engines. The reader is introduced to the general structure of a typical OCR engine so as to facilitate an understanding of the various steps that take place when digitising text contained within a document. The chapter opens with a discussion on the general background of OCR engines and the typical challenges faced when digitising text within an image. This is followed by a more in-depth discussion of each of the major phases that constitute a typical OCR engine. Thereafter, a discourse on different OCR evaluation and metrics is provided. Two prominent OCR engines considered in this thesis, *i.e.* Tesseract and EasyOCR, are explored. The chapter concludes with a concise summary of the discussed literature.

3.1 OCR engines

The amount of data in the world is growing exponentially. It has become impractical and inefficient to store these data in physical paper format — hence the need for data to be stored

digitally. Paper-based documents are, however, still integrated into many active systems in the industry. Consequently, a need for automated text recognition arose in order to incorporate paper-based documents into digital systems. The trained human brain possesses the capabilities to recognise text/characters from an image with ease, even from a young age, while machines still lack the intelligence to perform these actions to the same degree of accuracy and/or speed [8]. Consequently, a plethora of research efforts and development have been proffered in an attempt to transform document images accurately into machine encoded data.

3.1.1 Origins of OCR engines

Commonly known as OCR, the process of digitising text is defined as the recognition of text (specifically machine printed or handwritten text) within image data by employing a computerised system with an optical mechanism for data extraction [96, 196, 294]. This operation is performed to enable the text to be searched, stored more efficiently, displayed on-line, edited, and utilised in computerised processes. OCR engines are inspired by the human sense of sight and are modelled to resemble how a human would perform the same task, utilising the eyes as input devices and the brain as a data processor. After the introduction of OCR in the 1950s, the traditional use-case was to *recognise* characters on a provided image by matching (with the use of some known metric) the characters against a limited *reference* set of known patterns in the alphabet. Accordingly, OCR was considered as a branch of the field of pattern recognition [28].

Throughout most of the twentieth century, various OCR engines have been developed for commercial use. These commercially focused OCR engines may primarily be categorised into one of two groups, namely: *Task-specific* OCR engines and *general purpose* OCR engines [272]. A task-specific OCR engine is designed to read only a specific document type. Examples of task-specific documents include process forms, credit card slips, mailed letters, or bank cheques. These OCR engines are usually designed to capture a number of custom-made predefined document regions. In order to elucidate this concept, consider the components of a standard bank cheque. An OCR engine designed to digitise bank cheques may be customised to only scan the amount block, name block, and date block, thereby decreasing the complexity of the system significantly. Task-specific systems are commonly characterised by high throughput rates and low error rates. A general purpose OCR engine is designed to be capable of reading various document types. A single general purpose OCR engine may be employed, for example, to read newspapers, business letters, and technical letters. A general purpose OCR engine is significantly more complex than a task-specific OCR engine. Typically, a general purpose OCR engine captures the selected document image, separates the text and non-text regions of the image, and applies OCR to the text regions. General purpose OCR engines are typically characterised by lower data throughput than a task-specific OCR engine, however, recent advances in the field have led to a significant increase in the data throughput rate of high-end page readers [196].

Factors that might contribute to the weak performance of the classifier include image defects, similar symbols, punctuation, and typography. Typical examples of an error prone misclassification of symbols are when the punctuation symbol “,” is confused with the punctuation symbol “.”, when the numeral “0” is confused with the letter “O”, or when the numeral “1” is confused with the letter “I” or the letter “l”. Although research in developing new classifier methods continues to be active, the industry has shifted its focus primarily to the implementation of tried and tested conventional methods.

3.1.2 OCR challenges when digitising printed text

While general purpose OCR engines showcase improved performance in respect of digitising machine printed text, when compared with handwritten text, character misclassification can still occur when digitising machine printed text if the circumstances are considered to be substandard. In order to achieve improved recognition accuracy, the OCR engine requires high quality and high resolution input images. Moreover, it is preferable for these images to have pronounced differentiable structural properties between the text and the background of the image [196]. If an inadequate classifier method is employed, the OCR engine might lack a proper response, resulting in classification errors. Consequently, it is essential to ensure that the image is captured at the best possible quality. There are several image capturing challenges that can lead to a substandard captured image. Discussions on some of the most prevalent challenges when digitising text, specifically pertaining to the capturing of the image, follow:

- Scene complexity: A device used regularly to capture document images is a cellphone camera. When this informal capturing method is employed, it is common for objects of the environment (*e.g.* table textures, other background pages, furniture) to be included on the outer edges of the image. A large number of objects in the natural environment might have comparative structures to that of printed characters, thereby rendering it challenging for the OCR engine to segregate text from non-text [111]. Likewise, if a dirty and/or faulty scanner is used to scan a paper document, unintended marks might be present on the page, increasing the complexity of the recognition process.
- Inconsistent lighting conditions: Capturing a document image in an environment that is not characterised by even lighting can prove troublesome. This might not be problematic to the human eye, however, inconsistent lighting might result in significant portions of the document image having a different shade when compared with the remainder of the page, increasing the complexity of the task [218]. In the case of a flatbed scanner, this might arise if the paper document has various folds and creases, causing shadows on the affected areas.
- Skewness (rotation): Most open-source OCR engines are trained to receive as input text lines that are parallel to the horizontal baseline of the image. These OCR engines might struggle to produce an acceptable outcome when attempting to classify characters from document images that are captured with a unique orientation, *i.e.* scanned in skew or taken from a hand-held camera [257].
- Blurring and degradation: In order to achieve the best possible recognition, character sharpness is essential. There are various document scanners and digital cameras available for capturing images of text, all of that comprise a unique set of characteristics which influence the sharpness of the captured image. If the characters are blurry, the OCR engine might struggle to identify discriminative features used to recognise characters [273].
- Perspective distortion (tilting): Text captured with document scanners are usually constantly parallel to the plane of the sensor capturing the text, however, this is not always the case when capturing text in a natural environment. It is common for certain characters in the text to be closer to the camera. Accordingly, some characters might seem to be larger in scale when compared with the remainder of the characters, even though all the characters are actually similar in size. Most OCR engines are not perspective intolerant, resulting in lower recognition accuracy [190].

- **Warping:** Text on objects of varying geometrics might result in twisted characters. This capturing defect is observed in document scanners when the text of a thick book is scanned, and the entire page could not be exactly parallel to the sensor surface.

Printed text possesses various structural properties, namely font style, font size, font colour, spacing between words, underlined words, bolded words, language, and alphabet type. Several of these properties might add to the complexity of the digitisation process, especially if different combinations of these properties are found on a single captured image. Discussions on some of the most prevalent challenges when digitising text, specifically pertaining to the text properties, follow:

- **Aspect ratios:** Different use-cases of text might require different aspect ratios. The aspect ratio of text on traffic stop signs differs from the aspect ratio of the text used on document pages. Consequently, location, scale, and length of captured text needs to be considered when recognising characters [111]. This, in turn, introduces increased computational complexity.
- **Fonts:** Various styles have been developed for different use-cases. Some complex font styles, *e.g.* italic fonts, might overlap between characters, increasing the complexity of performing character segmentation. Other font styles comprise thick (*i.e.* bold) line strokes, potentially reducing the structural detail of the characters. Another property to consider is the number of fonts captured on an image. If multiple font styles are captured in a single image, the OCR engine must consider a large number of different character classes, thereby increasing the complexity of the task. According to Manna *et al.* [165], OCR engines trained on a single font, known as *mono-font* OCR engines, perform well when digitising a single font type, but struggle when multiple fonts are introduced.
- **Multilingual environments:** Although many languages based on the Latin alphabet have a considerable number of characters, they remain relatively small when compared with the number of character classes of other alphabets, *e.g.* Arabic, Chinese, Russian, and Korean. Moreover, some alphabets are regarded as even more complex as some characters are connected when typed or written as a word, changing the writing shape of the characters. The differences in alphabetic character types and writing styles between some of these languages are considerable. In order to showcase these stark differences of character shapes and connections, the English word for “cat” is visualised in Figure 3.1, using five different alphabetic systems. Consequently, OCR engines trained on a specific alphabetic system cannot be used to recognise characters of another alphabetic system [111].

3.2 Major phases of optical character recognition engines

Several computerised OCR engines have been developed over the years with some of the oldest dating back to the 1980s [28]. A fully functional OCR engine comprises many components, grouped into three stages, namely: The image acquisition stage, the prediction stage, and the output distribution stage. These three sequential stages, specifically employed for document images, are visualised in Figure 3.2. The document digitisation stage is employed in order to capture the physical document into a document image. Thereafter, the captured document image undergoes several processing steps, facilitating the character classifications. Finally, the classified characters are exported as the primary output of the OCR engine in the output distribution stage.

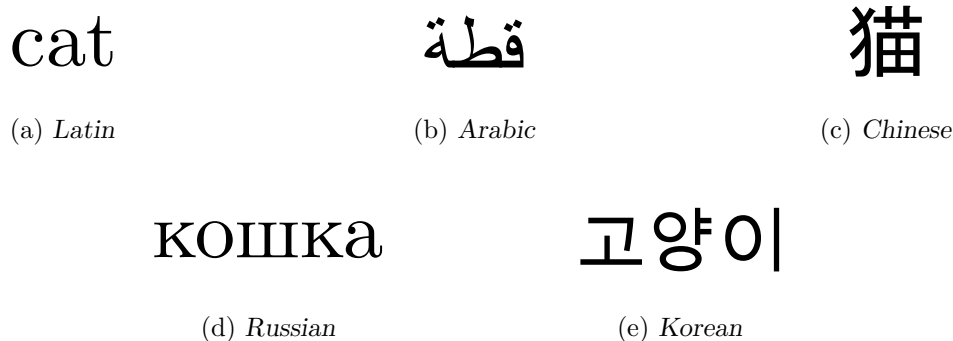


FIGURE 3.1: Visualisation of the English word “cat” where (a) employs the Latin alphabet, (b) the Arabic alphabet, (c) the Chinese alphabet, (d) the Russian alphabet, and (e) the Korean alphabet.

3.2.1 Image acquisition

When digitising text in a document, the first step is to capture the document into an appropriate structure that can be viably processed by a computer. This is commonly achieved by optically scanning the document with a document flatbed scanner. In order to utilise the scanned image data in the second stage of the OCR engine, modern optical recognition methods require that the scan be performed with a scanner that can output an image spatial resolution of at least 300 *dots per inch* (DPI)¹ and can greatly benefit if *grey-scale text imagery* (*i.e.* a range of different monochromatic shades from black to white) is available [125]. It has been reported that utilising low resolution and *bi-tonal thresholding* (*i.e.* only pure black and white) may lead to the scan breaking thin lines and/or filling gaps, which distorts character features required for the character recognition stage [28]. Fortunately, recent advances made in the field of optical scanner technology have resulted in high resolution scanners to be widely available for commercial use. After successfully scanning the text document, the attained image data are then used as input for the character recognition stage.

3.2.2 Preprocessing

The performance of an OCR engine is directly dependent on the quality of the input image data [188]. In the case of OCR tasks, a good quality input document image is defined as an image in which characters are easily distinguishable from the background upon which it is printed, enabling the OCR engine to recognise the characters easily. Some of the characteristics associated with high quality document images are sharp character borders, high contrasts, suitably-aligned characters, and as little pixel noise as possible [65]. Some of the challenges discussed in §3.1.2 (*e.g.* uneven lighting, skewness, blurring, tilting, and warping) may increase the complexity of the captured image, impeding the performance of the OCR engine. Accordingly, enhancing the quality of the image by employing image enhancement techniques is an essential step of any OCR engine. The workings of specific *document* image enhancement techniques are elaborated upon in more detail later in this thesis.

¹DPI is the physical dot density of an image used to measure the resolution of scanned images.

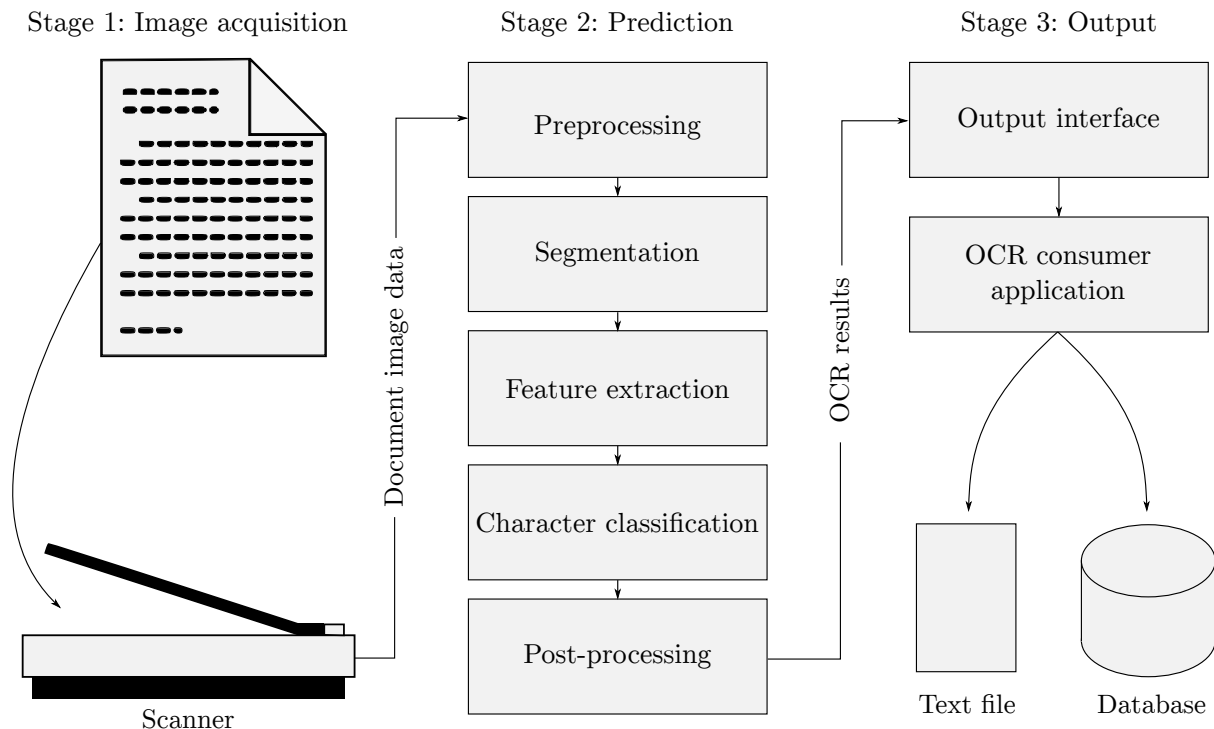


FIGURE 3.2: The three sequential stages of a typical OCR engine.

3.2.3 Segmentation

After the document image is preprocessed, segmentation of the text ensues. Segmentation is the process of classifying the document image into homogeneous zones, *i.e.* areas of specific information types, whereafter the text in the document image can be isolated from the background of the document image [111]. Document segmentation ensures that when the document is converted into an electronic format, the logical structure and format of the original document is preserved in order for the output to be fully utilised [28]. Generally, this is achieved by first sequentially segmenting the text lines, followed by the words within those text line segments, and thereafter, the characters within those word segments. For most OCR engines, the success of this step is a major contributor to the performance of the overall OCR output.

There are three categories of algorithms that may be followed for segmenting the image, namely: Top-down methods, base-up methods, and hybrid methods. The most popular of these methods is the top-down methods which starts with segmenting the document into large regions, recursively, and then segmenting each of those regions into smaller subregions. This is repeated until a stopping criterion is met. Base-up methods, on the other hand, start by locating *interest pixels* and then grouping these pixels into connected components that form part of text characters on the document image. These characters may then be combined in order to form words, which combine to form text lines. The hybrid methods utilise a combination of the top-down and base-up methods. The output obtained after employing a segmentation method, is a variety of isolated characters. These characters are then to be standardised into a particular size for processing [189].

3.2.4 Feature extraction

Feature extraction is then performed, where essential characteristics of the symbols in the text are captured and profiled as feature vectors [136]. These feature vectors are then to be used as input for the classifier to predict the characters. The selection of which and how many features to extract remains an important research question. Geometrical features (*e.g.* loops and strokes) and statistical features (*e.g.* moments) are examples of possible features to extract [8]. There are several techniques in the literature for extracting features from the segmented characters. One such technique is called *zoning*, as graphically illustrated in Figure 3.3(a). A character can typically be separated into several zones of predefined size, such as 2×2 , 4×4 , and so forth. The pixel density of the image may then be broken down into each zone. The number of pixels associated with each zone represents the numeric values used for this feature. *Distance profile features* is a category of features that measures the distance from the merge box of the image to the edge of the character by number of pixels [289]. Examples of horizontal and vertical profile projections of the character “A” are graphically illustrated in Figure 3.3(b) and Figure 3.3(c), respectively. Another technique is to extract *projection histogram features*. Projection histograms calculate the number of pixels associated with the character in bins along a specific direction. These are features that are projected in horizontal, vertical, or diagonal directions [87].

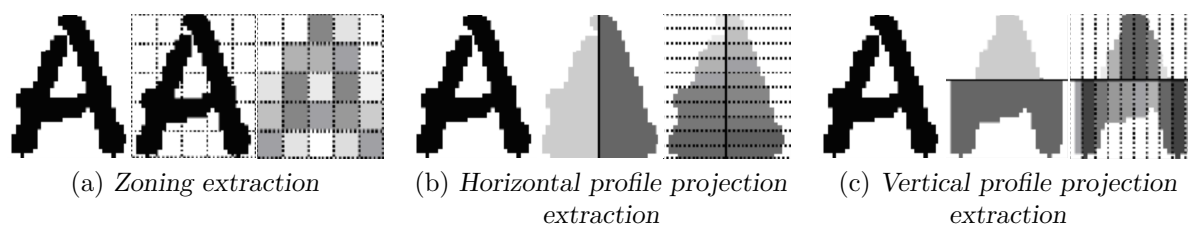


FIGURE 3.3: Feature extraction techniques visualised for the character “A”, where (a) employs zoning, (b) employs horizontal projection, and (c) employs vertical projection [289].

3.2.5 Character classification

The character profiles of the text symbols are used as input for a classifier that attempts to recognise the text. OCR engines broadly use methodologies that utilise pattern recognition, where each segmented character is assigned a probability of the character belonging to a certain class. The feature vectors provided as input must have contrasting characteristics in order to be successfully recognised [189]. When developing an OCR engine there are various classifier types to choose from, each with their own advantages and use-cases. The selection of the classifier depends on several factors, the most important being the nature of the data to be trained on [111]. The least complex classifier method involves utilising template matching. Stored vectors are simply matched to the shape of the segmented characters. Gathering the shape, curvature, and pixel densities, a level of correspondence between the stored vectors and the segmented characters is determined. This method is notably sensitive to noise and subtle disfigurements. The classifier that is utilised the most in OCR engines is the probabilistic neural network classifier. The process of humans classifying characters is seen as heuristic rationale — people can improve upon their recognition skills through experience. Neural network are deemed an appropriate classifier for this task [67]. This classifier employs fully connected FNNs in order to perform the predictions. A softmax activation function must be used in the final fully connected FFN as there are multiple character classes.

3.2.6 Post-processing

Post-processing is performed with the aim to reduce possible errors made by the classifier. While preprocessing aims to “clean” document images, post-processing, on the other hand, takes aim solely at the text output of the classification step. Exhibiting semantic, context, or linguistical information (in respect of the document image) may greatly contribute to the success of post-processing [111]. The least complex example of post-processing corresponds to the case in which linguistic information is known, rendering it feasible to utilise a dictionary for comparing model output words with the words in a dictionary. If a word produced by the engine is not a real word in the specific language, yet markedly similar to a word in the dictionary, then it may be assumed that it was an OCR error. The word may then be processed into the dictionary word if certain conditions are met. This is the same principle adopted in the context of spell checking software.

3.2.7 Output distribution

The output distribution stage receives the predicted and post-processed OCR text of the character recognition stage and presents it for further analyses. Although this stage is simple to execute, it represents the primary communication method of the OCR results and ought to be considered as a vital part of the OCR engine. It is common for these results to be exported directly into *comma separated value* (CSV) spreadsheets, word processors, and/or databases. Another method is for the OCR results to be used directly as input to a downstream automated computer process.

3.3 Optical character recognition evaluation metrics

Research on OCR engines has been active for several decades, as alluded to earlier. There are a wide variety of standardised evaluation metrics available, developed through research outputs and open-source implementations in the industry. These evaluation metrics, however, only provide a partial perspective of the real-world performance of the OCR engines. For the remainder of this thesis, the actual text on a document image (*i.e.* text that can be annotated if required) is referred to as the *ground truth*, whereas the text produced by the OCR engine is referred to as the *OCR text*. In this section, the OCR accuracy analysis workflow is discussed, where string comparison considerations, commonly used evaluation metrics and tools, and the challenges of these approaches are explored.

The OCR accuracy analysis workflow comprises the steps performed in order to compute and analyse the performance of the selected OCR engine [280]. This workflow is graphically illustrated in Figure 3.4. The process starts by receiving two flows of information, *i.e.* the ground truth labels in a structured format, and the captured document images. For the first information stream, the ground truth is stripped from any tags and/or styles embedded within the labelled text. This may include punctuation marks, number words, white spacing conventions, and upper-case words. Thereafter, the ground truth text is processed into an appropriate array of text in order to be compared with the supplementary flow of information. For the second information stream, the captured document images undergo transformation from pixel data into machine-encoded data, *i.e.* the OCR text. The output is then processed into an array of text, in a similar fashion as the ground truth, in order to be fairly compared. Thereafter, the two arrays are compared, utilising some preselected distance algorithm. The results of this algorithm are then used to compute an evaluation metric, representing the performance of the OCR engine.

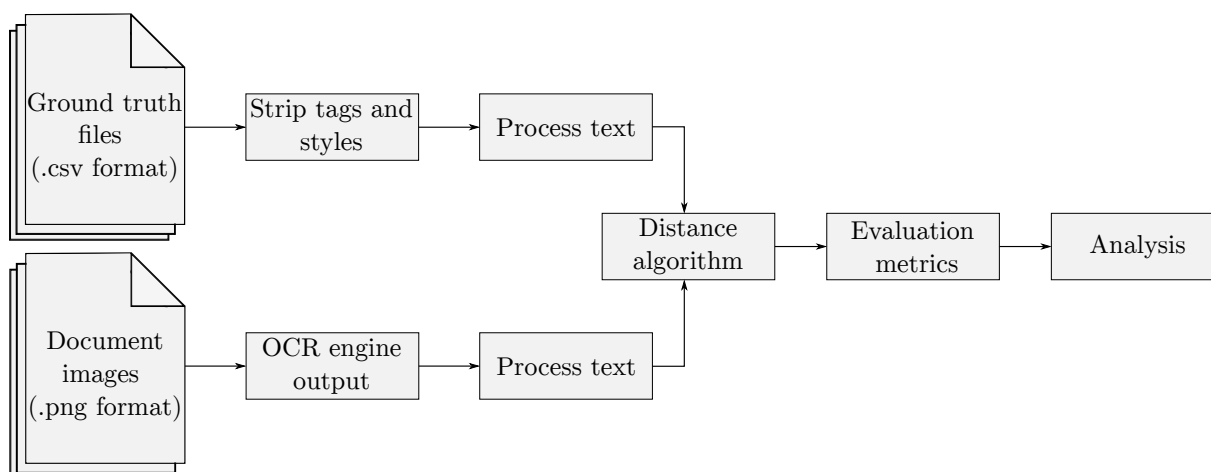


FIGURE 3.4: A typical OCR accuracy analysis workflow [280].

In order to develop an evaluation metric for OCR engines, it is important to understand how strings are compared with one another. The most widely used metric for measuring the difference between two strings is called the Levenshtein distance [170]. Conceptually, the Levenshtein distance algorithm is defined as the minimum number of single-character modifications required to change a string of characters into another string of characters. There are three different change operators that may be applied, namely: Insertions, deletions, and substitutions. An insertion is the act of appending a new character into the string, a deletion is the act of removing a character from the string, and substitution is the act of replacing an existing character with a new character.

Computing an evaluation metric may become complicated when certain scenarios arise [94]. In order to elucidate a scenario called *case folding*, consider three strings extracted from full sentences, *i.e.* string A comprising the characters “White House”, string B comprising the characters “white house”, and string C comprising the characters “White house”. It is clear for a human that string A refers to the official home of the President of the United States, while both string B and C refers to a simple house that is white, but that the upper case “W” in string C is most probably the beginning of a sentence. Depending on the domain of the ground truth, the developer of the evaluation metric ought to make an informed decision whether the case of a letter must be discounted or not.

Another scenario to consider is *white spacing*. Blank or white spaces play an important role in sentences, as it is the primary operator to separate words from one another (*e.g.* the string “mangoes” differs significantly in meaning when compared with the string “man goes”). Consequently, the manner in which white spaces are treated must be considered when computing an evaluation metric. There is, however, special cases to consider in scenarios where multiple white spaces are detected. For example, there is a character difference between the string “deep learning” with a single space and the string “deep learning” with two spaces, yet the meaning of the strings is understood to be the same by a human. There are various such recognition errors where the developer of the evaluation metric must decide whether to consider these scenarios as erroneous in comparison with the ground truth, or not.

The origins of the most common evaluation metrics used in OCR related research dates back to Stephan V. Rice [224]. Described by Rice in his doctoral dissertation in 1996, ascertaining the quality of text recognised by an OCR engine is regarded as the measurement of the manipulation of character strings, which are transformed by an edit distance algorithm. When computing the

edit distance, Rice recommends Ukkonen's algorithm [287], a close variant of the Levenshtein distance algorithm. Moreover, Rice notes the importance distinguishing *character-level accuracy* from *word level accuracy*, as in some cases it is more important to measure the accuracy of absolute words when compared with the general character accuracy of the text. In 1996, the evaluation methods recommended by Rice were implemented in the *Information Science Research Institute Evaluation Tools* [93].

Another approach to the evaluation of OCR performance is not to use character accuracy, but rather to use the *character error rate* (CER), *i.e.* the inverted accuracy [94]. CER is defined as $CER = (i + s + d)/n$, where i denotes insertions, s denotes substitutions, d denotes deletions, and n denotes the total number of characters in the ground truth. Similarly, the *word error rate* (WER) may be defined as $WER = (i_w + s_w + d_w)/n_w$, where i_w denotes the number of words inserted, s_w denotes the number of words substituted, d_w denotes the number of words deleted, and n_w denotes the total number of words in the ground truth.

Several systematic studies of OCR quality, in the context of mass recognition, have been performed throughout the past two decades. In 2009, Tanner *et al.* [280] examined the OCR text quality of the digitised British Library newspaper archive. Although CER was used for most OCR evaluation metrics at the time, Tanner *et al.* argued that the evaluation metrics ought to be orientated to the intellectual aims desired by the user of the OCR text, and that CER does not achieve that aim. In order to elucidate this predicament, consider the following example where a given paper document image comprises 1 000 words with a total of 5 000 characters. If the OCR engine yields a 10% CER, a total of 500 misclassified characters is to be expected. These 500 characters are spread throughout the 1 000 words. If we assume each word comprises five characters, the following two possible extremes are possible: The 500 incorrect characters might all be in words where all the characters are misclassified, resulting in a maximum WER of 10% (*i.e.* 100 words are misclassified), or on the other extreme, if a single incorrect character is found within 500 different words, it would yield a WER of 50% (*i.e.* 500 words are misclassified). As a result of this example, Tanner *et al.* show that CER does not represent the utility of the OCR text to the end-user, and is therefore not an appropriate evaluation metric for real-world applications. Accordingly, they proposed the so-called *significant-word-accuracy-rate*. This evaluation metric only considers the proportion of *significant* words that are incorrectly recognised. Tanner *et al.* defined a word as significant only if the word is relevant to the capturing of the document contents. In order to select these significant words, it was recommended to keep the following considerations in mind:

- Search accuracy,
- volume of search results returned,
- ability to structure search results,
- extent of correction required to achieve desired performance, and
- accuracy of result rankings.

The *Pattern Recognition and Image Analysis* (PRImA) research lab has made major contributions to the evaluation of OCR engines, developing several OCR labelling and evaluation standards. The *Page Analysis and Ground-Truth Elements* (PAGE) format [216], released in 2010, is an XLM-based² standard for ground truth text. It enables the user to capture granular

²Extensible Markup Language is a markup language and file format for storing, transmitting, and reconstructing arbitrary data.

image features, structural layout, and reading order information. In 2020, PRImA developed *Flexible Character Accuracy* (FCA) [53] which facilitates the evaluation of the reading order of the recognised word. This is a particularly important evaluation metric to consider as the meaning of most sentences and words may differ significantly depending on their position in a sequence of words.

When constructing an evaluation metric, it is crucial to take the amount of available ground truth data into consideration. This may refer to the number of document images that are considered fully labelled (*i.e.* the entirety of the text is annotated), or the number of pages of the document images that are partially labelled (*i.e.* some of the text is annotated), or the number of document images that are not labelled. Another strategy to evaluate the OCR engine, if sufficient ground truth data are available, is to apply random sampling of the ground truth on the document pages [280]. The primary advantage of this method is that the adherence of the evaluation metric is computed on randomised samples, rendering the metric more reliable when compared with manual selection. A significant disadvantage of this method, however, is the notion that not every aspect of a document is deemed equally important [200]. When extracting text from a newspaper, for example, it is considered crucial to correctly recognise the heading of the article, while correctly recognising every word in the body of the article as being less important. Another example to consider is the relative importance of recognising the address block of a letter, in comparison with the closing lines of the content itself.

3.4 Prominent optical character recognition engines

With the advent of GPU-computation and the rise of deep learning methodologies in the past two decades, several new open-source OCR engines have been developed. There are a multitude of approaches ranging from notably simple ones, which can only recognise single language pre-processed characters, to significantly advanced ones, which employ advanced machine learning techniques and recognise both handwritten and printed text in multiple languages. The timeline from 1985–2021 of some of the most well-known OCR engines are graphically illustrated in Figure 3.5. The remainder of this section is dedicated to in-depth discussions of Tesseract [92], the most prominent OCR engine to date, and EasyOCR [141], one of the latest OCR engines which employs a variety of modern deep learning techniques.

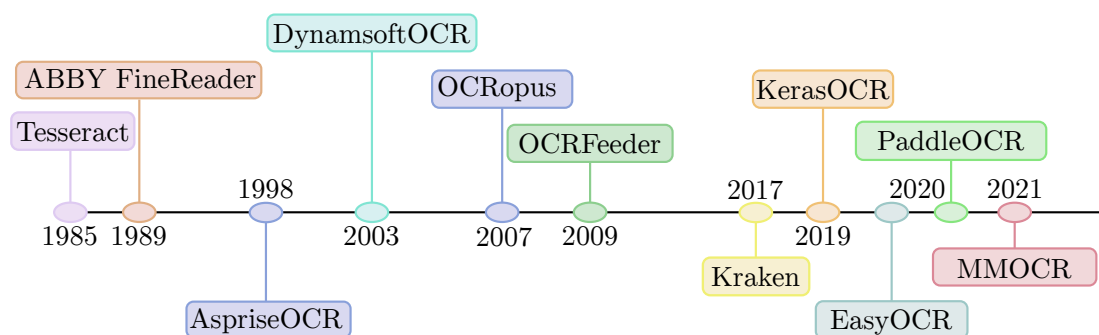


FIGURE 3.5: A chronological depiction of prominent OCR engine releases.

3.4.1 Tesseract

The Tesseract OCR engine was developed by the computer hardware company HP in 1985 [268]. The engine was first introduced to the public at *The 4th University of Las Vegas Annual Test*

for *OCR Accuracy* [223] where the engine showcased excellent performance and was on par with well-known commercial engines of the time. In 2005, HP released the Tesseract OCR engine as open-source, whereafter, in late 2006, the development of the engine was taken over by Google. Tesseract OCR had five major stable release versions, with the latest version 5.1.0 being released in November 2021 [92]. Tesseract version 5.1.0 has a new LSTM-based engine. The engine requires a significant amount of training data and is a lot more computationally expensive when compared with its earlier versions [52]. Consequently, the Tesseract engine that is discussed and utilised in this thesis is the Tesseract version 3.0.0.

The processing performed by Tesseract version 3.0.0 is in the form of a traditional pipeline, however, some of the stages were seen as unusual in comparison with other commercial approaches of the time. The first major step of the Tesseract architecture is to perform *page layout analysis* in order to segment the image into text and non-text regions. The process removes vertical lines and images utilising morphological processing from Leptonica [26], whereafter candidate text blocks are created around the remaining text regions [4]. Thereafter, the lines, words, and characters must be segmented from each other. This is achieved by implementing a *connected component analysis* in order to identify the outlines of components. These outlines, known as *blobs*, comprise a nesting of outlines, with a number of child outlines, and grandchild outlines within.

After attaining the component blobs, line and word finding take place. This is done in four sequential steps, namely: Line finding, baseline fitting, fixed-pitch detection and chopping, and proportional word finding. A key part of line finding is to filter out blobs and line construction. Blob filtering is performed by removing components that are smaller than some fraction of the median height of text approximated by the engine. A simple line finding algorithm, published by Smith [267], was designed in order to attempt to identify skew lines. Blobs overlapping by at least half horizontally are grouped as text lines. Thereafter, baseline fitting takes place where the text lines are fitted to the horizontal baseline utilising the *quadratic spline interpolation*³ method [194], which also enables Tesseract to handle images with curved baselines. Next, the engine searches for text lines having a fixed-pitch, whereafter it separates the blobs into characters by *chopping* the word up into segments. The separation process of the blobs is graphically illustrated in Figure 3.6. This can, however, only be applied to fixed-pitch text lines.

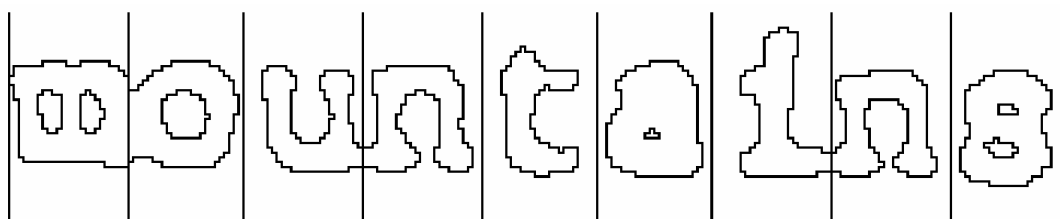


FIGURE 3.6: A graphical illustration of the fixed-pitch word “mountains” separated into characters [268].

Proportional word finding, the final step in the word finding procedure, is implemented in order to find words on a non-fixed-pitch text line. Typical problems faced when searching for words are shown in Figure 3.7. Consider the string “11.9”% shown in this figure. The space between the “11” and the “.” is similar to the general white space, and is certainly larger than the space between the bounding boxes of the two strings “erated” and “junk”. There is almost no space between the bounding boxes of “of” and “financial”. These mentioned examples are common problems faced by any OCR engine. Tesseract tackles these obstacles by measuring the gaps in the limited vertical range between the baseline and the mean line. If the engine struggles to

³A way of finding a curve that connects data points with a degree of three or less.

**of 9.5% annually while the Fed-
erated junk fund returned 11.9%
*fear of financial collapse,***

FIGURE 3.7: A graphical illustration of various scenarios faced by OCR engines [268].

segment certain parts of the text line, the final decision on the word segmentation will be made after the word recognition stage.

After the initial round of segmentation, several words have been segmented into characters coming from the fixed-pitch text lines, and other words from the non-fixed-pitch text lines, which still require further segmentation. Tesseract attempts to separate the non-fixed-pitch into characters by *chopping* the blobs up into different segments. In order to achieve satisfactory results, multiple candidate chopping points are identified on the blob. These candidate chopping points are found on concave vertices of a polygonal approximation [269] of the outline. A set of candidate chops are graphically illustrated in Figure 3.8 for the word segment “arm”. A candidate chop formed with two chop points can be seen between the “r” and the “m”.

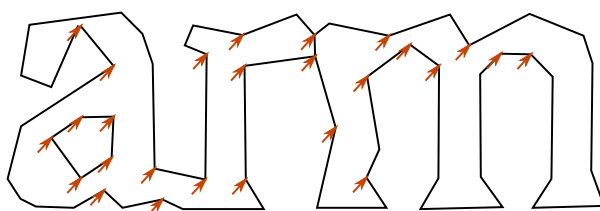


FIGURE 3.8: A graphical illustration of the word segment “arm” and all its candidate chopping point [268].

The next major step in the Tesseract engine is feature extraction. Topological features developed from the work of Shillman *et al.* [25] were utilised in earlier versions of the Tesseract engine. Bokser [27], however, argues that these features are not robust when it comes to real-world problems found in captured images. The breakthrough solution is to use segments of a polygonal approximation as features (shown in Figure 3.9(a)), and then to use features extracted from the outlines of the unknown character (shown in Figure 3.9(b)). The overlap of the physical features are shown in Figure 3.9(c), where the solid black lines are that of the polygonal approximation that are used as prototypes, and the broken red lines are that of the extracted outlines of the unknown character. A problem to take note of, however, is the computationally expensive nature of computing the distances between the different feature sets. Finally, classification may take place by computing the similarity of the unknown character to that of the different classes of polygonal approximations.

The Tesseract engine was originally developed in C++ and has wrappers for Python, Java and Ruby [265]. The engine can recognise text from more than a hundred languages. Moreover, the Tesseract engine provides notable flexibility with respect to the page segmentation procedure, enabling a user to segment the text regions into tables, paragraphs, text lines, or simply a single text block [92]. An engine mode may also be selected, ranging from either utilising the legacy engine (version 3.0.0), utilising the new LSTM engine, or combining the LSTM and legacy engines. The Tesseract engine is optimised for implementation on a CPU, and is known for its good performance when digitising scanned documents [177]. The most noteworthy drawbacks of the Tesseract engine include its inability to utilise the computational power of a GPU and

batch processing, and that the specific page layout analysis technique utilised by the Tesseract engine is seen as inferior to its deep learning counterparts.

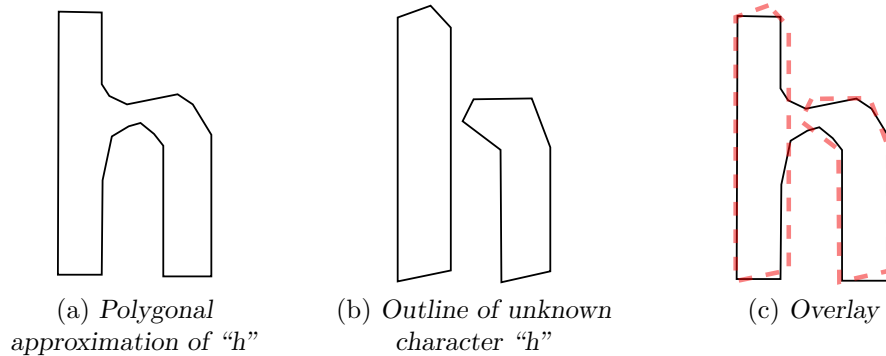


FIGURE 3.9: Classification of the character "h" graphically visualised, where (a) is the polygonal approximation of "h", (b) is the outline of unknown character "h", and (c) is the overlay [289].

3.4.2 EasyOCR

The EasyOCR engine is an open-source project created and maintained by Jaided AI [141], a company that specialises in OCR services. The engine was developed in Python using the PyTorch library [212], and was released in 2020. As the name suggests, the EasyOCR engine is simple and lightweight to implement when compared with its peers.

A primary function built into the EasyOCR framework is for users to be able to insert any state-of-the-art models into the EasyOCR engine. This enables the user to easily utilise the latest models by simply implementing them into the current EasyOCR pipeline [141]. The EasyOCR framework is visualised in Figure 3.10, where the two grey blocks represent the input image and the output OCR text, the green blocks represent the optional processing of the data, and the blue blocks the changeable modules.

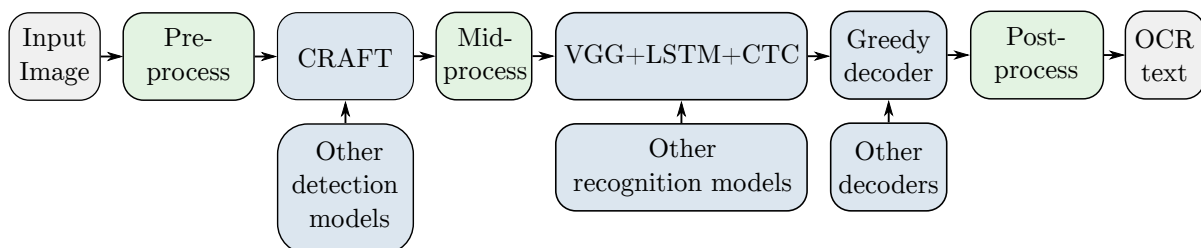


FIGURE 3.10: A graphical illustration of the EasyOCR framework [141].

The base framework initialises upon receiving the input images. Optional preprocessing may then be performed. Thereafter, the processed images undergo the segmentation stage. EasyOCR utilises the *Character Region Awareness for Text Detection* (CRAFT) algorithm [19] for segmenting the text from the background of the image. CRAFT is designed with a CNN which produces two different metrics, namely a *region score*, and an *affinity score*. The region score is used to localise all the individual characters on the provided image, whereas the affinity score is used to group these characters into a single instance. Utilising these two scores enables the CRAFT algorithm to exploit *character-level region awareness*. This newfound awareness, in turn, facilitates the detection of text with various shapes and locations on complex backgrounds.

After the segmentation step is performed, optional processing of the segmented data may then be applied.

In the next step, a recognition model ought to be employed in order to classify the segmented text. EasyOCR uses the *convolutional recurrent neural network* (CRNN) architecture [256] comprising three main parts, namely: A feature extractor, a sequence labeller, and a decoder. The CRNN architecture was specifically developed to perform image-based sequence recognition, as the authors recognised a need for a CNN-based architecture that can identify objects that tend to occur in the form of a sequence. Unlike typical object recognition tasks, which are only expected to recognise a single object, sequence recognition requires the engine to recognise a series of objects. Examples of such real-world object sequences are musical scores [130], handwritten text [153], scene text [75], and printed text [3]. Adding to the complexity of sequence recognition, is that the length of the objects might differ significantly. Popular object recognition architectures, such as *deep convolutional neural networks* (DCNN) [162], cannot be applied to sequence prediction problems where the objects vary in length, since DCNNs often operate on an input and output having similar dimensions [256]. Consequently, these architectures are unable to produce noteworthy results for variable-length label sequences.

The first part of the CRNN architecture is the feature extractor. The purpose of this component is to extract a sequential feature representation from the segmented characters received from the output of the CRAFT algorithm. It is constructed from the convolutional and max-pooling layers of a standard CNN architecture — the fully connected layers are not included. The layers of several standard CNN architectures may be used for this step, with the VGG architecture being a popular choice [141]. Feature sequence vectors are then extracted from the feature maps produced by the convolutions, corresponding to frames on the receptive field. A visualisation of feature sequence vectors and their corresponding frames are shown in Figure 3.11. These feature sequence vectors and frames are used as input for the next part of the architecture.

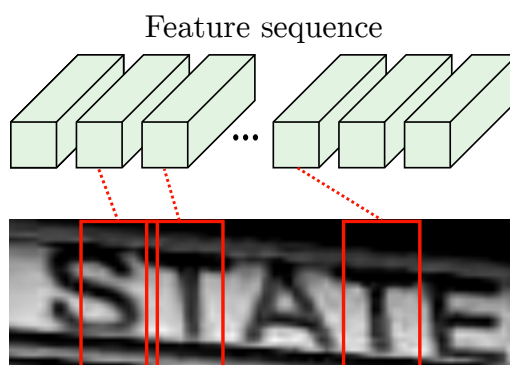


FIGURE 3.11: A graphical illustration of the feature sequences and frames extracted by the convolutional and max-pooling layers [141].

The next part of the CRNN architecture is a *deep bidirectional recurrent neural network* built on top of the feature extractor. This part is responsible for sequence labelling by predicting a label distribution y_t for each frame x_t in the feature sequence space $\mathbf{x} = x_1, \dots, x_T$ [256]. There are three main advantages when incorporating recurrent layers into the architecture. First, the RNN does not treat each character as independent from one another. Accordingly, the RNN uses image cues for image-based sequence recognition. Consider the string “il” comprising two frames. It is easier to distinguish the two characters when observing the difference between the two frames, *e.g.* the character heights in the frames, rather than simply looking at the two frames independently. Additionally, wide characters might comprise several frames, requiring

some dependency between them. Second, both the RNN and CNN may backpropagate error signals back through the network, enabling the architecture to be jointly trained in a unified network. Third, traversing from start to end, the RNN layers are able to operate on sequences of arbitrary lengths. LSTMs are the ideal RNN type for this application. Designed specifically to enable the capturing of long-range dependencies, which often occur in image-based sequences, LSTMs achieve all of the criteria supporting the case for its implementation in the CRNN architecture as the sequence labelling component. LSTMs are, however, directional, meaning it only passes past information through the module. Image-based sequences may gain insight from both the past and future frames. Accordingly, two LSTMs are combined, one forward and one backward, in order to form a bidirectional LSTM [102, 129].

The third and final part of the CRNN architecture comprises the transcription layers. Shie *et al.* [256] describe transcription as the process of converting the per-frame predictions made by the LSTM into a label sequence. Transcriptions may be *lexicon-based* or *lexicon-free*. A lexicon is a set of labels to which predictions are constrained. Examples of lexicons include: A spell checking dictionary, a set of medical terms, or a list of formal government terms. If the transcription is lexicon-based, the predictions made by the recognition model are based on choosing the label sequence in the lexicon with the highest probability. If the transcription is lexicon-free, the predictions made by the recognition model omit any lexicon constraints. The CRNN architecture utilises a lexicon-based *conditional probability* transcription approach defined in the CTC layer [101].

By utilising modern deep learning approaches and architectures, EasyOCR has shown to outperform many other prominent OCR engines [157]. The engine provides notable flexibility with functions such as segmentation modes, language selections, whitelisting and blacklisting of words, and vertical text support. Additionally, the inclusion of several changeable components enables a user to personalise the framework if unique circumstances require different detection, recognition, or decoding models. Developed using PyTorch, EasyOCR is capable of running on GPUs and can perform batch predictions, significantly increasing its prediction speed when compared with its peers. EasyOCR has proven to be a good option for recognising text in semi-structured document images (*e.g.* pdf files, receipts, and bills) [282].

3.5 Chapter summary

The aim in this chapter was to provide the reader with basic insight into the computer vision subfield of OCR so as to facilitate an understanding of the various steps and components that take place when digitising text on a document. The chapter opened, in §3.1, with an exploration of OCR engines and the typical challenges faced when digitising text within an image. This was followed in §3.2 by an in-depth discussion on each of the major phases of the typical OCR engine. Thereafter, in §3.3, a discourse was presented on the development of OCR evaluation metrics and tools, showcasing the arguments made by authors in the literature. Further elaboration on prominent OCR engines were provided in the form of a thorough discussion in §3.4, with a specific focus on the two OCR engines utilised in this thesis, *i.e.* Tesseract and EasyOCR.

CHAPTER 4

Document image enhancement

Contents

4.1 Significance of image enhancement for text recognition	61
4.2 Geometric transformations	63
4.2.1 Skew correction	63
4.2.2 Text orientation correction	66
4.2.3 Cropping	67
4.3 Pixel transformations	68
4.3.1 Line removal	68
4.3.2 Binarisation	69
4.3.3 Noise removal	72
4.3.4 Image sharpening	74
4.4 Chapter summary	76

The aim in this chapter is to review the pertinent literature pertaining to document image enhancement techniques for improved OCR recognition. The reader is introduced to the utility of document image enhancement for improved OCR recognition. The chapter opens with a discussion on the general background of OCR engines and the typical challenges faced when digitising text within an image. This is followed by an exploration of different document image enhancement techniques implemented when enhancing a document image for OCR purposes. First, geometric transformations are discussed, during which focus is placed on skew correction, orientation correction, and cropping of document images. Thereafter, a discourse on different pixel transformations is provided which are aimed at removing distortions and/or enhancing textual features. In conclusion, a concise summary is provided of the document image enhancement literature discussed in this chapter.

4.1 Significance of image enhancement for text recognition

In the preceding chapter, two prominent OCR engines, *i.e.* Tesseract OCR and EasyOCR, are discussed in detail. The flexibility of these engines are showcased during the discussion on the vast number of hyperparameters applicable. Tuning hyperparameters of these engines alone is, however, rarely sufficient as most real-world document images lack in resolution and/or capturing quality. Consequently, these images ought to undergo image processing in order to enhance the image to an acceptable quality before presenting it to the OCR engine.

Consider the image visualised in Figure 4.1, where the numeric text “12-14” is captured in a black font and with a *complex* (noisy) background. Although a human can clearly identify the numeric characters on the image, OCR engines produce extremely poor results when tasked with extracting the desired text [230]. Even the Tesseract engine, one of the most prevalent OCR engines available at present, is unable to identify a single character in this image and returns only an empty output (based on default engine settings). The best result attained by the Tesseract engine was achieved after multiple iterations of hyperparameter tuning, returning the string “T2et@ce”, blatantly failing to recognise the actual text.

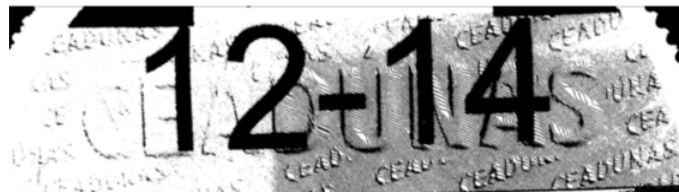


FIGURE 4.1: An example of a captured image with numeric text on a complex background [318].

The poor performance is credited to the complex background of this particular captured image. According to Ye and Doermann [311], the most notable challenge faced by the majority of vision-based systems is background complexity. There are four main characteristics affecting the complexity of an image’s background. First, the background is textured. The small background text has a distorted line pattern, increasing the level of detail captured in the image. Second, the background shading is inconsistent. The left-hand side of the background is significantly lighter than the right-hand side of the image. This might hinder the process of segmenting the full text line from the background. Third, the small background text is slightly skew, potentially rendering it more difficult for the OCR engine to detect the orientation of the actual text line. Fourth, there are numerous black blobs (*i.e.* areas with the same colour as the text) on the image border, adding to the complexity of the text segmentation.

The purpose of document image enhancement techniques is to make the text and the background easily distinguishable by reducing the complexity of the captured image. This is accomplished by suppressing undesired distortions or enhancing some characteristic relevant to the recognition of the text [104]. A cleaned version of the example’s captured image is visualised in Figure 4.2, where the complex background was successfully removed with the use of various document image enhancement techniques. The Tesseract engine was applied on the enhanced image, returning the desired ground truth output of “12-14”.

FIGURE 4.2: An example of a captured image with numeric text after the complex background was removed by means of various document image enhancement techniques [318].

It is important to note that the application of a document image enhancement technique does not always result in an enhanced captured image. There are many types of document image enhancement techniques, each with their own purpose and appropriate use case. According to Dey [63], the selection of an appropriate document image enhancement technique is the most essential step when developing a recognition system, even more important than the selection of the OCR engine itself. Consider the case where the incorrect document image enhancement techniques are applied to the aforementioned example’s captured image, resulting in the outcome

depicted in Figure 4.3. Although the applied document image enhancement techniques are deemed as standard and popular techniques, they are (clearly) not suited for this particular image with this set of degradations.



FIGURE 4.3: An example of a captured image with numeric text after the incorrect document image enhancement techniques were applied.

Due to the stages in an OCR system being organised in a pipeline fashion, the success of each stage depends on the quality of the output of the previous stage [7]. Consequently, it is crucial to understand the fundamental working of the different techniques in order to be able to apply a document image enhancement technique only when it is appropriate.

4.2 Geometric transformations

The term *geometric* transformations refers to a category of document image enhancement techniques where the geometry of the image is altered, while the pixel values of the image remain unaltered [303]. Geometric transformations are employed in order to remove geometric distortions that can accompany the image capturing process [45]. The geometric transformations explored in this section are used specifically for document image enhancement (*i.e.* techniques employed on images of captured documents) in pursuit of improved OCR performance.

4.2.1 Skew correction

A document image is considered *skew* when the text on the image requires a (relatively) small angular tilt or rotation for it to be parallel to any of the horizontal borders of the image. Documents scanned in a skew manner is one of the most common distortions obtained when capturing a document image with a flatbed scanner [222]. According to Sarfraz [245], it is more common than not for images to be scanned in with a slight angle. If an OCR engine over- or underestimates the skew angle, the line segmentation stage of the OCR engine typically fails considerably. Correcting this distortion tends to improve the performance of an OCR engine. Skew correction is defined as the process of detecting the angle with which the textual body of the document needs to be tilted in order to be parallel to a horizontal border of the image. Even a slope that is slanted with a small degree might result in a significant OCR performance drop. Consider the example visualised in Figure 4.4(a) of the document image that was captured with a slight angle of only 6° to the baseline, and Figure 4.4(b) of the same document image that was captured precisely horizontal to the baseline. This slight slope may completely deteriorate the performance of the OCR engine.

Several algorithms and techniques have been developed throughout the last few decades to solve this problem. These algorithms may be categorised into four classes, namely: *Projection profile-based approaches*, *Hough transformer-based approaches*, *Nearest neighbour approaches*, and *Interline cross correlation clustering*.

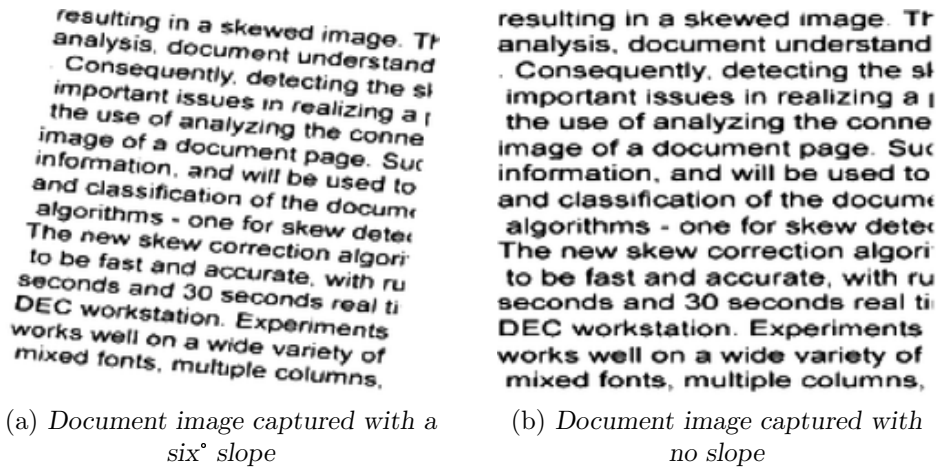


FIGURE 4.4: A visualisation of the same document captured with (a) a 6° slope and (b) with no slope [222].

A projection profile-based approach is a simple solution to detect skew text. The approach is to first transform each pixel into either a fully black or white pixel (*i.e.* binarisation), whereafter a series of horizontal lines is projected based on the number of black pixels in each row of the image. The profile having the maximum variation corresponds to the best alignment of text to the baseline. The horizontal profile projections of Figure 4.4(a) and Figure 4.4(b) are shown in Figure 4.5(a) and Figure 4.5(b), respectively. Note the clear separation of horizontal profile projections for the document image that is skew, compared with the more consistent projections of the correctly tilted document image. Several variants of this approach have been proposed in order to reduce the computation of image profiles [14, 51, 138, 146]. The performance of projection approaches are, however, limited when the image is too noisy, and/or if the skew angle exceeds $\pm 15^\circ$ [236]. Moreover, if the document does not have sufficiently numerous text — especially in paragraph format — the number of text line projections might be too limited to provide an accurate angle of skewness [193].

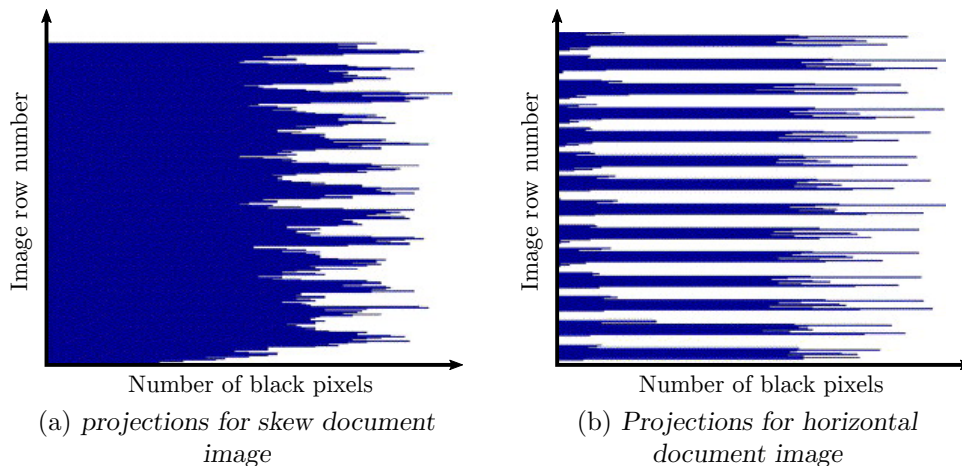


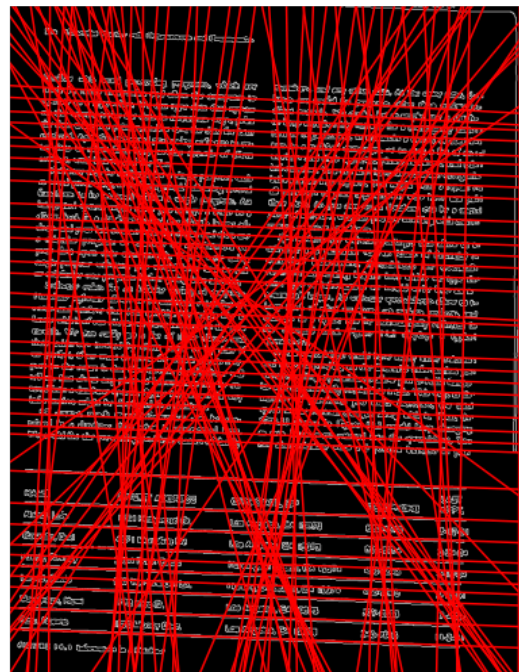
FIGURE 4.5: A visualisation of the horizontal projection lines of (a) a skew document image and (b) a deskewed document image [222].

Hough transformer-based approaches have been widely adopted to detect lines and curves found on images. The Hough transform algorithm [135] was first introduced as a line detector for

images by Duda [68]. Known as a feature extraction technique, the Hough transform algorithm converts an image from Cartesian to polar coordinates, whereafter it attempts to identify imperfect instances of a certain class shape (e.g. lines) by a specific voting procedure. In the case of searching for lines, the algorithm considers whether there is a large number of co-linear pixels on lines that are coincident with the baseline of the text [271]. Consider an input image shown in Figure 4.6(a) and its corresponding output shown in Figure 4.6(b), where the co-linear lines of a document image are graphically visualised in red. It is expected that text lines on the document image would result in several co-linear lines all having the same angle to the baseline of the image border. Therefore, the intuition is that the majority co-linear line angle ought to correspond to the angle of skewness of the text on the image. The image may then be rotated by the identified angle accordingly. Although researchers have proposed different strategies [20, 121, 164, 270, 299], Hough transformer-based approaches still require significant computational resources. Moreover, Hough transform-based approaches also require the image to contain minimal graphics or photos as possible.



(a) Input document image



(b) Document image with detected lines

FIGURE 4.6: A visualisation of the same document captured (a) as the original input image and (b) with the detected lines based on the Hough transformer-based approach [193].

Nearest neighbour approaches, also known as a bottom-up approach, aim to utilise the mutual distance and spatial relationship of objects on an image by connecting and clustering them. The angles between nearest neighbours are then collected in a histogram in order to estimate the skew angle (or skewness) of the text relative to the baseline of the image. Several researchers prefer nearest neighbour approaches [115, 143, 206]. The most notable drawback of nearest neighbour approaches, however, is that noisy subparts of characters might also be seen as objects, reducing the accuracy of this approach. Moreover, nearest neighbour approaches are markedly slow when compared with alternative approaches [222].

Interline cross correlation clustering assumes that a horizontally aligned document image presents a homogenous horizontal structure. Accordingly, these approaches aim to estimate the slope of the text by measuring horizontal and vertical deviations along the image [47]. The most notable

downside of this approach is its inability to deal with graphics and charts, as these figures usually comprise line graphs orientated in various directions. Moreover, if the angle of the slope exceeds $\pm 40^\circ$, a significant drop in performance can occur [222].

Based on the composition and quality of the input images, an appropriate skew correction approach may be implemented. It is, however, important to note that there are several exceptions that may occur, complicating the skew correction procedure. For example, if an image is scanned at an orientation that is closer to being considered upside down than right-side up, the skew correction approach will rotate the image so that the text is orientated parallel to the top border of the image, resulting in an upside-down (or flipped) document image.

4.2.2 Text orientation correction

Improper text orientation corresponds to a total rotation (*i.e.* increments of 90° angles) away from the correct and readable orientation (*i.e.* referred to as 00°), where the text lines are read from the top to the bottom of the page. When document images are captured using a flatbed scanner, it is common for the orientation of the image to be captured incorrectly to the left-hand side, right-hand side, or flipped. Improper text orientation can have a 90° , 180° , or 270° angle from the proper readable orientation of the document image. The correct orientation and the three incorrectly oriented variants of an example document image are shown in Figure 4.7. As discussed in the previous subsection, the implementation of skew correction approaches might also result in flipped document images. The performance of OCR engines is considerably limited in respect of recognising text that is not orientated correctly [217]. Therefore, it is imperative to orientate the document image to be readable for the OCR engine.

A simple, yet extremely effective method for detecting whether a document image is oriented correctly, is to take the input document image and create three additional variants of it, each with a new orientation angle. One of the four orientations is the correct variant (*i.e.* readable from top to bottom). All four variants may then be passed through an OCR engine, comparing the outputs with a dictionary corresponding to the specific language of the document. The orientation having the largest number of identified words corresponds to the correct and readable orientation of the document image. This is by far the most popular technique in literature, however, it is a markedly computationally expensive operation — each considered document image must be subjected four times to an OCR engine.

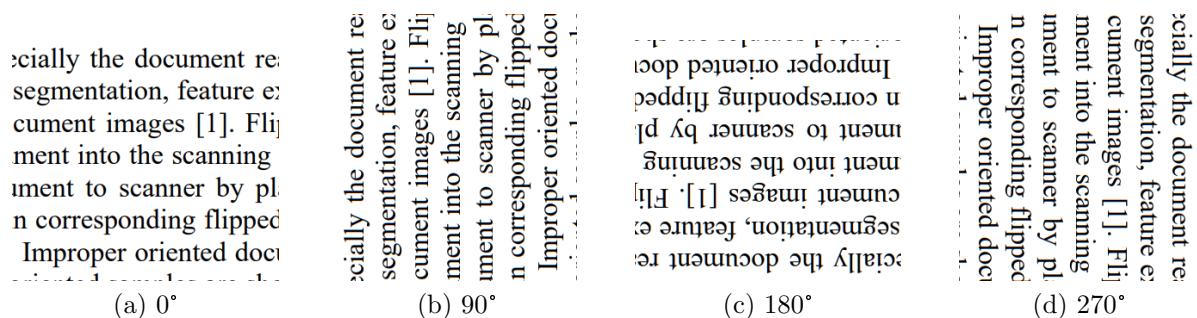


FIGURE 4.7: A visualisation of a document image with four different orientations.

4.2.3 Cropping

Document image cropping is a useful document image enhancement technique whereby only the relevant part of the document image is considered, disregarding the rest of the image [13]. The correct implementation of this document image enhancement technique may reduce the complexity of the image, since unnecessary components are removed from the image, thereby potentially improving OCR performance.

Consider the receipt document image, visualised in Figure 4.8(a) [225]. Although the receipt document image may be deemed as relatively acceptable in terms of quality, the unnecessary brown background surrounding the actual paper receipt may confuse the OCR engine. The first stage of the cropping procedure involves detecting the areas of the image which is considered to be a part of the paper document, *i.e.* the desired receipt document image. Since paper-based documents are generally printed on rectangular-shaped pages, a polygon of four vertices may describe the sought after region, *i.e.* $\{p_1, p_2, p_3, p_4\}$, where each of the four vertices comprise an x and y -coordinate. These four vertices are acquired through the implementation of three sub-steps. First, the image ought to be grey-scaled, showcasing the contrast between the actual paper-based document and the environment around it. Second, the image pixels are binarised to be either a pure white or pure black pixel, after which *blob detection* [112] is used to remove all but the largest components of the image, resulting in Figure 4.8(b). Third, the outlines of the processed image are detected using *binary morphology* [290] in combination with a *probabilistic Hough transform* [156] in order to obtain the start and end points. The second stage of the cropping procedure involves utilising the obtained coordinates and cutting the polygon out of the original image, resulting in the cropped receipt document image, showcased in Figure 4.8(c).

The aforementioned example demonstrated the cropping procedure successfully, however, it is important to note that it is dependent on the correct identification of the polygon coordinates. Important information might be cropped out if incorrect coordinates are obtained. Therefore, the application of cropping should only be applied on document images where there is a high probability of detecting the correct coordinates.

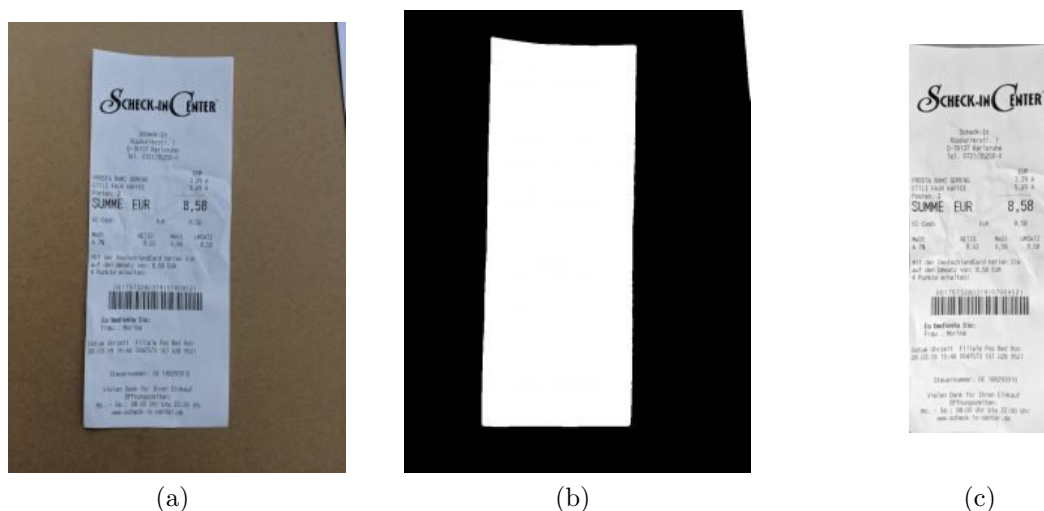


FIGURE 4.8: A visualisation of the cropping procedure on a receipt document image where (a) is the original receipt document image, (b) the detected outlines of the receipt document image, and (c) the final cropped receipt document image [225].

4.3 Pixel transformations

A document image usually comprises textual and graphical components. The graphical components may include figures, logos, graphs, table lines, border lines, watermarks, artistic symbols, and signatures. The removal of these graphical components and the enhancement of the textual components may increase the performance of an OCR engine [258]. The desired document image enhancement techniques required to achieve these enhancements might differ from the techniques implemented on non-document images. The desired document image enhancement techniques also differ between the type of document images, as different document types comprise unique compositions of the previously mentioned textual and graphical components. The main challenge faced when attempting text recognition in a document image is that the text is not located in an isolated part of the image. Therefore, in order to preprocess the document image, several pixel transformations may have to be utilised. The most prominent pixel transformation techniques, specifically for document images, include line removal, binerisation, noise removal, and image sharpening.

4.3.1 Line removal

Document images often comprise horizontal and vertical lines for page borders and/or tables. Bank cheques, mail order forms, bank statements, tax forms, payslips, and invoices are all examples of document types that utilise lines to structure textual information. An example of an information rich semi-structured document image is shown in Figure 4.9(a). The horizontal and vertical lines add value to the interpretation of the document from a visual perspective. Although these lines aid a human operator to understand the textual information, the line segments may potentially interfere with how certain OCR engines segment textual components from graphical components, and deteriorate the OCR performance if the lines are too close to characters [15]. Therefore, it is beneficial to remove these lines before OCR is performed [221]. The document image without any horizontal or vertical lines is shown in Figure 4.9(b), only comprising textual components — a suitable condition for successful text recognition.

平成24年分 給与所得の源泉徴収票

支払を受ける者	住所又は居所	氏名	(受給者番号)
		(フリガナ)	
		(役職名)	
種別	支払金額	給与所得控除後の金額	所得控除の額の合計額
給与・賞与	6 835 000	4 951 500	2 227 254
			源泉徴収税額
			34 900
控除対象配偶者の有無等	配偶者特別控除の額	控除対象扶養親族の数(配偶者を除く)	障害者の額(本人を除く)
			社会保険料等の金額
			生命保険料
			地震保険料
			住宅借入金等特別控除の額
特定老人	その他	千円	千円
1		992 454	50 000
			44 800
			140 000
源泉徴収税額			176,460
前払年金等特別控除の金額			
前払年金等特別控除の金額	H21.3.14		
配当金の金額			
配当金の金額			
配当金の金額			
配当金の金額			
中途退社・退職			
受給者生年月日			
24			○ 56 01 25
住所(居所)又は所在地	千葉県浦安市舞浜2-5-25		
氏名又は名称	夢希望〇〇株式会社 (電話) 045-000-0000		

(a) Input document image

平成24年分 給与所得の源泉徴収票

支払を受ける者	住所又は居所	氏名	(受給者番号)
		(フリガナ)	
		(役職名)	
種別	支払金額	給与所得控除後の金額	所得控除の額の合計額
給与・賞与	6 835 000	4 951 500	2 227 254
			源泉徴収税額
			34 900
控除対象配偶者の有無等	配偶者特別控除の額	控除対象扶養親族の数(本人を除く)	障害者の額
			社会保険料等の金額
			生命保険料
			地震保険料
			住宅借入金等特別控除の額
特定老人	その他	千円	千円
1		992 454	50 000
			44 800
			140 000
源泉徴収税額			176,460
前払年金等特別控除の金額			
前払年金等特別控除の金額	H21.3.14		
配当金の金額			
配当金の金額			
配当金の金額			
配当金の金額			
中途退社・退職			
受給者生年月日			
24			○ 56 01 25
住所(居所)又は所在地	千葉県浦安市舞浜2-5-25		
氏名又は名称	夢希望〇〇株式会社 (電話) 045-000-0000		

(b) Horizontal and vertical lines removed

FIGURE 4.9: A visualisation of the same document as (a) the original input image and (b) without any horizontal or vertical lines [197].

According to Reffay [220], successfully removing lines from a document image is considered to be one of the more difficult document image enhancement techniques. Line characteristics (e.g. line thickness, line type, line quality, and line curvature) contribute to the complexity

of detecting and removing line segments. Several approaches have been proposed for vertical and horizontal line removal in the literature. By far the most prominent approach is to utilise *morphological operators* [98] — a range of image processing methods that process the shape (*i.e.* the morphology) of an image based on predefined structural elements, referred to as kernels. By defining the shape and size of the morphological kernel, an operator that is sensitive to specific shapes may be constructed. In the case of line detection, the kernel is set to horizontal and vertical *black pixel runs* [83].

Two of the most widely employed and basic morphological operators are *dilation* and *erosion*. Dilation makes an object more visible by filling small holes in objects, resulting in lines appearing thicker. Erosion removes floating pixels and thin lines so that only substantive objects remain, resulting in lines appearing thinner. By setting the horizontal and vertical black pixel runs as kernels, the dilation operator outputs the *maximum* value of all the pixels that fall within the kernel shape and size, while the erosion operator outputs the *minimum* value of all the pixels that fall within the kernel shape and size [208]. Utilising the maximum and minimum values, according to the predefined kernel shape and size, all the horizontal and vertical line segments can be identified and replaced by white pixels (*i.e.* the colour of a document's background), thereby removing them from the document image. It is important to note that the success of this document image enhancement technique is notably sensitive to the selection of the kernel shape and size.

In order to elucidate this predicament, consider document image A comprising a small text font and short table lines, and document image B comprising a large text font and long table lines. Removing the vertical line segments of A requires a short vertical kernel, however, if the same short vertical kernel is applied to B, the morphological operators might identify certain textual characters as line segments, thereby removing these segments, resulting in deteriorated OCR performance. Consequently, the automation of line removal is still being actively researched in order to avoid character restoration in post-processing [221].

4.3.2 Binarisation

Document image binarisation is the process of separating the foreground and background layers of a document image. This is achieved by converting a grey-scaled document image into a bi-level document image — image pixels are transformed into either pure black or pure white [48]. When successfully applied, it is one of the most efficient document image enhancement techniques. The simplest and most established category of binarisation techniques, known as thresholding, is usually performed by comparing the pixels of an input image to a selected threshold. If the intensity of the input pixel is greater than the selected threshold value, the pixel is transformed into a white pixel, and conversely, into a black pixel. It essentially reduces the pixel information contained within the document image from 256 shades of grey (0 to 255) to only two colours [70]. As is the case with many other document image enhancement techniques, however, binarisation is not trivial, as the success of the operation is solely dependent on the components within the image. Consequently, a plethora of binarisation techniques have been developed. Thresholding-based techniques, the primary binarisation category considered for this thesis, may be subdivided into global and local image thresholding algorithms [239].

Global thresholding

Global threshold algorithms employ a single global threshold value for all pixels in a document image. Consider the grey-scaled sample image histogram graphically depicted in Figure 4.10

which showcases the number of pixels corresponding to each of the 256 grey-scale values. By utilising a global threshold, all pixels larger than the threshold (*i.e.* the background) are converted into white, thereby removing it, while converting the foreground pixels into black. The success of the binarisation process is markedly sensitive to the selection of the threshold value. Therefore, most global thresholding algorithms automatically compute a threshold value based on some type of discriminant criterion. Currently, the most popular global binarisation algorithm is *Otsu thresholding* [209]. The Otsu algorithm has no parameters to tune, and is known to handle document images with *uniform* backgrounds and *bimodal*¹ histogram distributions effectively [16, 281].

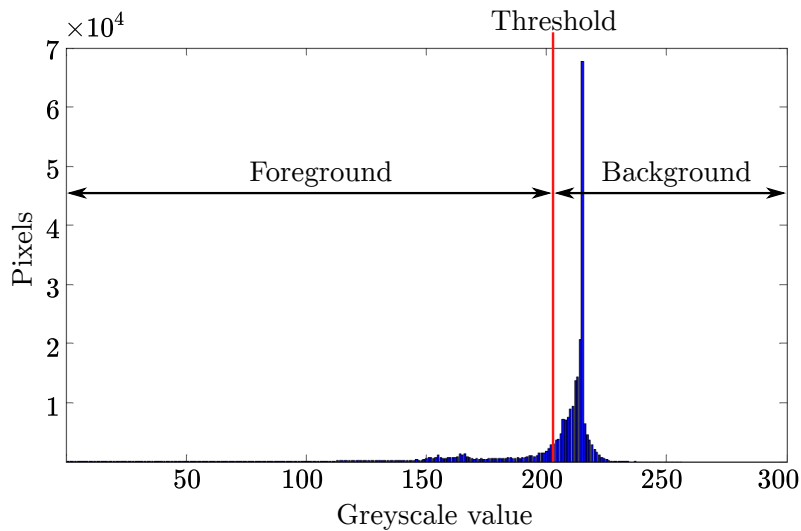


FIGURE 4.10: A sample histogram with a global threshold [16].

The core idea of the Otsu algorithm is to separate the image histogram pixel values, denoted by h , into two classes, denoted by fg for foreground and bg for background, based on a threshold, denoted by TH , which minimises the weighted variance between these two classes, denoted by σ_W^2 , or maximises the weighted variance within the two classes, denoted by σ_B^2 [192]. The respective probabilities of class occurrence (*i.e.* p_{fg} and p_{bg}) and the class mean level (*i.e.* μ_{fg} and μ_{bg}) are obtained by the following expressions

$$p_{fg} = \sum_{i=0}^{TH} p_i = \omega(TH), \quad (4.1)$$

$$p_{bg} = \sum_{i=TH+1}^L p_i = 1 - \omega(TH), \quad (4.2)$$

$$\mu_{fg} = \frac{\sum_{i=0}^{TH} ip_i}{p_{fg}} = \frac{\mu(TH)}{\omega(TH)}, \quad (4.3)$$

and

$$\mu_{bg} = \frac{\sum_{i=TH+1}^L ip_i}{p_{bg}} = \frac{\mu(L) - \mu(TH)}{1 - \omega(TH)}, \quad (4.4)$$

¹A histogram of a bimodal distribution that exhibits two peaks.

where L is the 256 grey-levels and p_i is a probability distribution for level i . Accordingly, the different class variances are given by

$$\sigma_{fg}^2 = \sum_{i=0}^{TH} \frac{(i - \mu_{fg})^2 p_i}{p_{fg}}, \quad (4.5)$$

$$\sigma_{bg}^2 = \sum_{i=TH+1}^l \frac{(i - \mu_{bg})^2 p_i}{p_{bg}}, \quad (4.6)$$

$$\sigma_W^2 = \omega_{fg} \sigma_{fg}^2 + \omega_{bg} \sigma_{bg}^2, \quad (4.7)$$

and

$$\sigma_B^2 = \omega_{fg} \omega_{bg} (\mu_{bg} - \mu_{fg})^2. \quad (4.8)$$

The Otsu algorithm considers all possible threshold values $0 \leq T \leq L$ in order to find the best threshold value that both minimises the variance within-class and maximises the variance between-class. Consequently, the optimal threshold, denoted by η , is computed by

$$\eta = \max_{1 \leq TH \leq L} \frac{\sigma_B^2}{\sigma_W^2}. \quad (4.9)$$

As noted earlier, a major pitfall of Otsu's algorithm is that it can easily fail on document images that have complex backgrounds, or document images that are significantly blurred [16].

Local adaptive thresholding

The previous subdivision of binarisation methods, *i.e.* global thresholding, compares all the pixel values in an image with a single computed threshold. In local adaptive thresholding methods, a threshold is computed for each individual pixel, based on some local statistics of the neighbourhood pixels (*e.g.* range, variance, surface-fitting parameters) [250]. The neighbourhood pixels are defined as pixels located within a local window centred on the pixel of interest. The size of the local window determines the extent to which the document image context is taken into account when computing these statistics [16]. The selection of the window size becomes considerably important when the document image contains graphical illustrations. If the window is located entirely within a graphic, the computation of the threshold does not consider the contents located close to the graphic, thereby distorting the result.

A simple local adaptive threshold technique was proposed by Niblack [203]. The technique is effective at handling cases of foreground and background pixel distribution overlap. Niblack's thresholding uses the local mean and local standard deviation, given by

$$\mu(i, j) = \frac{1}{w^2} \sum_{i'=i-w}^{i+w} \sum_{j'=j-w}^{j+w} I(i', j'), \quad (4.10)$$

and

$$\sigma(i, j) = \sqrt{\frac{\sum_{i'=i-w}^{i+w} \sum_{j'=j-w}^{j+w} (I(i', j') - \mu(i, j))^2}{w^2}}, \quad (4.11)$$

respectively, where the window size is denoted by w , the x coordinate of the pixel by i , and the y coordinate of the pixel by j . The per-pixel Niblack threshold is then computed according to

$$T_N(i, j) = \mu(i, j) + k\sigma(i, j), \quad (4.12)$$

where k is a parameter that represents the trade-off between foreground detection precision and recall. Although a parameter setting of $k = -0.2$ is generally recommended, the k and w values for each unique document image ought to be determined independently based on the document characteristics [281]. The final binarisation is then performed by means of

$$B(i, j) = \begin{cases} 0 & I(i, j) < T_N(i, j) \\ 255 & I(i, j) \geq T_N(i, j) \end{cases}. \quad (4.13)$$

Since the success of this method is dependent upon the local pixels of a central cell, document images having large open white areas are negatively affected, as it causes any background noise to be brought to the forefront, resulting in a degraded output. Although the added noise may be of significance, the immediate background around the text can be effectively identified.

In order to solve this problem, which often arises when document images have large black background areas, Sauvola and Pietikainen [246] proposed a modern variant of the Niblack threshold, expressed as

$$T_S(i, j) = \mu(i, j) \left[1 + k \left(\frac{\sigma(i, j)}{R} - 1 \right) \right], \quad (4.14)$$

where the local mean and local standard deviation are computed according to the Niblack threshold, $k = 0.5$, and R is a constant set to $R = 128$, the maximum possible standard deviation when $L = 255$. The Niblack threshold adjusts the local mean downwards based only on the local standard deviation, while the Sauvola threshold adjusts it downwards based on the product of the local mean and the local standard deviation. In windows where there is only a black background, the local mean is relatively large, resulting in $T_S < T_N$. Consequently, it is expected that the Sauvola threshold outputs fewer of these background pixels to the forefront [281]. This method is markedly effective when a document image contains uneven illumination and/or stains [48].

Graphical illustrations of the outputs of the Otsu, Niblack, and Sauvola techniques are provided in Figure 4.11, showcasing the different strengths and weaknesses of the well-known binarisation techniques.

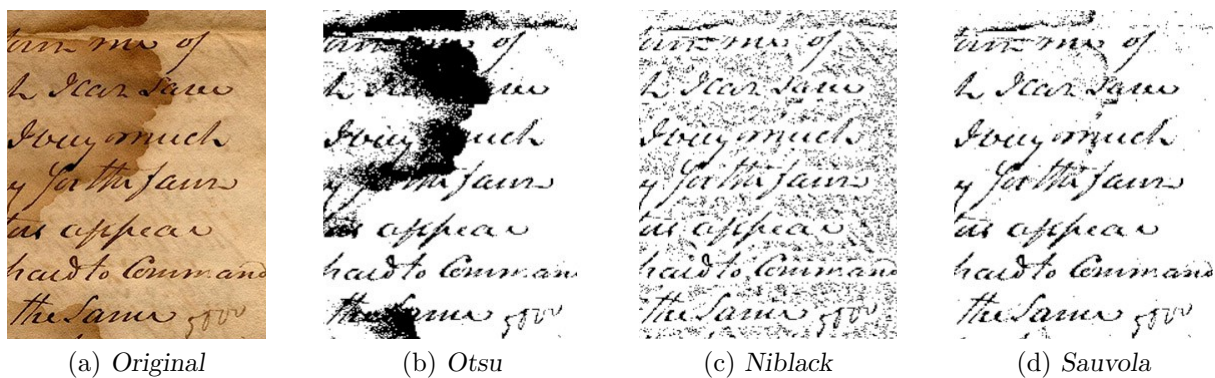


FIGURE 4.11: Examples of results obtained with well-known binarisation algorithms [229].

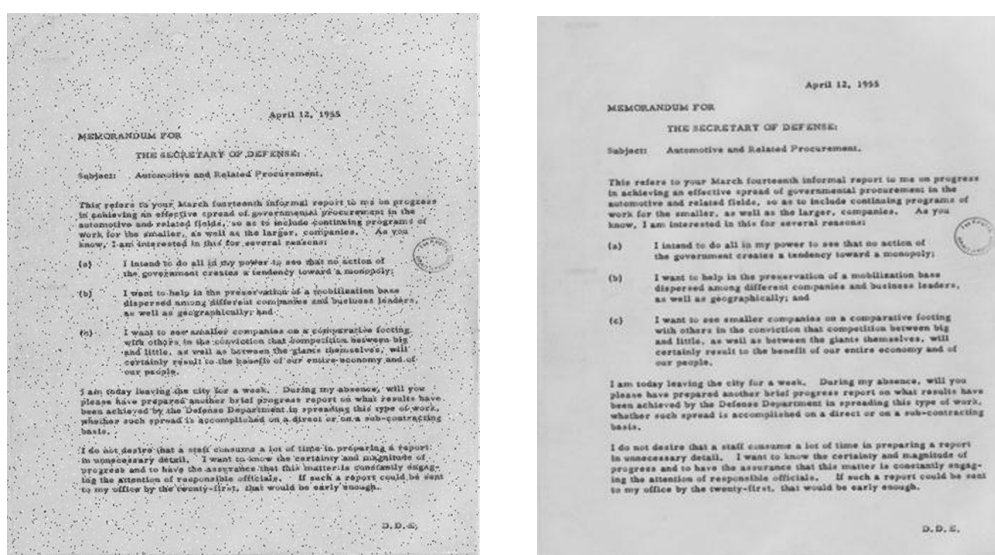
4.3.3 Noise removal

Noise removal techniques have become an essential practice in most (if not all) image processing application [71]. As previously mentioned, the quality of a document image is highly dependent on the grade of scanner or camera that captures the document, and the conditions under which the capturing was performed. Moreover, a paper document itself tends to have a reduced level

of quality due to the accumulation of defects through time and repeated usage (*e.g.* fingerprints, dirt, and wrinkles). The presence of these imperfections are referred to as image *noise*. By simply implementing a noise removal technique, the influence of these imperfections caused by the document capturing process and quality of the paper document itself may be reduced, thereby enabling the OCR engine to produce better results. A matter thoroughly discussed in literature, various denoising techniques have been developed throughout the past few decades, with the most commonly used filters being the *Median filter* [34] (used to remove salt-and-pepper noise types) and the *Gaussian filter* [61] (used to remove Gaussian and impulse noise types).

Median filter

Median filtering, classified as a non-linear filtering method, is employed in order to remove noisy and unwanted pixels from images [283]. It is known for effectively removing salt-and-pepper type noise, prevalent when copying and scanning document images, while preserving important features [74]. The filter moves through each pixel of the image with a sliding window, replacing each pixel with the median value of the pixels within the window. The median is computed by sorting all the pixel values within the window into numerical order, whereafter the considered pixel is replaced with the pixel value that is in the middle of the sorted list. As this filter utilises a sliding window to compute the median, it is important to consider how to handle the pixels located on the boundaries of the image. There are three approaches that are commonly implemented. First, the processing of the boundaries pixels can be avoided if the boundaries do not influence the contextual information of the image. Second, shrinking the window size when near the boundaries, ensuring that every window is full. Third, extending the boundaries with a padded layer with values equal to the original boundaries [283]. Visualised in Figure 4.12(a) is an input document image containing a significant amount of salt-and-pepper noise. The Median filter technique is then implemented on the document image in order to produce the denoised version thereof, visualised in Figure 4.12(b).



(a) Salt-and-pepper noise input document image

(b) Noise removed with Median filter

FIGURE 4.12: A visualisation of the same document (a) as the original input image with salt-and-pepper noise and (b) without noise after implementing the Median filter [76].

Gaussian filter

The Gaussian filter — a linear low-pass filter — has been thoroughly studied in both image processing and computer vision literature, and is deemed as one of the more successful image noise removal methods [61]. The filter is used to remove additive noise (*e.g.* Gaussian or impulse noise) which was added to the image during transmissions and/or the capturing process. The filter achieves a reduction in noise by slightly blurring the image by a Gaussian function [275]. Similar to many other filters, the Gaussian filter convolves the input document image with a sliding window and a kernel. The filter then performs a weighted average of neighbourhood of the centre pixel, where distant pixels are allocated smaller weights relative to pixels closer to the centre pixel. Accordingly, the Gaussian filter produces an output that is slightly blurred, however, it maintains the edges of the input image better than other smoothing algorithms [79]. The proper weights of the kernel may be calculated using either a one-dimensional Gaussian function denoted by $G(x)_1$, or a two-dimensional Gaussian function denoted by $G(x)_2$, respectively expressed as

$$G(x)_1 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \quad (4.15)$$

and

$$G(x)_2 = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (4.16)$$

where σ^2 is the variance of the Gaussian filters, and the size of the kernel $l(-l \leq x, y \leq l)$ is determined by omitting values smaller than five percent of the maximum value of the kernel [61]. The Gaussian filter is known as a *separable* filter, as it is a product of one-dimensional Gaussians. This is a noteworthy advantage of the Gaussian filter, as separable filters require less computing power.

Although the Gaussian filter has received considerable praise in literature, it ought to be noted that the filter is only used for specific Gaussian or impulse noise [275]. If wrongly applied to a document image, the filter may blur the detail on the image, thereby hampering the performance of an OCR engine. Consequently, several variants of the Gaussian filter have been developed in combination with edge detection techniques, in order to reduce the blurring effect on pertinent textual information [214, 285].

4.3.4 Image sharpening

Most document image capturing equipment result in a document image having some degree of *blur*. Image sharpening is a powerful image enhancement technique utilised to improve the texture of a captured image by highlighting the edges and finer details, thereby reducing the effect of image blur [234]. Specifically, when analysing document images containing fine textual detail, improving the sharpness of the image may result in significant OCR improvements. There is a tendency, however, to go to extremes by *oversharpening* the image, thereby highlighting unwanted distorted pixels. Image sharpening involves the addition of a high-pass filtered version of the input image to the input image itself [149]. The graphic in Figure 4.13 illustrates the image sharpening principle. The selection of which high-pass filter to use is critical to the success of the sharpening procedure.

Most image sharpening techniques involve applying a so-called *unsharp mask*, which amplifies the brightness difference along edges within an image. Unsharp masks are the most common type of image sharpening, and can be implemented on most images. The main idea of an unsharp mask

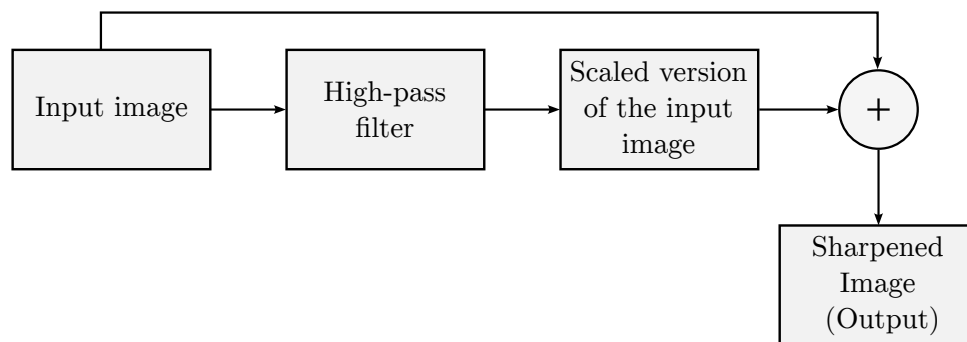


FIGURE 4.13: Image sharpening principle [16].

is to create an appearance of added detail by increasing small-scale *acutance*² [38]. It exploits a property of the human visual system called *simultaneous contrast*, where the perceived brightness of neighbouring regions is similar to the perceived level of sharpness [10]. An example of this phenomenon is shown in Figure 4.14 where an edge with low acutance, *i.e.* Figure 4.14(a), is transformed into an edge with high acutance, *i.e.* Figure 4.14(b). Note the smooth and slow increase in brightness of the original edge, compared with the significant difference in contrast of the edge after implementing image sharpening. Accordingly, by pronouncing the changes between regions, the image sharpness may be increased. An unsharp mask is acquired by subtracting a slightly more blurred version of the input image, obtained by employing a low-pass filter, from the input image itself. Thereafter, the newly attained unsharp mask is added to the input image, effectively creating a high-pass filter. This results in increased contrast along any of the edges on the image, leaving behind a sharper image [38].

It is important to note that applying too much sharpening may lead to the introduction of *halo artefacts* to the original image. These are extremely visible dark/light outlines near edges. Additionally, the unsharp mask technique is notably sensitive to noise as it tends to amplify unwanted pixels [38, 149].

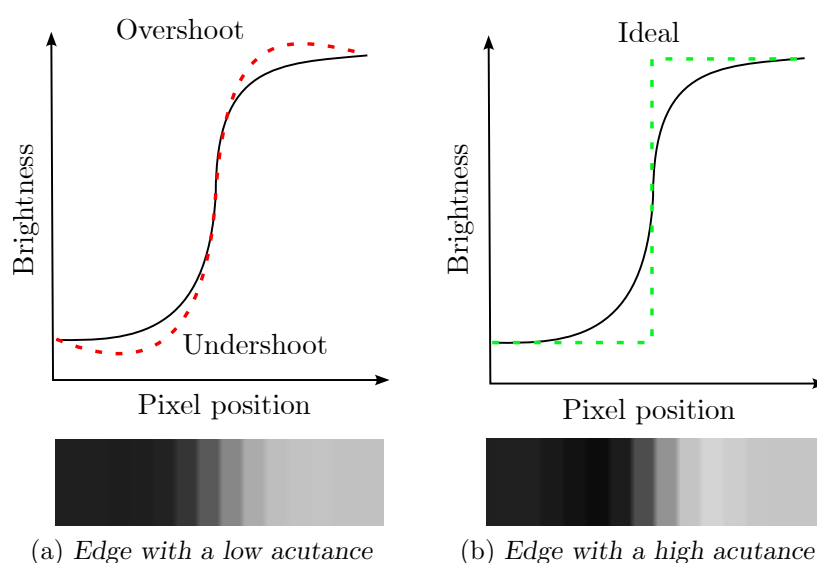


FIGURE 4.14: A visualisation of the same edge (a) with low acutance before image sharpening and (b) with high acutance after image sharpening [38].

²Describes how quickly image information transitions at an edge.

4.4 Chapter summary

The focus of this chapter was placed on the most common document image enhancement techniques found in the literature which are specifically implemented in pursuit of improved OCR performance. The chapter opened with a discussion dedicated to providing the reader with an understanding of the importance of document image enhancement in §4.1. In the next section of the chapter, *i.e.* §4.2, focus was placed on the most prevalent document enhancement techniques which alter the geometry of a document image. Thereafter, in §4.3, focus was shifted to document image enhancement techniques which alter the pixel values of an image, whereby line removal, binarisation, noise removal, and image sharpening were expanded upon. The advantages and disadvantages of each document image enhancement techniques were highlighted throughout the chapter to introduce the reader to the complexity of processing document images.

CHAPTER 5

Intelligent document image enhancement framework

Contents

5.1 A generic data mining process	77
5.2 A high-level modular data mining framework	80
5.3 Document image enhancement framework	81
5.3.1 The Configure subcomponent	83
5.3.2 The Labelling subcomponent	88
5.3.3 The Modelling subcomponent	91
5.3.4 The Analysis subcomponent	94
5.4 Chapter summary	95

The aim in this chapter is to provide the reader with a detailed top-down description of the *Intelligent Document Image Enhancement* (InDIE) framework proposed in this thesis for the purpose of enhancing document images for improved OCR performance. The chapter opens with an overview of the major steps involved in a generic data mining process, serving as a preliminary discussion for exploring the typical composition of a modular data mining framework. This modular design is then adopted as the foundational building blocks for the proposed framework structure. The proposed InDIE framework is then presented, followed by an in-depth exploration of the various subcomponents constituting the InDIE framework. The chapter is concluded with a brief summary of its contents.

5.1 A generic data mining process

In recent years, the development of specialised software has increasingly aided the data mining process [183]. Similar to concepts such as an automobile or a building, software is an engineered construct. A software programme comprises many parts or *components*. Accordingly, as in the case of a building, software may be said to have an *architecture*, organising the various components into layers (or a hierarchy) and controlling the manner according to which they interact with one another. As the architecture of a building comprises doors and windows, enabling flow and movement from one compartment to another, the architecture of a software solution may enable (or inhibit) the flow of information between components. Based on this engineering construct of the term software, three design principles may be outlined so as to guide

the development of a well-designed software product. First, the software must possess a clear and organised architecture, capable of supporting the evolution of the software product. Secondly, the architecture ought to be organised into hierarchical layers, enabling multiple technologies existing in the software product to be viewed in terms of functionality. Finally, the layers ought to comprise a modular set of components, each with its own function and purpose.

The *cross-industry standard process for data mining* (CRISP-DM) is an organised methodology of sequential steps, created to guide and govern data mining projects and software [228]. Released in 1996, the CRISP-DM methodology was developed by four industry leaders in the data mining industry, which considered the input from over 200 data mining practitioners [254]. The developers aimed at formalising the various stages of a data mining project in order to establish a common reference point amongst data practitioners. The CRISP-DM methodology, illustrated in Figure 5.1, comprises six phases in order to conceive and carry out a data mining project, enabling cyclical iterations based on the needs of the developer and other stakeholders.

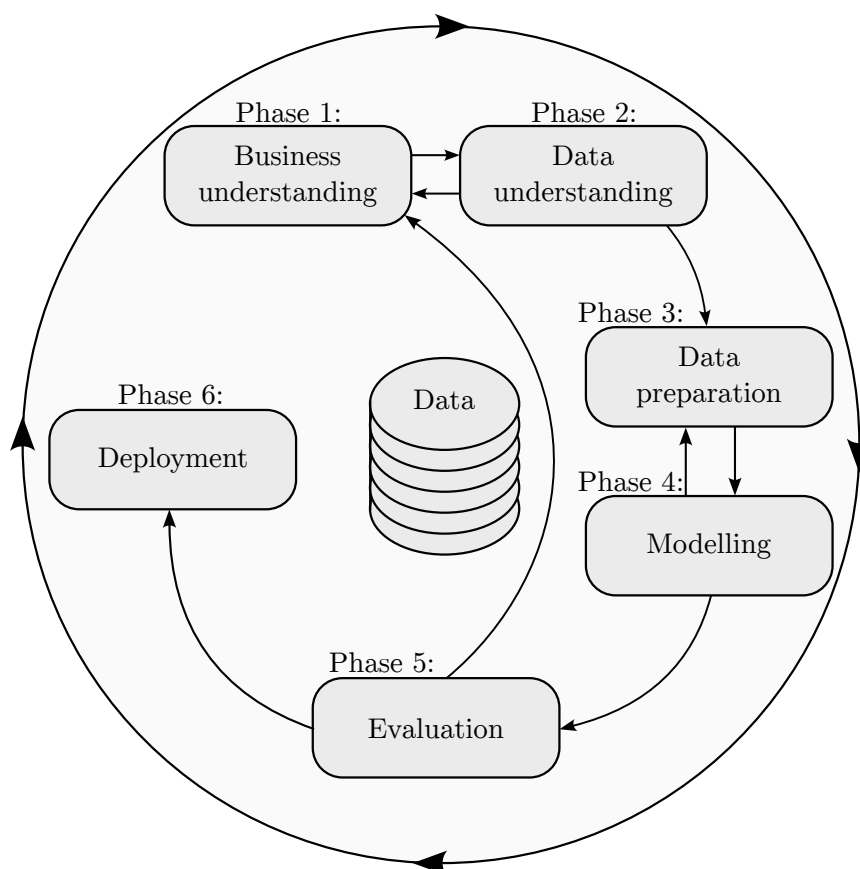


FIGURE 5.1: The six phases of the CRISP-DM reference model, adapted from Azevedo [18].

Perhaps the most important phase of the CRISP-DM reference model is *business understanding* — a phase during which the business problem is converted into a well-defined analytics problem. It is of paramount importance that the business domain and principles are fully grasped, enabling the developer to translate business requirements into design requirements [191]. This phase comprises four steps of analysis which aims to: (1) *Determine the business objectives clearly* in order to identify what precisely ought to be accomplished by the data mining project, (2) *evaluate the situation* where as much detail as possible ought to be outlined, (3) *determine the goals* of the project and how it aligns with the business, and (4) *produce a project plan* which lays out every step that is intended to be executed until the project is completed and reviewed.

The outcome of this phase is an in-depth understanding of the business context of the project, with a strategic execution plan.

The second phase of the CRISP-DM methodology is *data understanding*, which goes hand-in-hand with the business understanding phase. Focus is placed on ascertaining, assembling, and scrutinising data. This phase comprises four steps and aims to: (1) *Collect the data* indicated by the strategic execution plan, (2) *examine the data* in order to document its surface properties and/or attributes, (3) *explore the data* through visualisations and/or queries, and (4) *verify the data quality* by identifying special cases and erroneous values. After completing this phase, an understanding of the data quality, potential insights, and possible problems ought to be attained.

Data preparation, also known as *data cleaning* or *data wrangling*, is the third phase of the CRISP-DM methodology and aims at converting the data into an appropriate format for modelling. It is arguably the most time-consuming phase of a data mining project [49]. The activities involved in this phase may be implemented numerous times as new insights are gathered from the following modelling phase. The tasks associated with this phase may differ based on the type of data and the requirements of the modelling phase. Typically, data preparation comprises the following five tasks, namely: (1) *Select data* whereby a decision is made regarding the data to be included or excluded from the modelling phase, (2) *clean the data* by attempting to remove erroneous values, (3) *construct data* by engineering new features, (4) *integrate data* by creating new data sets through combining data from multiple sources, and (5) *format data* as necessary. In summary, the data preparation phase is where the domain knowledge obtained in the first phase, facilitates the formatting and creation of new and relevant features, consequently enabling better predictive performance.

The core of any data mining project is the fourth phase, referred to as the *modelling* phase. The aim of this stage is to produce the results that should satisfy the project goals determined in the business understanding phase [228]. This phase has four tasks: (1) *Select model techniques* based on domain knowledge and problem space, (2) *generate test design* where the data are separated into appropriate sets for training, testing, and validation purposes, (3) *build and train models*, and (4) *assess models* by comparing model results based on the domain knowledge and/or a set of pre-defined success criteria. This is typically an iterative phase, carried out continuously until results are obtained which are considered to be acceptable.

The fifth phase initials the *evaluation* and subsequent identification of which model meets the business requirements set out in the strategic execution plan. The phase comprises three tasks, namely: (1) *Evaluate results* by means of a comparison with the business success criteria, (2) *review the process* which involves critiquing all work completed thus far in order to determine whether the project was executed properly, and (3) *determine next steps* where the decision ought to be made regarding whether more iterations are needed or if deployment may proceed.

Deployment is the sixth and final phase of the CRISP-DM methodology. The tasks accompanying this phase may vary widely based on the business type, the chosen model, and the associated business regulations. Typically, the phase has four tasks: (1) *Plan the deployment*, where a formal document is drawn up stating common points of consideration, (2) *plan the monitoring and maintenance* to avoid issues when the model is operational, (3) *produce a final report* documenting a summary of the project for future reference, and (4) *review the project* in order to identify what was learnt and how it could be implemented better in the future.

5.2 A high-level modular data mining framework

As stated in the introductory chapter, the primary research aim in this thesis is to propose a generic framework for intelligently improving OCR performance of document images, mainly by means of the utilisation of computer vision and deep learning techniques. Taking into consideration the preceding section, during which the development of a well-designed software architecture and data mining process was explored, it is proposed that this framework should be *modular* in nature. This enables the modules proposed in this thesis to be exchangeable, modified, or deleted, thereby allowing the framework to be re-purposed for various domain-specific image types, increasing its generalisability. Furthermore, in fulfilment of the research aim, it is proposed that all modules should be conceived in a manner that encapsulates the major phases of the CRISP-DM data mining reference model, *i.e.* data understanding, data preparation, modelling, and evaluation (represented by phases 2–5 in Figure 5.1).

A high-level visualisation of the proposed framework structure is shown in Figure 5.2. The structure comprises two primary components, *i.e.* the *database*, in which relevant data are stored, and a *central functional component*, which is partitioned into four sequential subcomponents dedicated to the processes associated with *configuring*, *labelling*, *modelling*, and *analysing* data. The i -th subcomponent comprises m_i number of modules (for $i \in \{1, 2, 3, 4\}$) which are instantiated to accomplish a specific task at hand. Accordingly, if additional functionality is required, modules may be adjusted and/or new modules may be added to the subcomponents in order to fulfil those needs.

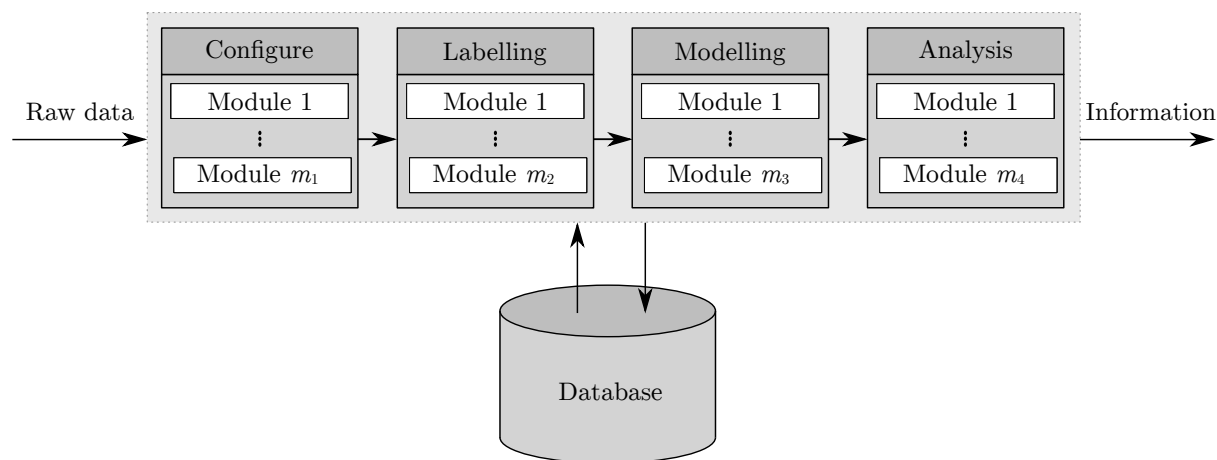


FIGURE 5.2: Schematic representation of a generic data mining framework showing the directional flow of data between the functional elements.

The database module serves as a central repository for all input, transformed, stored, and generated data used during the execution of the data mining process. This enables the central functional component to perform queries in order to retrieve relevant data required by the various subcomponents, and store the produced outputs.

The primary input to this framework is a set of raw data, comprising correlating image-based and text-based data. In this thesis, it is assumed that the *data collection* step of the data understanding phase has already been completed. The raw data are used as input to the Configure subcomponent, encapsulating the data understanding and, partly, the data preparation phases of the CRISP-DM methodology. Facilitating the *formatting*, *filtering*, *cleaning*, and *exploration* of the data attributes, the modules within the Configure subcomponent ought to yield structured data sets, considered appropriately configured, and stored in the database.

One of the most value adding (but also time-consuming and typically difficult) steps of the data preparation phase of the CRISP-DM methodology, is to construct new data attributes and/or features [202]. The data records within a data set may, for example, possess certain patterns not represented by the original input data. Accordingly, a new feature may be engineered to represent this characteristic, enriching the data set by emphasising the intrinsic data patterns. In the case of the Labelling subcomponent, the modules are responsible for the *engineering* and *assignment* of categorical labels to input images, based to some evaluation criterion, thereby creating a new feature which represents a specific characteristic of each image. Therefore, the output of this subcomponent is a new categorical feature which may be used as a target feature in the modelling stage of the data mining process.

After the data sets have been cleaned and enriched by the first two subcomponents, it is passed on to the modules within the Modelling subcomponent. These modules are responsible for the *model selection*, *model building*, and *model evaluation* steps, encapsulating all the steps of the fourth phase of the CRISP-DM methodology. The nature of the models included is determined by the specific business objectives and data characteristics. The output of this subcomponent is thus a trained model capable of categorical class predictions corresponding to the input images, and the transformed format of those images.

The predictions may then be implemented in respect of a test set. In order to ensure that the implementation of the model predictions corresponds with the business requirements of the data mining process, the results and transformed images are passed to the modules in the Analysis subcomponent. The evaluation performed by these models encapsulates the tasks set out by the evaluation phase of the CRISP-DM methodology. The four subcomponents of the proposed framework structure therefore encompass phases 2–6 of the CRISP-DM methodology.

5.3 Document image enhancement framework

From the literature reviewed in Chapters 3 and 4, it is well-established that the successful enhancement of document images may greatly improve the text recognition capabilities of an OCR engine — this is, however, no trivial task. In fact, as alluded to in §4.1, §4.2, and §4.3, the selection and implementation of some of these techniques can also result in severe deterioration of the document image, hampering the OCR performance. Despite immense effort by researchers to improve the generalisability of these enhancement techniques, the endless plethora of unique conditions that a document image may be subjected to, renders it markedly difficult to automate (and scale) the enhancement of document images for improved OCR performance. The capability to enhance document images *intelligently* for improved OCR performance by employing deep learning methods is therefore pursued by the framework proposed in this thesis.

The framework is titled the InDIE (*Intelligent Document Image Enhancement*) framework. In this section, the working of the proposed InDIE framework is formally documented, introducing the reader to the various modules and their functions within each subcomponent. A high-level overview of the framework is shown schematically in Figure 5.3 where the generic subcomponents and database (proposed in §5.2) have been populated with domain-specific modules.

The database is shown to hold five data stores (enumerated as D1–D5) where D3 holds pre-trained models, and D4 holds lexical resources relevant to the considered languages and alphabet dictionaries. Before exploring the newly populated subcomponents, however, it is necessary to state the assumptions made related to the type, format, and relationships of the different input sources

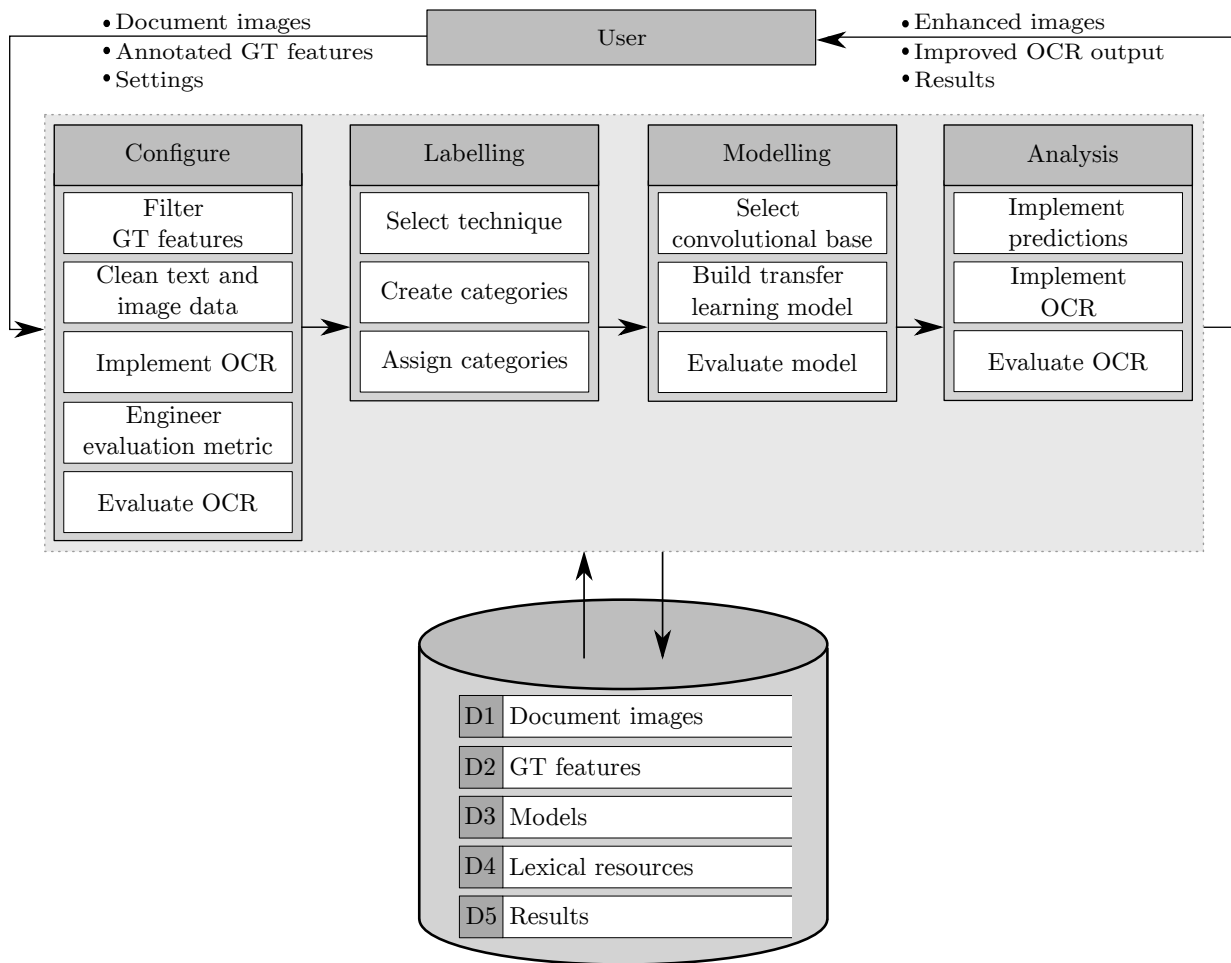


FIGURE 5.3: Schematic representation of a populated generic data mining framework with specific modules showing the directional flow of data between the functional elements.

The framework assumes (and requires) two data inputs, namely:

1. A folder of domain specific document images (specifically captured with a flatbed scanner) in PDF or PNG format, and
2. the *ground truth* (GT) feature entries comprising feature strings (*e.g.* names, amounts, addresses, cellphone numbers, purchases, and dates) in tabular format, corresponding to the captured document images (assumed to be annotated by human operators).

The framework is initialised by invoking the Configure subcomponent. This subcomponent starts by filtering out erroneous documents and its corresponding data records, followed by several data cleaning modules. Thereafter, OCR is performed (discussed in §3.2) and the output is compared with the GT feature entries (according to a predetermined evaluation metric, as reported in §3.3) in order to determine a baseline accuracy of the OCR performance. The cleaned data and base OCR results corresponding to each document image are stored in the database.

After having executed the Configure subcomponent, the document image enhancement technique(s) (discussed in §4.2–§4.3) that would produce the best OCR outcomes for each unique document image is determined in the Labelling subcomponent. This is achieved by engineering multiple enhancement procedure categories comprising unique combinations and specific

sequences of selected document image enhancement techniques and applying these procedures to each document image — creating several variants of each document image. Subsequently, the OCR operation is then performed on all the variants of each document image, facilitating a quantitative metric for identifying the enhancement procedure that would yield the best OCR performance for each individual document image. The best enhancement procedure category is then assigned to each document image, resulting in an engineered target feature employed in the following modelling subcomponent.

During the implementation of the Modelling subcomponent, transfer learning (reviewed in §2.6) is employed by selecting and utilising a pre-trained CNN architecture (explored in §2.5.2) in order to extract image feature maps from each document image. The combination of extracted feature maps and the previously engineered and assigned target feature is then used as input for a supervised machine learning procedure, whereby a selected classifier is trained to predict the best enhancement procedure category for unseen document images by learning the intrinsic patterns related to the extracted image feature maps.

In the final Analysis subcomponent, the model predictions may be evaluated by implementing the model predictions on a test set of document images, and performing OCR, thereby facilitating the direct comparison of the base OCR performance before the intelligent enhancement procedure with the OCR performance after the intelligent enhancement procedure.

The remainder of this section is dedicated to in-depth discussions pertaining to each of the subcomponents of the InDIE framework. Central to this discussion is the utilisation of *data flow diagrams* (DFDs)¹ so as to visualise and describe the modules involved within each of the subcomponents graphically, as well as the data flows, data types, and data transformations between these modules.

5.3.1 The Configure subcomponent

As mentioned above, the Configure subcomponent is the first subcomponent of the proposed InDIE framework and is employed in order to configure the input data. The subcomponent comprises five modules, numbered 1.0 to 5.0, as illustrated in the *level-one* DFD in Figure 5.4. Data stores D1, D2, D4, and D5 are utilised by the modules in the Configure subcomponent, with D1 providing storage for the document images, D2 storage for the GT features, D4 providing access to dictionaries, and D5 storage for results. Modules 1.0–3.0 are executed in succession (according to the number convention adopted), providing Module 5.0 with the GT features lists and corresponding OCR output lists, while Module 4.0 provides Module 5.0 with the engineered evaluation metric. This provides Module 5.0 with the required data to evaluate the base OCR accuracies for each document image and average base OCR accuracy for the entire data set — the primary outputs of the Configure subcomponent.

Filter features

The input received by the first module comprises data records of GT features (*i.e.* text corresponding to the document images) annotated by a human operator. In most cases, these annotations were originally performed in pursuit of unique business requirements and without considering that the annotations might be employed in data mining processes. Accordingly, the required format, quality, and type of features are determined based on what the requirements

¹A *data flow diagram* (DFD) maps out the flow of information for any process or system. The DFD generating guidelines is set out by Kendall and Kendall [150].

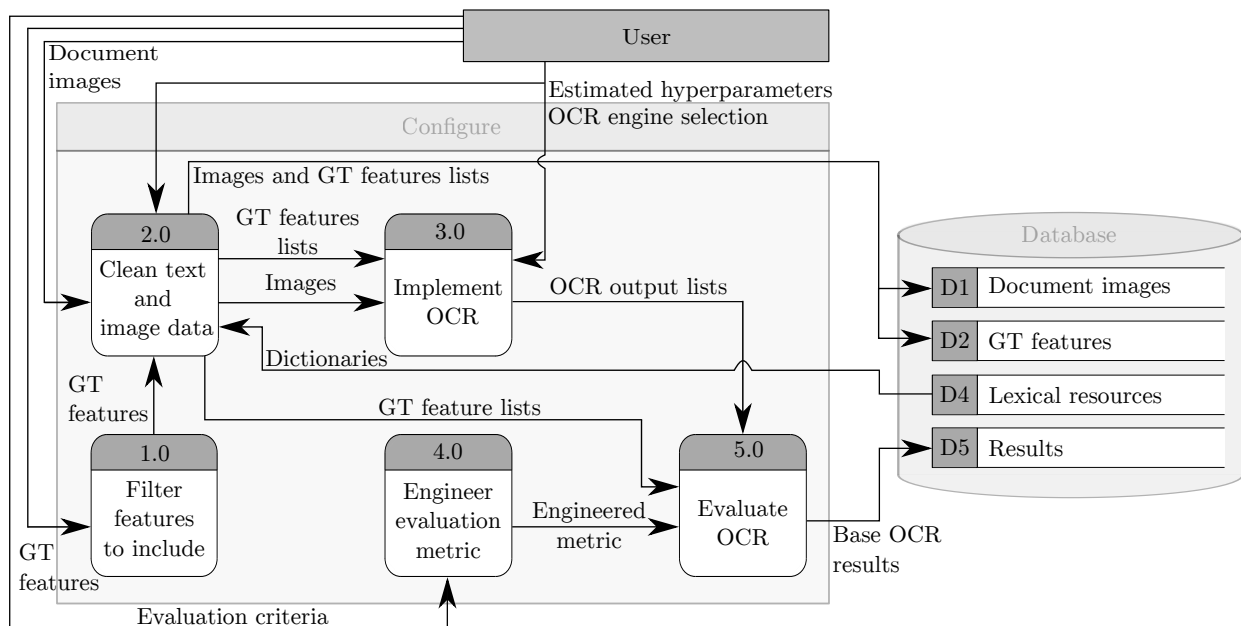


FIGURE 5.4: Level-one DFD of the Configure subcomponent.

of the original business need are. This might differ significantly from what is required by a data mining process and, in this case, an OCR data mining procedure. Consequently, some annotated GT features might not be suited for inclusion in the OCR procedure.

The first module in the Configure subcomponent functions as a filter where the user is prompted to select which GT features ought to be removed before proceeding to the following module, *i.e.* the data cleaning module. The user begins by identifying which factors should be considered when deciding whether to remove (or include) a GT feature. Domain knowledge greatly aids the user in order to determine which factors to consider, and what weight of importance should be assigned to those factors. Typical factors to consider include the following: *GT feature importance ranking*, *GT feature proximity*, *annotation accuracy*, and *annotated format*.

GT feature importance ranking requires the user to utilise attained domain knowledge and/or to consult domain experts to rank the GT features according to their relevance to the business goals. Creating a list of ranked GT features facilitates the process of deciding which features may be removed when considering all the factors discussed in the rest of this section. Consider, as an example, ranking the annotated GT features of construction company quotes. If the business goal is to compare construction company quotes based on the total cost to the hiring company, the GT feature importance of *total amount* would greatly outweigh the importance of the GT feature *address*, however, if the primary business goal is to engage with a construction company that is in close proximity to the headquarters of the hiring company, then the *address* GT feature might outweigh the *total amount* GT feature.

Spatial information refers to the physical coordinates where a GT feature is located on the document image. If GT features are typically located near one another, it is rather likely that the GT features will equally benefit from a specific enhancement technique, while if they are typically located far from each other, an enhancement technique might improve the recognition of one GT feature, but deteriorate the recognition of the other. Consider a payslip, for example, where it is typical for the name and address of the client to be located on the upper third of the document, while it is typical for the net pay to be located on the lower third of the document. If the ranking of features resulted in the address to be ranked of low importance to the business

goals, the user might consider prioritising the recognition of net pay and rather remove the address from the data mining process. Therefore, only in cases where it is applicable, it would be advantageous if the user can select GT features that are located within a close proximity to important GT features while removing those considered less important and that are located far away from the highly ranked GT features.

The accuracy of the annotations refers to how accurately a human operator annotated the text on the document in respect of the tabular data set. This is a critical factor to consider when selecting which GT features to include. Since the evaluation of an OCR operation is performed according to character- or word-level, it is of paramount importance that there is an acceptable level of veracity and credibility in the GT features. Accordingly, the user ought to examine a sample of document images and manually compare the document text with the corresponding GT feature. In order to elucidate this predicament, consider a sample of captured bank statements and a corresponding GT feature, *i.e. printed date*. If it is found, after examining several bank statements, that significant errors were regularly made with the annotation of the printed date GT feature, and that the effect of the error cannot be mitigated through data cleaning or post-processing, it might be beneficial to remove the GT feature from the data set.

The format of the GT feature refers to the difference in how the human operator annotated the GT feature, and how it is presented on the document itself. Most of the deviation in format may be resolved in the data cleaning module, however, it must be considered whether it is worth the effort to configure the GT feature into the correct format. Removing the currency symbol that accompanies a monetary value is considered an easy data cleaning procedure, however, formatting different variations of the feature *address* into a single format may prove to be a troublesome task.

Clean text and image data

After the GT features are filtered according to the procedures discussed above, the GT features and document images undergo several data wrangling steps. The primary purpose of this module is twofold — first, to remove or replace erroneous data records and document images, and second, to format the data as necessary. The module receives the document images, the filtered GT features, dictionaries, and user selected settings as input.

It is common for real-world data records to contain missing values. In the case of data records of annotated features, this may occur when either the human annotator could not clearly identify the characters of the GT feature on the document image, therefore, leaving it out, or the feature was not present on the document image. Missing GT feature entries are specifically common when the document type varies significantly in format, resulting in cases where some documents do not contain all the sought-after GT features. Since the GT features are used to measure the OCR performance, it is not practical to include a data record if some of its GT features entries are missing. Traditional approaches towards handling missing values prove to be unavailing in the case of measuring OCR performance, as the exact GT entry is required for accurate representation of the OCR performance. Popular approaches for handling missing numerical values, (*e.g.* imputation by replacing missing values with a mode, median, or interpolation value) would all yield incorrect values. Consequently, it is recommended to remove all data records with missing GT feature entries (and their accompanying document images) from the considered data sets.

Since the GT features are to be compared with the OCR output on a character- or word-level, it is important to wrangle all the text-based GT features into a similar format. Consider a GT feature entry “5031.00” and the corresponding text “R 5,030.99” as seen on the document image.

Assume an OCR engine was implemented and all of the characters were correctly recognised. For a human, there is no significant difference between the two amount values, as the first amount only differs from the second amount by a single cent. For a computer, however, these two data points have almost nothing in common. First, the GT feature entry comprises seven characters, while the recognised value comprises ten characters. Second, the GT feature entry comprises only numerical characters and a single fullstop character, while the recognised value comprises numerical characters, alphabetical characters, a comma as well as fullstop character, and a blank space. Third, the number format of the GT feature entry does not have a thousand separator, while the recognised value has a “,” as a thousand separator. Consequently, although the OCR engine recognised all the characters correctly, an evaluation metric would reflect markedly poor results. As illustrated by this single example, the format of the GT features is notably important. Therefore, the user ought to decide on a fixed standard for each GT feature and then format the data accordingly. Finally, the correctly standardised GT features ought to be formatted into a single list for comparison purposes throughout the framework.

With regards to the cleaning procedure for the document images, two operations ought to be performed, *i.e.* a skew correction operation and a text orientation correction operation. As stated in §4.2, performing these two geometric transformations will almost certainly improve the OCR performance. Consequently, it may be implemented on all document images. The user may utilise the knowledge attained from consulting the literature in §4.2, where the advantages and disadvantages of each skew correction method were discussed, in order to decide which methods to implement (based on the type and composition of the document images). Moreover, orientation correction may be performed utilising an OCR engine and a dictionary of words to compare the OCR output with. Since the accuracy of the recognised characters is not prioritised in this step, the selected OCR engine and engine parameters are not of importance.

To conclude this module, the expected output is a list of standardised text-based feature strings, and correctly orientated document images.

Implement OCR

Before the third module may commence, the user is required to select an OCR engine for this task. The OCR engine ought to be selected based on the task specifications and the document characteristics. The composition of some document images might be considered to be simple and standard, requiring minimal OCR engine hyperparameter tuning. In this case, the user might consider the Tesseract OCR engine, while if the document image composition is considered to be complex and/or non-standard, the user might require additional flexibility, thereby rather considering the EasyOCR engine instead. Consulting the literature discussed in §3.4 ought to aid the user in deciding which OCR engine is most appropriate to implement according to the available data.

The third module receives the document images from the second module, the user selected OCR engine, and the estimated engine hyperparameters. The primary function of this module is to perform OCR on the document images and produce a text-based OCR output corresponding to each document image. The process commences by preparing an experimental sample of the document images to run a few experiments on. The purpose of this step is to compare the OCR output of the experimental sample document images with their corresponding GT features in order to fine-tune the OCR engine hyperparameters. Consider the case in which a considerable number of document images comprise sentences with abnormally small blank spaces between words. This might result in the OCR engine to misinterpret these small blank spaces as merely spaces between characters, thereby deteriorating the word recognition capabilities. The user

would therefore fine-tune the OCR engine hyperparameters to be more sensitive to the size of blank spaces, enabling the OCR engine to correctly segment sentences into separate words. When selecting document images for the experimental sample, it is advised to select document images that represent the composition of various formats and characteristics of the data set. For example, if a considerable portion of the document images contains large graphics, it would be advantageous to include some of these document images in this experimental sample in order to tune the hyperparameters accordingly.

After the hyperparameters are estimated, OCR recognition may be performed on all the document images in the data set, resulting in a text output corresponding to each document image. Since the recognised text may originate from a sentence and paragraphs, several reading signs and symbols, *e.g.* “,”, “.”, and/or “()”, may be included in the OCR output. In order to compare the OCR output with the GT features, it is required to clean and format the OCR output into the same fixed standard implemented in Module 2.0. Moreover, it is common for OCR engines to output the recognised text into a single paragraph or a single sentence. Therefore, it is necessary to format the output into a list of entries, enabling the comparison of two lists, one containing the GT feature entries and the other with the OCR output strings. Accordingly, final and expected output of this module is a list of standardised text-based strings without any unintended reading signs or symbols.

Engineer evaluation metric

Engineering an evaluation metric is one of the most critical points of an OCR data mining process. The criterion according to which the performance of the OCR engine is measured directly influences several outcomes and actions in the InDIE framework. The fourth module in the Configure subcomponent functions as an opportunity to engineer an evaluation metric in order to measure the performance of the OCR engine. In pursuit of achieving the business goals, the user ought to make use of a combination of domain knowledge and an understanding of the type of data produced by the OCR output to ensure that the evaluation metric is a true reflection of the performance of the OCR engine.

As mentioned in §3.3, several approaches may be adopted to design an evaluation metric for OCR purposes. In the case of document images comprising a majority non-dictionary GT features, it is recommended that the performance of the OCR engine be measured on a word-level. The text constituting the body of information within a document commonly includes names and surnames, company names, dates, amounts, and reference codes. It would be detrimental to most business goals if even a single character is miss-recognised within an amount. Consider a quote on an invoice where the OCR engine outputs a total amount of “9000” instead of the GT amount of “2000”. By simply miss-recognising a single character, the real amount value of the invoice increased by a factor of almost five. Conversely, if the document images comprise a majority dictionary-based GT features, it might be beneficial to measure the performance of the OCR engine on a character-level, resulting in a more sensitive and accurate evaluation metric.

OCR evaluation

The primary function of the fifth and final module of the Configure subcomponent is to compute a single value that represents the OCR performance achieved by the OCR engine when implemented on the specific document image data set. The module receives the engineered evaluation metric, lists containing GT features entries, and corresponding lists comprising OCR output strings as input.

If a word-level evaluation metric was selected, the accuracy may be computed by taking the intersect of the items within each of the corresponding lists and dividing the result by the number of GT feature entries. This provides the base OCR performance achieved on each individual document image. The intersection between two lists is graphically illustrated in Figure 5.5 where the grey object represents a single list comprising all the strings recognised by the OCR engine and the green object represents the corresponding list comprising all the GT feature entries. It is expected that an OCR output list contains significantly more items in comparison with the items in its corresponding GT feature list. The average base OCR engine accuracy for the specific data set may be computed by taking the average of all the document image accuracies. The individual base OCR accuracies and the average base OCR accuracy are stored in D5, to be utilised in the subsequent subcomponents.

In summary, the Configure subcomponent facilitates the user in preparing the appropriate GT feature and document image data to be subjected to OCR recognition, followed by the primary outputs of this subcomponent, *i.e.* the individual base OCR accuracies for each document image in the data set, and the average base OCR accuracy for the entire data set.

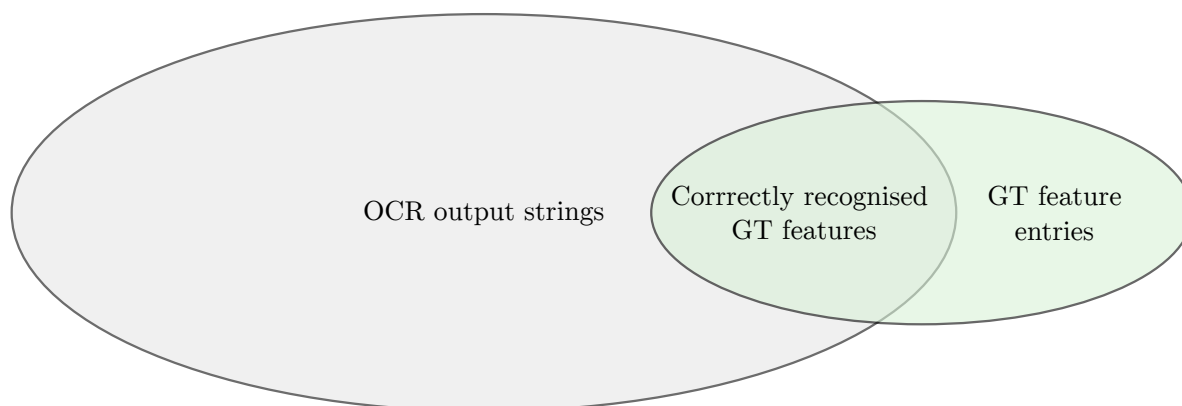


FIGURE 5.5: Graphical illustration of the intersection of the correctly recognised GT features where the grey object represents a single list comprising all the strings recognised by the OCR engine and the green object represents the corresponding list comprising all the GT feature entries.

5.3.2 The Labelling subcomponent

In order for the InDIE framework to introduce a form of *machine intelligence* into the data mining process, the framework employs an approach from the supervised learning paradigm. As discussed in §2.1.3, the supervised learning paradigm aims to predict a target feature, which is assigned to each data record in the data set, according to the intrinsic patterns embedded within the data record features. Therefore, in order to employ a supervised learning algorithm, a data set comprising two types of data are required, namely (1) a target feature assigned to each data record, and (2) corresponding descriptive features describing each data record.

The primary function of the second subcomponent of the InDIE framework, *i.e.* Labelling, is to facilitate the engineering and assignment of a categorical target feature to each document image, thereby fulfilling the first data requirement for the implementation of supervised learning. The to-be assigned target feature categories represent different document image enhancement procedures, comprising specific combinations and sequences of document image enhancement techniques, which may be implemented on each document image in pursuit of improved OCR performance.

The subcomponent is graphically illustrated in the level-one DFD provided in Figure 5.6. The subcomponent comprises three modules, *i.e.* Modules 6.0–8.0, facilitating the engineering of the categorical target feature and its assignment. The aim of this subcomponent is achieved by engineering several enhancement procedures and applying all of them on variants of each document image. Thereafter, the OCR performance of each variant is compared in order to determine the best performing enhancement procedure category for each document image. The identified category may then be assigned as the target label for each document image. The subcomponent calls data from several data stores. The document images and GT feature lists are called from D1 and D2, respectively, and are both used in the Modules 6.0 and 8.0. Moreover, Modules 6.0 and 8.0 calls the previously computed base OCR performances from D5, and also receives the selected OCR engine and estimated hyperparameters from the user.

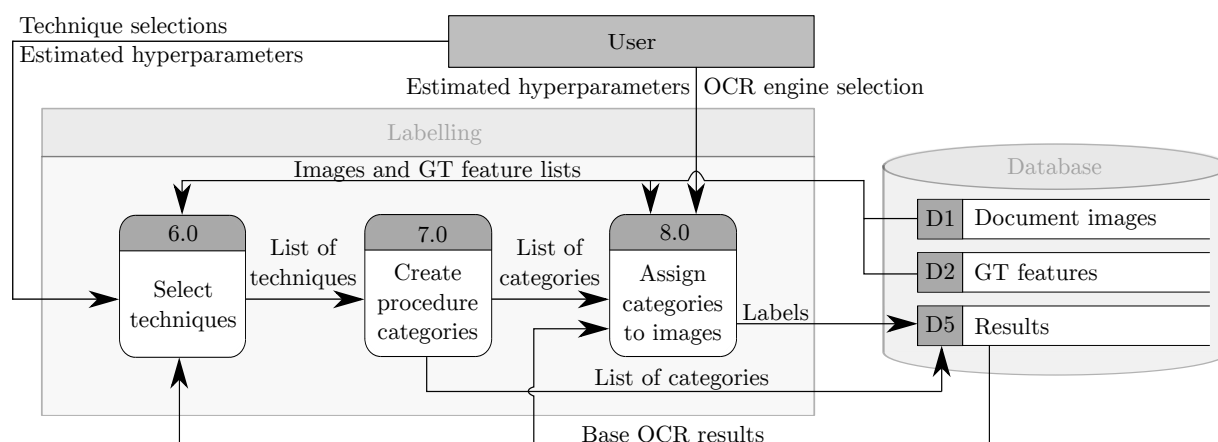


FIGURE 5.6: Level-one DFD of the Labelling subcomponent.

Select enhancement techniques

The purpose of this module is to select which document image enhancement techniques ought to be considered when engineering the enhancement procedure categories, which will be specifically tailored for the document image data set at hand. Accordingly, the primary output of this module is a list of document image enhancement techniques (and their estimated hyperparameters) which may be included in the enhancement procedures. The input received by this module comprises the document images with their corresponding GT feature lists, the base OCR results computed in the previous subcomponent, and the previously selected OCR engine (and its estimated hyperparameters).

The process commences by subjecting the previously created experimental sample of document images to various document image enhancement techniques in pursuit of identifying which techniques are most appropriate for the particular data set at hand. It is advantageous if the experimental sample comprises a diverse set of document images which closely represent the data set, not only in terms of composition (*e.g.* graphics, lines, tables, and watermarks), but also in terms of the previously obtained base OCR accuracies. Accordingly, it is recommended to expand upon the experimental sample and add document images that obtained a markedly high base OCR accuracy, and document images that obtained markedly low OCR accuracies. This will increase the probability that the user will be able to identify the effects of the transformations on document images considered to be of high and low image quality.

The process is iterative in nature, where a specific document image enhancement technique (together with several estimations of its hyperparameters) is applied to all the images in the

experimental sample, whereafter OCR is performed, and the performance is evaluated. The newly attained OCR accuracies may then be compared with the corresponding base OCR accuracies in order to identify whether the implemented technique succeeded in improving the OCR performance, or whether it deteriorated it. Moreover, tests may also be conducted where the effect of performing multiple enhancement techniques in a sequence are evaluated, thereby alluding to the creation of enhancement procedures. After performing these evaluation tests on the experimental sample of document images, the user ought to have a general understanding of how the OCR performance either improves or deteriorates the document images when the different enhancement techniques are incorporated. Accordingly, the selection of enhancement techniques (and their estimated hyperparameters) may be listed, forming the primary output of the module.

Create enhancement procedure categories

The second module within the Labelling subcomponent facilitates the engineering of the enhancement procedure categories, each comprising a single or a combination of document image enhancement techniques to be applied in a particular sequence. The engineering of these categories is a simple, yet crucial step of the framework, since the engineered categories are utilised to transform the document images. Each category is engineered based on the knowledge attained during the iterative experiments carried out on the experimental sample document images in the previous module.

Although each data mining process results in unique combinations and sequences of categories, a category comprising no document image enhancement techniques must always be included in the list of categories. This category is to be assigned to a document image when its base OCR accuracy, obtained in the first subcomponent, cannot be improved upon by the implementation of any of the engineered enhancement procedures. Moreover, the other categories may represent either a single document image enhancement technique or a combination of document image enhancement techniques, which are implemented according to a specific sequence. Accordingly, the output of this module is a list of enhancement procedure categories, constructed specifically for the data set at hand.

Four different examples of possible enhancement procedure categories are illustrated visually in Figure 5.7. The first category contains no document image enhancement techniques, as discussed above. The second category entails the sole implementation of a noise removal technique. The third category indicates the application of binarisation, noise removal, and image sharpening, in a sequential manner, whereas the fourth category refers to the sequential implementation of a line removal technique, followed by binarisation.

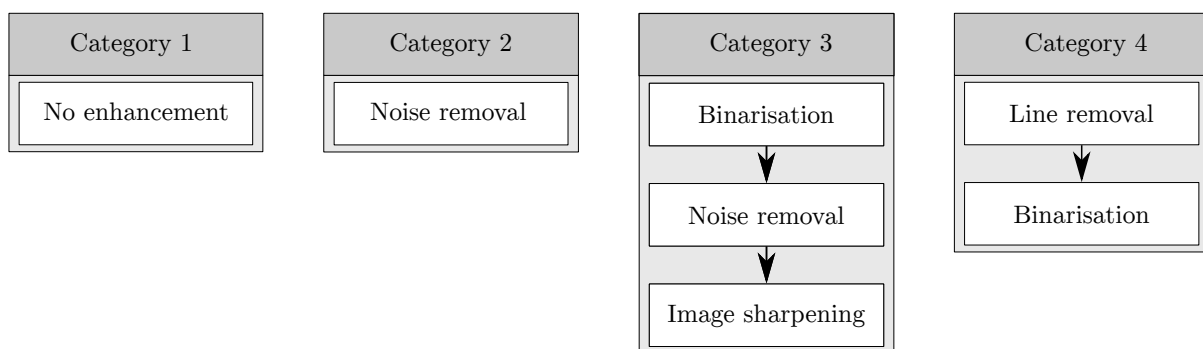


FIGURE 5.7: Four examples of possible enhancement procedure categories.

Assign enhancement procedure categories to images

After the list of enhancement procedure categories are engineered, the next step is to assign each document image an appropriate label. In order to achieve this, the third module of the Labelling subcomponent requires the list of enhancement procedure categories, the document images with their corresponding GT feature lists, the original baseline OCR results, and the OCR engine (with its estimated hyperparameters) as input.

This module functions as a data labelling pipeline. The module initialises by independently implementing each enhancement procedure category on a variant of a document image. All the different variants then undergo the same OCR recognition steps, as explained in Module 3.0. This is followed by computing the OCR accuracy for each variant document image based on the evaluation metric. The previously computed base OCR accuracy is then to be compared with all the newly attained variant OCR accuracies. The enhancement procedure category corresponding to the document image variant that showcased the best OCR accuracy therefore represents the best enhancement procedure for that specific document image (in terms of the engineered categories and selected techniques). The identified enhancement procedure category may then be assigned the target label of that document image. This process is repeated for all the document images in the entire data set, resulting in an assigned target label for each document image, yielding the primary output of the Labelling subcomponent.

5.3.3 The Modelling subcomponent

As mentioned earlier, the primary aim in this thesis is to design a generic framework for *intelligently* enhancing document images in pursuit of improved OCR performance. The third subcomponent of the framework involves the notion of *machine intelligence* and incorporating it into the data mining process through the implementation of computer vision, deep learning, and transfer learning.

As mentioned earlier, the InDIE framework employs a supervised learning approach, thereby requiring a data set comprising (1) a target feature assigned to each data record and (2) corresponding features describing each data record. In the second subcomponent of the InDIE framework, each document image is labelled according to an enhancement procedure category based on the OCR accuracy achieved (in terms of the GT features), thereby satisfying the first data requirement. The InDIE framework attempts to satisfy the second data requirement, *i.e.* features describing each document image, by invoking a pre-trained CNN model as a feature map extractor (*i.e.* the convolutional base of the model). An untrained supervised learning algorithm is added to the convolutional base, representing the classifier. The combination of a pre-trained convolutional base and the classifier forms a transfer learning model, as discussed in §2.6. Utilising the assigned target labels and the extracted feature maps, the transfer learning model may then be trained to predict an appropriate document-specific enhancement procedure to be applied.

The Modelling subcomponent comprises three modules, as illustrated in Figure 5.8. The selection of an appropriate pre-trained model to employ as a convolutional base is facilitated by Module 9.0. The transfer learning model building and fine-tuning is performed in Module 10.0. Finally, the evaluation of the model performance is conducted in Module 11.0, facilitated by a test set of document images unseen by the trained model. In terms of the flow of data and information, the subcomponent takes as input a training, validation, and testing set of document images from D1, the assigned enhancement procedure categories from D5, and several model selections and hyperparameter estimations from the user. The subcomponent stores the

final trained transfer learning model in D3, and the list of test set predictions and the model evaluation results in D5 — the primary outputs of this subcomponent.

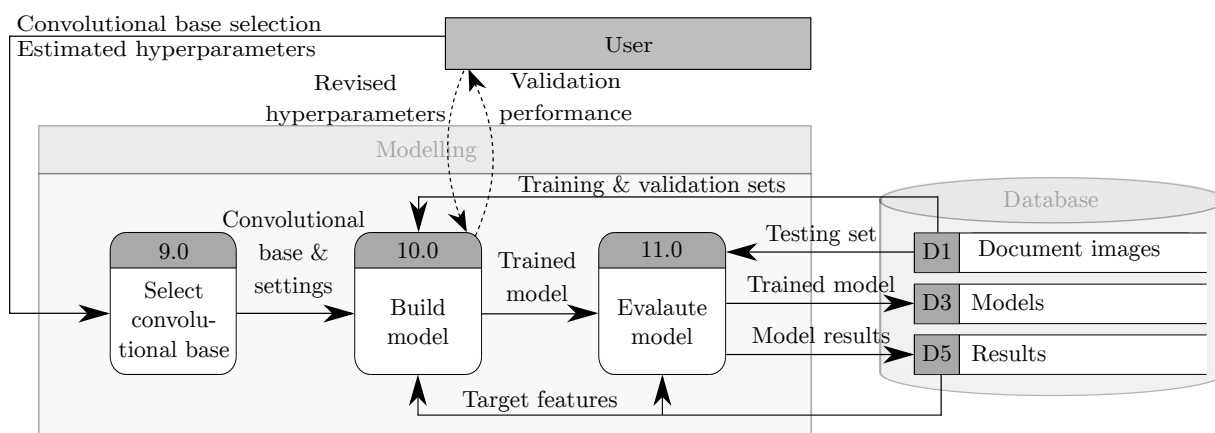


FIGURE 5.8: *Level-one DFD of the Modelling subcomponent.*

Select convolutional base

As discussed in §2.5, feature maps describing an image are usually not interpretable by humans. These image feature maps comprise numerical representations of various characteristics of an image, namely: Corners, blobs, edges, histogram of oriented gradients, textures, and scale-invariant feature transform descriptors [231]. Consequently, CNNs ought to be employed in order to extract these feature maps, as discussed in §2.5 and §2.6.

The primary function of this module is to facilitate the selection of the convolutional base, *i.e.* the first part of the model. The user has a broad range of well-known pre-trained CNN architectures to select from, as discussed in §2.5.2. The pre-trained models considered within the scope of this thesis include the AlexNet, VGG-16, and EfficientNet architectures. The user may utilise their domain knowledge of the document image data set and the literature reviewed pertaining to the advantages and disadvantages of the different CNN architectures in order to select which model to implement in the transfer learning procedure. Accordingly, the output of this module is the selected model for the document image feature extraction procedure.

Model building

The selection of the convolutional base, in combination with a classifier, is to be used to generate a deployable model for enhancement procedure predictions for document images unseen by the model. The selected convolutional base may be invoked by the training data in order to transform the image data into feature maps. Receiving the feature maps and a corresponding target feature as input, the classifier part of the model may be trained to perform the required predictions.

Module 10.0, the building of the transfer learning model, may be partitioned into two smaller processes (*i.e.* child processes), aimed at estimating the parameters of the model and the tuning of the hyperparameters, respectively. A more detailed representation of these child processes is provided in the *level-two* DFD in Figure 5.9. In this diagram, the two child processes of the model building module is visualised, each representing a step which may be iteratively executed in the training of the transfer learning model. Module 10.1 receives as input a document image training and validation sets, the engineered target feature, and the selected convolutional base

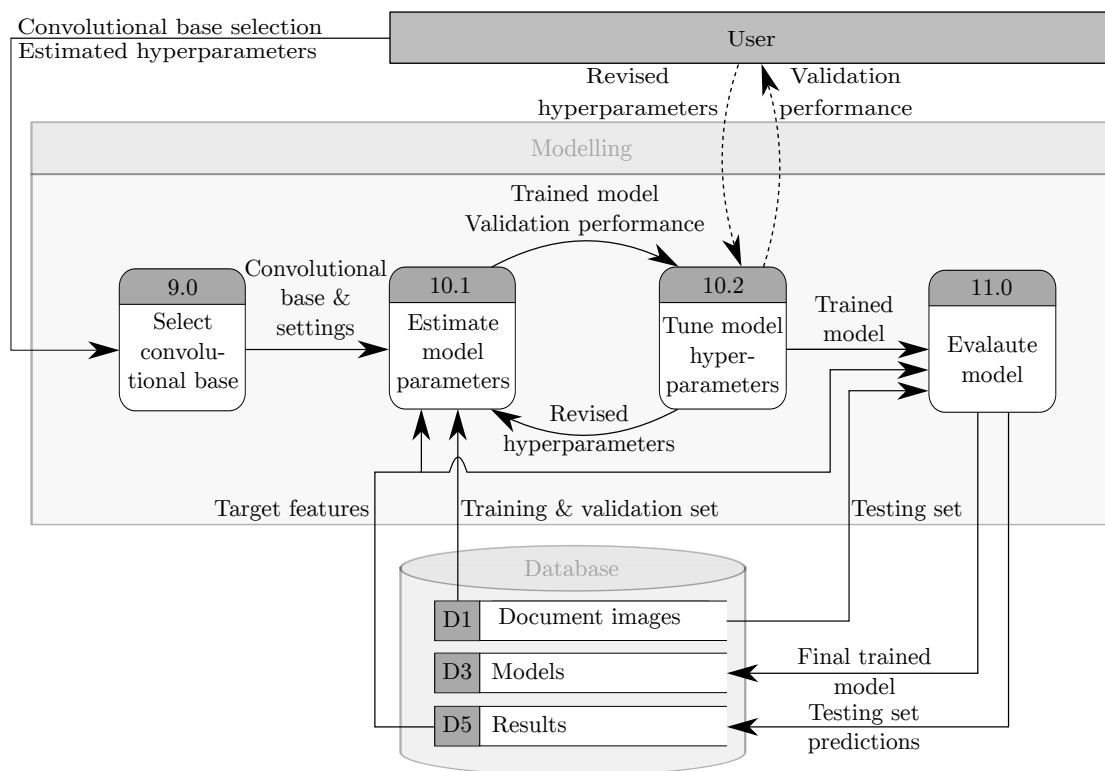


FIGURE 5.9: Level-two DFD of the Modelling subcomponent.

with estimated hyperparameters. In order to initiate the training procedure, several settings ought to be estimated beforehand. The selected convolutional base (*i.e.* the selected pre-trained CNN model) is the first part of the model to be implemented and requires numerous hyperparameters to be estimated before the feature maps can be extracted. Note that a pre-trained model was trained on a specific data set and hyperparameters, and therefore, similar data and hyperparameters are expected when employed in a transfer learning procedure. This may be exploited by the user, referencing the original hyperparameters implemented during the pre-trained model's training, and using it as an appropriate starting point for the estimations.

After the feature maps are extracted, they are flattened and, together with the target feature, presented to the classifier model, which comprises several fully connected layers. Moreover, the user must also estimate several hyperparameters for the classifier, and may use the literature reviewed in §2.1.1 and §2.5 as guidance. With the model hyperparameters estimated, the classifier may be trained, producing a trained model and validation performance results.

Module 10.2 receives the output of Module 10.1 and facilitates the tuning of the hyperparameters of the convolutional base and the classifier. The inspection of the validation performance and the input of the revised hyperparameters are shown in Figure 5.9 by the dotted lines between the user and Module 10.2. The revised hyperparameters are then fed back into Module 10.1, facilitating the model to be retrained with more appropriate hyperparameters. The loop between Module 10.1 and Module 10.2 continues iteratively until some selected stopping condition is met.

Evaluate model

The final module in this subcomponent (*i.e.* Module 11.0) facilitates the evaluation of the model on a test set, comprising document images unseen by the model training procedure. The module

receives the trained model and the test set, and saves predictions for the test set into the results datastore (D5) while saving the final prediction model into the models datastore (D3). It is important to emphasise the significance of the test set enhancement procedure predictions, as this is the primary output of the modelling subcomponent, representing the *intelligence* imbued into this data mining process.

5.3.4 The Analysis subcomponent

After the enhancement procedure predictions of the test set document images are attained from the output of the Modelling subcomponent, it is required to determine whether the *intelligence* added by the modelling subcomponent truly improved the average OCR performance of the data set. Consequently, the final subcomponent of the InDIE framework draws the attention of the user back to the evaluation of the OCR performance, this time making use of the intelligently (*i.e.* automatically) predicted enhancement procedure categories.

The working of the final subcomponent of the InDIE framework, illustrated in Figure 5.10, facilitates the implementation of the enhancement procedure predictions of the test set document images, the OCR of these enhanced document images, and the evaluation of the newly computed average enhanced OCR performance. This is achieved by executing the three sequential modules which perform the respective aforementioned steps. The subcomponent receives the enhancement procedure predictions from the results datastore (D5), the test set document images from the document image datastore (D1), the GT feature lists from the text features datastore (D2), and the original base OCR results from the results data store (D5) as input.

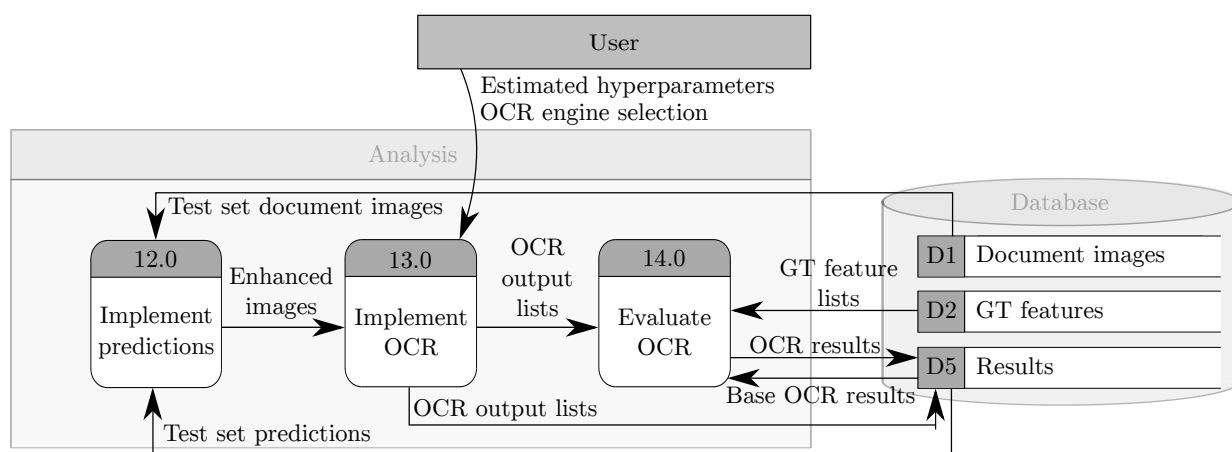


FIGURE 5.10: Level-one DFD of the Analysis subcomponent.

Implement predictions

The first module in the analysis subcomponent is devoted to the execution of the predicted enhancement procedure categories on the test set document images. The input required by this module comprises the test set document images and the predictions made in respect of the test set document images. This is a rather simple module to execute, as all the enhancement procedures are developed and tuned in the Labelling subcomponent (*i.e.* Modules 6.0–7.0). Accordingly, the procedure applied to each document image in the test set is as follows: A document image and its correlating predicted enhancement procedure category is received. The predicted enhancement procedure category is then called and executed, in sequential order, on

the document image. Consider the example case where the predicted category for a specific image is the third category illustrated in Figure 5.7. The enhancement procedure would invoke the binarisation of the document image, followed by noise removal, and finally a sharpening procedure, resulting in the (expected) intelligently enhanced document image. This procedure is repeated for each document image in the test set. Consequently, the output of this module is a data set of intelligently enhanced document images.

Implement OCR

In order to truly determine if the intelligently predicted enhancement procedures improved upon the average base OCR accuracy (*i.e.* before intelligent enhancement), it is to be compared with a newly computed average enhanced OCR accuracy (*i.e.* after intelligent enhancement). In the first subcomponent of the InDIE framework, *i.e.* the Configuring subcomponent, OCR was performed on all the document images, which includes all the document images within the test set, resulting in a base OCR accuracy for each image. Accordingly, the user can compute the average base OCR accuracy of the test set before the intelligent enhancement. The user may therefore perform the same OCR procedure (as in Module 3.0) on the intelligently enhanced document images in order to compute the new enhanced OCR accuracies, and thereafter compute the average enhanced OCR accuracy.

Module 14.0 facilitates the procedure discussed above. It receives the enhanced test set document images from Module 12.0, and the OCR engine and its settings from the user. The exact same OCR procedure performed in Module 3.0 is then performed for Module 13.0. This includes the post-processing performed on the OCR output in order to transform the data into the same format as the GT feature lists, resulting in an intelligently enhanced OCR output.

OCR evaluation

Module 14.0, the third and final module in the Analysis subcomponent, receives the new OCR output lists, generated by Module 13.0, the GT feature lists from D2, and the base OCR results computed in the Configuring subcomponent. This final module comprise two steps, first, the evaluation of the newly generated OCR output lists in terms of the GT feature lists, and then, secondly, the comparison between the average base OCR accuracy for the test set and the average enhanced OCR accuracy for the test set. The first step follows the same procedure as discussed in respect of Module 5.0, utilising the engineered evaluation metric, the GT feature text, and the newly generated OCR output lists to measure the performance of the OCR engine in terms of the intelligently enhanced document images. After the OCR performance is computed for all the intelligently enhanced document images in the test set, the average enhanced OCR accuracy is to be computed, and directly compared with the average base OCR accuracy of the test set, thereby showcasing the true value gained by utilising the InDIE framework.

5.4 Chapter summary

In this chapter, the framework proposed in this thesis, called the InDIE framework, was introduced to the reader. The chapter opened in §5.1 with a discussion of the generic data mining process, exploring the typical architecture of a software program, followed by the steps performed in a data mining process, specifically referencing the renowned CRISP-DM methodology. Utilising this attained knowledge of the data mining process, a high-level structure of a generic

framework was proposed in §5.2, encapsulating the steps of the CRISP-DM methodology. Subsequently, the InDIE framework was proposed in §5.3, containing in-depth discussions of each subcomponent, *i.e.* the Configure subcomponent, the Labelling subcomponent, the Modelling subcomponent, and the Analysis subcomponent, exploring the respective modules and flow of information within.

CHAPTER 6

Proof-of-concept implementation

Contents

6.1 Data set background	97
6.2 Implementation of InDIE framework	101
<i>6.2.1 Implementation of the Configure subcomponent</i>	102
<i>6.2.2 Implementation of the Labelling subcomponent</i>	108
<i>6.2.3 Implementation of the Modelling subcomponent</i>	114
<i>6.2.4 Implementation of the Analysis subcomponent</i>	117
6.3 Results produced by InDIE framework implementation	117
6.4 Chapter summary	128

In pursuit of demonstrating the utility of the InDIE framework proposed in the previous chapter, an instantiation of the framework is implemented in respect of an open-source data set from the literature which pertains to receipt document images. The generic modules within the four subcomponents of the InDIE framework were populated with specific algorithms and user-defined settings in order to illustrate the working of the framework. First, an overview of the selected benchmark data set is given, during which the prominence of the data set in OCR literature is addressed. Subsequently, the implementation of the four subcomponents of the InDIE framework — *i.e.* the Configure subcomponent, the Labelling subcomponent, the Modelling subcomponent, and the Analysis subcomponent — is described in detail. During the description of the implementation of each of these subcomponents, an account of the specific design, algorithm, and settings chosen for the proof-of-concept implementation are provided. Lastly, the results attained are explored and scrutinised, which includes the verification and validation procedures performed during the proof-of-concept implementation.

6.1 Data set background

The availability of an appropriate data set that contains a reasonable number of acceptable quality samples is essential for any data mining research project [215]. Ideally, if a data set is utilised for algorithm development in a data mining procedure, it ought to reflect the application domain as closely as possible, thereby enabling the researcher to project the experimentally achieved performance to real-world instances [109]. In the realm of computer vision, and specifically OCR research, gathering appropriate data for research purposes is difficult and costly, as the manual annotation of images can become mundane, time consuming, and error-prone.

In order to select a data set for the proof-of-concept phase of the research project, various requirements ought to be met. First, the document images ought to comprise mostly flatbed captured document images with corresponding high quality annotations. Second, to reflect the challenges faced by most practitioners in industry, the data set ought to be limited in respect of the extent of data entries, as annotated document image data sets are rare to acquire. Moreover, the data set must comprise document images subjected to various combinations of document image quality degradations (*i.e.* fold lines and shadows, stains, varying fonts and text sizes, and noise, to name but a few) in order to enable the implementation of the InDIE framework to attempt to intelligently predict the best (in terms of OCR performance) enhancement procedure for each document image.

While there are several prominent data sets containing annotated images for *scene text recognition* (*i.e.* images of natural scenes containing logo text, motorcar registration numbers, street signs, and billboards in the background), as referenced in the literature [43, 88, 89, 167, 263], data sets containing annotated document images are markedly scarce. There are two primary factors contributing to this data set scarcity, namely *document image annotation cost* and *information privacy*. In contrast to scene text recognition, which would usually only contain a small number of text to annotate, a single document image may comprise hundreds of text strings, resulting in the document image to be either too expensive to fully annotate, or only a few selected strings having to be annotated, according to the specific business or research goal at hand, thereby limiting the use cases of the data set. Information privacy is another noteworthy contributing factor to the scarcity of openly available annotated document images. Document images containing any potentially sensitive information of any person or company may not be openly shared to the public, thereby reducing the available data sets.

The *International Conference on Document Analysis and Recognition* (ICDAR) [134], established in 1992, is a successful conference series and is regularly referred to in literature as the premier international gathering for researchers, scientist and practitioners in the document analysis community. As a part of the scientific event, the ICDAR organises several competitions dedicated to a large set of unique document analysis problems [133]. The objectives of these competitions are to compare the quality of new and innovative document analysis algorithms on different categories of challenges. A corresponding data set is sourced for each competition by the ICDAR competition committee and released to the public beforehand. The competition committee gathers the data sets and related business goals by allowing any entity to freely share their data with ICDAR by submitting it to the ICDAR website. Throughout the past few years, a plethora of high quality data sets were released to the public through the ICDAR competitions, making it one of the most popular sources of benchmark data sets. The 2019 ICDAR competitions (and released data sets) included different challenges in several document analysis domains, namely: Handwritten historical document layout recognition [181], historical handwritten script analysis [243], document recognition (layout analysis and text recognition) [81], handwriting recognition [178], document image binarisation [184], robust reading [274], post-OCR correction [226], and chart parsing [144].

With respect to the financial, accounting, and taxation industries in particular, extracting information from receipts plays a critical role in streamlining (and automating) many document-intensive processes. With the recent breakthroughs in deep learning technologies — in terms of processing speeds and accuracy — OCR developed into a mature and viable option for many practical tasks (*e.g.* licence plate recognition and name card recognition). In terms of receipt document images, however, the OCR accuracy is required to be significantly higher when compared with general commercial tasks. Moreover, it is well-known that receipts are one of the paper-based document categories that degrades the most through repeated usage and storage.

Consequently, in most systems currently used, many financial systems are still markedly dependent on human resources.

Recognising the above challenges, ICDAR 2019 held a *scanned receipts OCR and information extraction* (SROIE) themed competition within the document recognition competition category. Recognised today as a benchmark data set for *receipts OCR analysis* is the *ICDAR 2019 SROIE data set*, comprising semi-annotated receipt document images. The data set was made available specifically for the *ICDAR 2019 robust reading challenge on scanned receipts OCR and information extraction* [227]. According to Huang *et al.* [128], this was the first open-source semi-annotated receipt data set. Compared with traditional ICDAR and other OCR data sets, the ICDAR 2019 SROIE data set came with a few unique challenges pertaining to the image quality. The data set comprises receipt document images with multiple quality issues, namely: Poor paper quality, low scanner resolution, scanning distortions, paper fold shadows, poor ink and printing quality, and stains. Moreover, in terms of document structure and layout, the receipt document images varies significantly in terms of text length, font size, handwritten text over printed text, and interfering text. In order to address potential privacy issues discussed earlier, some sensitive information fields of individuals (*e.g.* employee contact numbers, employee addresses, and employee names and surnames) were manually redacted.

The data set comprises a thousand semi-annotated scanned receipt document images in *Joint Photographic Experts Group* (JPEG)¹ format. The receipt document images originate from various industries in Malaysia, such as restaurants, grocery stores, hardware stores, health boutiques, and many more. The oldest receipts were printed in 2016, with the majority printed in 2017 and 2018. Most of the receipts were scanned using a flatbed scanner, with the occasional receipt captured by means of a smartphone camera. Although the receipts originated in Malaysia, all annotated characters and words stem from the English alphabet.

Five distinct receipts are visualised in Figure 6.1 so as to illustrate the quality issues and heterogeneity present within the data. The receipt shown in Figure 6.1(a) has several dark smudges on the top third and bottom third of the receipt paper. This increases the complexity of the image, potentially hindering the OCR engine when attempting to recognise the text. The text on the receipt visualised in Figure 6.1(b) is printed in blue ink and on a red-shaded background which varies in intensity. Moreover, handwritten words are written with black ink on the top open space of the receipt, potentially confusing the OCR engine on which colour/shade characters to search for. The third example receipt, shown in Figure 6.1(c), is clearly captured with a smartphone camera, indicated by the dark border and the fingers of the operator holding the receipt. Additionally, it also comprises blue printed text. The presence of these side-effects add complexity to the otherwise clear background. Moreover, the receipt document image was subjected to unfavourable lighting conditions, resulting in the lower half of the image showcasing a darker shade than the upper half. Noted on the top half of the receipt document image is an area digitally edited out, resulting in a white-shaded smudge. This is possibly an area where private information was written down, and redacted for confidentiality. Although redacting private and sensitive information is a required step for the utilisation of the receipt document images, the digital editing remnants may increase image complexity, hampering the OCR engine performance. The receipt shown in Figure 6.1(d) includes multiple fold lines and shadows throughout the captured receipt document image. The shadows and fold lines can significantly deteriorate the performance of an OCR engine. Lastly, a slightly skew receipt document image is visualised in Figure 6.1(e), comprising once again of blue printed text. Significantly increasing the complexity of the image, however, is the red-coloured text stamped over some of the printed

¹A digital image format which contains compressed image data.

text in the upper right-hand corner of the receipt, rendering it almost impossible for an OCR engine to read the overlapping characters.

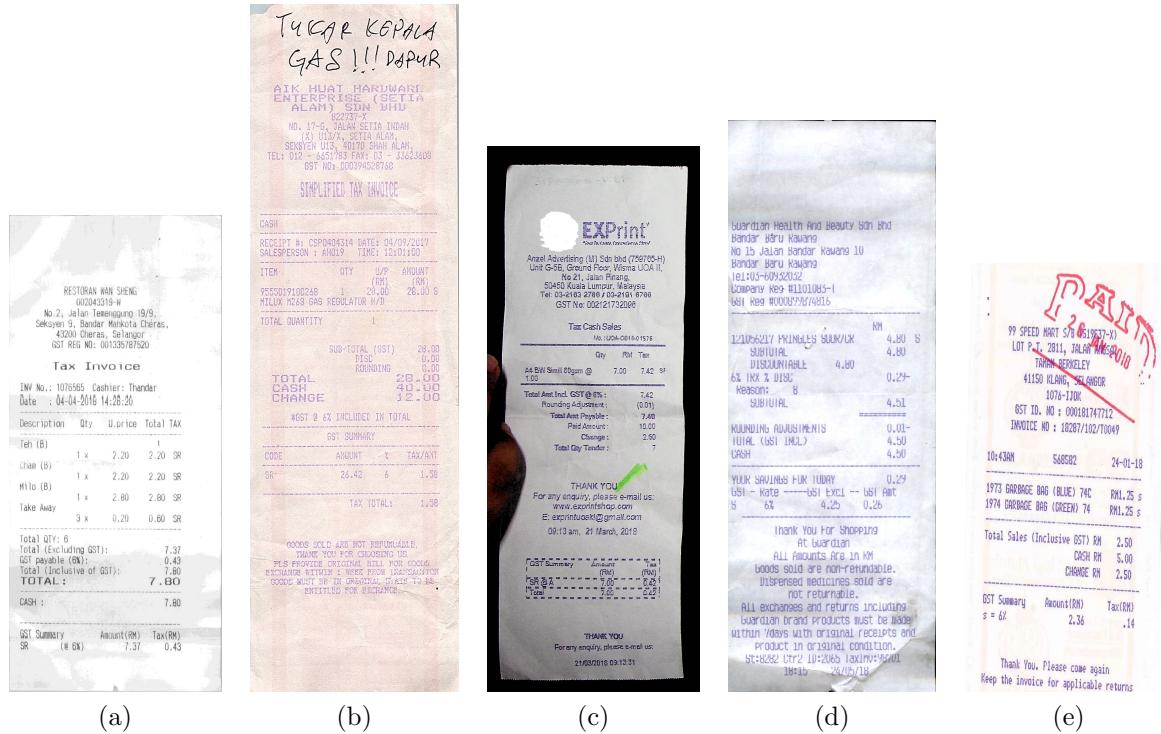


FIGURE 6.1: Examples of five degraded SROIE receipt document images [227].

Each receipt contains around four (if available on the paper-based document) key annotations provided in a *JavaScript Object Notation* (JSON)² file format, namely *company name*, *company address*, *purchase date*, and *receipt total*. Screenshots of four JSON files are provided in Figure 6.2 to visualise the attribute-value pairs of the annotations. The first example, presented in Figure 6.2(a), shows the attribute-value pairs for the aforementioned key annotation fields. Note when comparing the date field of the JSON file screenshots of Figure 6.2(a), *i.e.* 09/03/2018, Figure 6.2(b), *i.e.* 1 OCT 2017, and Figure 6.2(d), *i.e.* 2018/02/22, that the annotated date formats differ for all three receipts. This is an important finding and will be a crucial factor to explore if used to evaluate the performance of an OCR engine. Additionally, the receipt total field of the JSON files screenshots of Figure 6.2(b), *i.e.* 250.00 with no currency symbol, Figure 6.2(c), *i.e.* \$7.60 with a dollar currency symbol, and Figure 6.2(d), *i.e.* RM39.00 with the Malaysian Ringgit abbreviation, all differ in format, also requiring standardisation. It is also noted that the address field contains several punctuation characters, which include “,”, “:”, “/”, “.”, and “-”, in addition to both alphabetical and numerical characters, significantly increasing the need for standardisation and processing of the address field.

To summarise, the ICDAR 2019 SROIE data set is a renowned data set in the document analysis community, regarded as a benchmark data set for modern OCR and information extraction research projects. The data set comprises a small number of predominantly flatbed captured document images which includes limited corresponding annotations for each image. Moreover, it is evident that the captured document images possess a combination of poor quality and degraded elements which would most likely (if left as-is), significantly reduce the performance

²JSON is a data interchange format that uses human-readable text to store and transmit data objects consisting of attribute-value pairs and arrays.

of an OCR engine. Consequently, the ICDAR 2019 SROIE data set is an appropriate data set for the implementation of a proof-of-concept study in order to evaluate the potential utility of the InDIE framework. Therefore, the ICDAR 2019 SROIE data set is selected as the primary data set in respect of the proof-of-concept proffered in this research project.

```
{
  "company": "RESTAURANT SIN DU",
  "date": "09/03/2018",
  "address": "K3-113, JL IBRAHIM SULTAN 80300 JOHOR BAHRU JOHOR",
  "total": "170.00"
}
```

(a)

```
{
  "company": "ELKEN SERVICE SDN BHD",
  "date": "10 OCT 2017",
  "address": "NO 2-1 JALAN 3/137B BATU 5 JALAN KELANG LAMA 58000 KUALA LUMPUR",
  "total": "250.00"
}
```

(b)

```
{
  "company": "UNIHAKKA INTERNATIONAL SDN BHD",
  "date": "10 APR 2018",
  "address": "12, JALAN TAMPOI 7/4, KAWASAN PERINDUSTRIAN TAMPOI, 81200 JOHOR BAHRU, JOHOR",
  "total": "$7.60"
}
```

(c)

```
{
  "company": "NADEJE PRESTIGE SDN BHD",
  "date": "2018/02/22",
  "address": "LOT NO. : G1 . 116A, GROUND FLOOR , SUNWAY PYRAMID , NO.3, JALAN PJS 11/15",
  "total": "RMB9.00"
}
```

(d)

FIGURE 6.2: Examples of four JSON files comprising annotations in respect to the corresponding SROIE receipt document images.

6.2 Implementation of InDIE framework

As previously mentioned, the ICDAR 2019 SROIE data set comprises semi-annotated receipt document images recognised as having substandard and degraded quality (by design). The data set has been used in various previous OCR-related studies, therefore the results obtained during these studies can be used as a benchmark. This data set is therefore deemed appropriate for the purposes of exploring the potential benefit of implementing the InDIE framework, the main aim of which is the intelligent enhancement of document images in pursuit of improving the performance of an OCR engine, specifically by means of computer vision and deep learning techniques. Accordingly, the aim of this section is to attempt to improve the overall ICDAR 2019 SROIE data set OCR performance by predicting which enhancement procedure to implement for each receipt document image, as a proof-of-concept. The following subsections are devoted

to an in-depth description of the implementation of the various subcomponents of the InDIE framework, whereafter the results obtained are analysed.

6.2.1 Implementation of the Configure subcomponent

The first InDIE framework subcomponent to be implemented is the Configure subcomponent, comprising the five modules elaborated upon in §5.3.1. The subcomponent receives the ICDAR 2019 SROIE data set (*i.e.* the receipt document images in JPEG format and the corresponding annotations in JSON format) as input. Additionally, the user also provides an OCR evaluation criterion and a selection of the OCR engine with its accompanying settings.

Filter features

First, Module 1.0 is invoked, whereby the features that ought to be included in the OCR evaluation are selected. The ICDAR 2019 SROIE data set already has a limited number of features, as it is a semi-annotated data set — in line with the expected real-world circumstances. The four features include company name, company address, purchase date, and receipt total.

As discussed in §5.3.1, the following four factors ought to be considered when selecting features: GT feature importance ranking, GT feature proximity, annotation accuracy, and annotated format. In terms of ranking the ICDAR 2019 SROIE data features, the receipt total is seen as the most important captured feature, as a receipt is a financial document. Ranked second is the company name, followed by the date the purchase was made. Ranked as fourth would be the address, because if the company name is known, it is likely that the address can be obtained from other internet sources. In terms of GT feature proximity, it is identified that the company name and address are almost always located on the upper third of the receipts, the receipts total located on the lower third of the receipt, while the date locations fluctuates between the upper and lower thirds. Annotation accuracy is determined by inspecting a sample of the GT feature entries and comparing it with the receipt document images. After a thorough inspection, it is found that no significant annotation errors are present in the sampled annotations. Accordingly, the accuracy of the annotations is deemed as acceptable. Finally, with regards to the annotation format, it is recognised that the date feature differs in format (as showcased in Figure 6.2) and that the address and receipt total features seem to be constructed out of symbols, numerical characters, and alphabetic characters, increasing the complexity of the OCR recognition and the standardisation of the OCR output format.

After exploring and considering all of the aforementioned factors, it is concluded that the recognition of all four features are deemed as important for possible business requirements. Accordingly, business value is added by including these features, all while the annotation quality is high enough to not hamper the OCR performance. Consequently, it is decided to include all four features.

Clean text and image data

The primary purpose of Module 2.0 is to wrangle the text and image data into appropriate formats for downstream processing and analysis. In order to easily inspect and wrangle the text data, the data are extracted from the JSON files and imported into a CSV for further analysis. The first step in wrangling the text data is to remove all entries with missing values. After inspecting the ICDAR 2019 SROIE annotation data, no missing values are found, thereby

resulting in the inclusion of all the receipt document images within the ICDAR 2019 SROIE data set.

The next step, and arguably the most important data wrangling step, is to format all the entries within each feature into an appropriate format in order to be easily compared with the corresponding OCR output. It is observed that all annotations that include alphabetic characters were captured in uppercase characters, accordingly, all alphabetic characters are left as-is. It is observed that some captured addresses end with a fullstop punctuation mark, but others do not. Consequently, in order to standardise the address feature, fullstops at the end of address entries are removed. Moreover, all currency symbols and/or abbreviations in the receipts total feature are removed, standardising all the amounts as entries comprising only numerical values with a decimal separator represented by a fullstop. For wrangling the date strings into a standardised form, it is noted that the dates are annotated precisely as it is printed on the receipts. Accordingly, the global removal of punctuation symbols that will be applied to the general OCR output ought to be applied to the GT date strings. The final wrangling step for the text data involves formatting the four captured features for each individual data entry into a list of strings. This is achieved by separating the information contained within a feature annotation into multiple strings through splitting the annotation content at each black space and adding all the strings to a list, separated by commas.

With regards to the wrangling of the receipts document images, Module 2.0 suggests the implementation of two cleaning procedures, *i.e.* (1) a skew correction operation and (2) a text orientation correction. By manually inspecting the ICDAR 2019 SROIE data set, it is observed that the majority of the receipts are scanned in with an acceptable orientation, however, if implemented correctly, deskewing and orientation correction will almost always improve the performance of the OCR engine. Therefore, both operations are applied to all the receipts document images, ensuring that the performance of the OCR engine will not be hampered by any orientation defect. For the skew correction, a Hough transformer-based approach (as discussed in §4.2.1) is implemented on the receipt document images. For the orientation correction, the word comparison method (discussed in §4.2.2) is implemented, whereby the EasyOCR engine is implemented on the four different angled receipt document image variants. The OCR output is then compared with the words within a list of the 10 000 most common English words, in order to identify the correct orientation. The number of English words correctly recognised by the EasyOCR engine for the correctly orientated receipt document images ought to easily outweigh the number of English words correctly recognised for the erroneously orientated receipt document image variants.

An illustration of these operations are visualised in Figure 6.3, where Figure 6.3(a) represents an original skew receipt document image, and Figure 6.3(b) represents the deskewed and correctly orientated receipt document image. The two cleaning procedures are implemented with respect to the entire ICDAR 2019 SROIE data set, whereafter another manual inspection is performed. It is found that all the receipts are deskewed and orientated correctly.

Implement OCR

Module 3.0 guides the user in implementing an OCR engine in order to obtain the base OCR output for each receipt document image, formatted into lists of standardised text-based strings. The module receives the correctly orientated receipt document images, the GT feature lists, and a user selected OCR engine with its accompanying estimated hyperparameters — the selection thereof guided by the literature reviewed in §3.4. By inspecting the receipt document images, it is found that the composition of the images are complex (*i.e.* they comprise various logos,

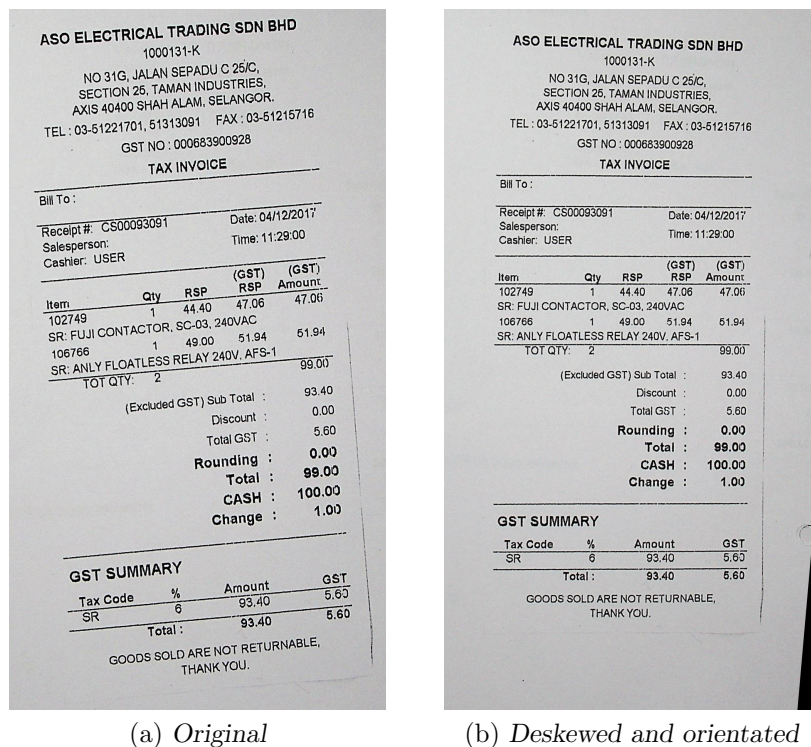


FIGURE 6.3: A visualisation of (a) an original receipt document image and (b) the receipt document image after skew and orientation correction is performed.

font types, font sizes, font colours, and pen marks) requiring an OCR engine with additional flexibility. The structure of the textual information, is well-organised and similar in terms of its layout throughout the data set. Taking the aforementioned into consideration, the EasyOCR engine is selected as the OCR engine for this data mining project as the required flexibility is provided — through the availability of numerous hyperparameters — to handle the variations of font types, colour, and sizes of the text contained within the ICDAR 2019 SROIE data set images.

For the implementation of the EasyOCR engine, several standard parameters ought to be specified. First, it is required to select a *detection language*. Although the ICDAR 2019 SROIE data set contains receipts from Malaysia, all the annotated text are in English. Therefore, the detection language is set to English. Next, the user may select to enable a GPU, if available. For this data mining project, the author employed *Google Colaboratory* [91], which provides a 12GB *NVIDIA Tesla K80 GPU*. Accordingly, the GPU parameter was set to true.

The implementation of Module 3.0 requires the inspection of a small experimental sample of the receipt document images used to fine-tune the hyperparameters of the OCR engine. The sample ought to include receipt document images representing various formats and compositions of the overall data set. Accordingly, 20 receipt document images are selected to form part of the experimental sample, three of which are visualised in Figure 6.4 to showcase the various characteristics. The receipt document image showed in Figure 6.4(a) comprise several font types and sizes, while the receipt document image in Figure 6.4(b) has a distorted red-shaded background, a red strip on the left-hand border, and with blue font. The third receipt document image is visualised in Figure 6.4(c), showcasing the receipt document image scanned as an A4 size image, a characteristic observed in a considerable portion of the ICDAR 2019 SROIE data set.

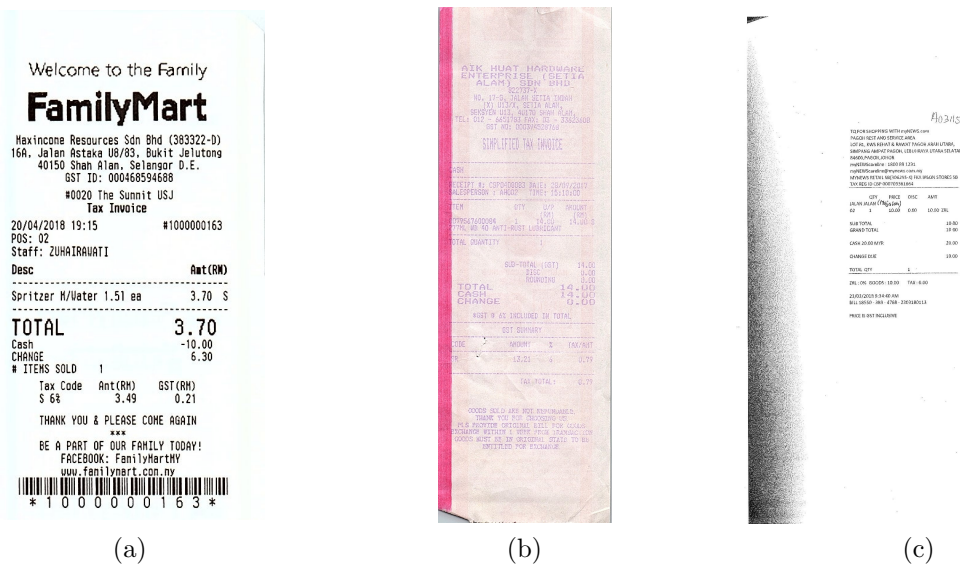


FIGURE 6.4: A visualisation of receipt document images where (a) represents a receipt document image with variation in font sizes and types, (b) represents a receipt document image with a red background and blue font colour, and (c) represents a receipt document image scanned in as an A4 size image.

The EasyOCR engine is employed on each receipt document image in the experimental sample, producing an output of recognised text strings, whereafter it is compared with the corresponding GT feature lists. The recognition errors are then identified and the hyperparameters are fine-tuned in order to mitigate these recognition errors. The process is iteratively repeated until acceptable outputs are obtained for the experimental sample.

By exploring the OCR output obtained from the 20 experimental receipt document images, it is found that the variety of font sizes require that the detection text boxes ought to be fine-tuned for the ICDAR 2019 SROIE data set. It is conjectured that the variation in font sizes may result in the OCR engine confusing fonts with large spacing between characters as blank spaces, splitting a single word up into multiple words. This is especially prevalent when detecting the numeric strings with decimal numbers (*e.g.* amounts with cents) as some fonts result in abnormal spacing after a fullstop reading sign. Accordingly, it is noted that several “amount” strings are (incorrectly) separated into two strings. EasyOCR provides various hyperparameters for this purpose. The *add_margin* hyperparameter may be tuned to extend the bounding box in all directions by a certain value. The default value is set to 0.1. Consequently, in order to reduce this occurrence, the *add_margin* hyperparameter is increased to 0.3, enlarging the detection text box, and thereby reducing the errors. Moreover, the *width_ths* hyperparameter is also fine-tuned for this purpose. This setting allows the user to increase or decrease the maximum distance between two detection text boxes before the boxes must merge into one. The default value is 0.5, however, in order to reduce the aforementioned error, the value is significantly reduced to 0.15.

After the hyperparameters are estimated, the EasyOCR engine is employed on the entire ICDAR 2019 SROIE data set, resulting in a text output (in a paragraph form) corresponding to each receipt document image. Next, the output is cleaned and transformed into the required list format. Several punctuation characters are removed from the recognised output, which includes “ , ”, “ (”, “) ”, “ : ”, “ ; ”, “ ’ ”, “ # ”, “ { ”, “ } ”, “ [”, and “] ”. Similar to the cleaning of the GT features, all currency symbols and/or abbreviations are removed. This is achieved by removing the strings “\$” and “RM” if they appear first in a string comprising solely

other numeric characters and/or a fullstop. *Regular expressions* (Regex), referred to as a mini programming language, is employed for the removal of these unwanted character strings [288]. Additionally, all the alphabetic characters within the output strings are converted into uppercase characters since the GT features are captured in uppercase characters.

The EasyOCR engine outputs the recognised text of each receipt document image into a single paragraph, whereby the recognised words are separated where a blank space character is located. Consequently, blank spaces are removed, and the output is formatted into list entries called OCR output lists, equivalent to the standardised GT feature lists. This enables the GT feature lists to be compared with the corresponding OCR output lists.

Engineer evaluation metric

The fourth module provides the user with the opportunity to engineer the evaluation metric used to compare corresponding GT feature lists with OCR output lists. As discussed in §3.3, OCR performance may be evaluated on a character- or word-based level. The annotations of the ICDAR 2019 SROIE data set comprise company names, dates, addresses, and amounts. Accordingly, receipts are classified as financial documents. Since the application of OCR on financial document images require a higher accuracy than standard scene text recognition or other non-numeric document image types, a word-based evaluation metric is selected, *i.e.* all the characters within a string ought to be recognised correctly for the string to be regarded as an accurately recognised string.

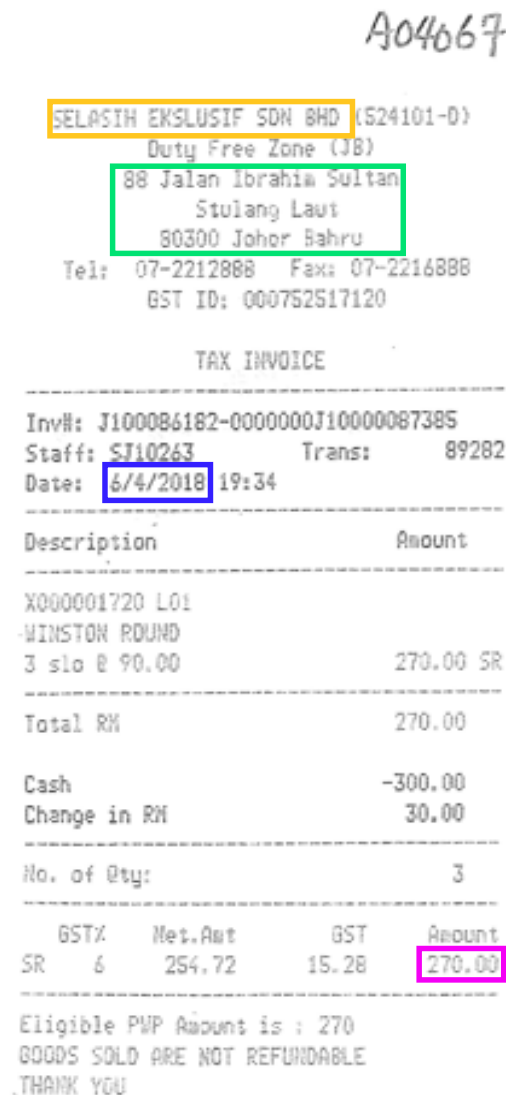
By inspecting the JSON files visualised in Figure 6.2, it is observed that the GT features such as company name, date, and address sometimes comprise several strings. Since the scope of this data mining project is to simply improve the overall OCR performance, and not to specifically extract specific targeted features, it is decided that each string would have the same weighting to the evaluation metric. Consequently, some features will have a greater impact on the OCR performance than others, as some features tend to have more strings than others. It is, however, noted that a user can easily add specific weightings to the features for evaluation purposes, if so required.

In order to elucidate the working of the evaluation metric, consider the receipt document image visualised in Figure 6.5(a), and its corresponding JSON file shown in Figure 6.5(b). The company name feature is indicated on the receipt document image by the yellow border, the date by the blue border, the address by the green border, and the receipt total by the purple border.

The company name is located on the top text line of the receipt and comprises four separate strings. The address is located on the upper third of the receipt, beneath the company name, and crosses three text lines while comprising nine separate strings. The date is located roughly in the middle of the receipt and comprises only a single string of numeric characters, separated by the “/”-symbol. Finally, the receipt total is located in the lower third of the receipt, comprising only a single string with no alphabetic characters.

The corresponding GT feature list (*i.e.* the output of Module 2.0) is shown in Figure 6.6(a), while the corresponding OCR output list (*i.e.* the output of Module 4.0) is shown in Figure 6.6(b). The GT feature list only contains the 15 annotated strings, while the OCR output list contains 68 strings, comprising all the OCR strings found on the receipt document image. Although it is found that all 15 GT feature strings are recognised by the OCR engine, not all are recognised correctly. Visualised in Figure 6.6(b), the correctly recognised GT strings are highlighted with a green border, while the incorrectly recognised GT strings are highlighted with a red border. Even though most of these errors are obtained from only a single character incorrectly recognised, the

evaluation metric is word-based, therefore all the characters in a string ought to be correctly recognised for the string to be deemed as accurate. Four of the 15 GT strings were incorrectly recognised, therefore, the EasyOCR engine obtained a base OCR accuracy of 0.73 for this specific receipt document image.



(a) Receipt document image

```
{
  "company": "SELASIH EKSLUSIF SDN BHD",
  "date": "6/4/2018",
  "address": "88 JALAN IBRAHIM SULTAN STULANG LAUT 80300 JOHOR BAHRU",
  "total": "270.00"
}
```

(b) Receipt JSON file containing annotations

FIGURE 6.5: A visualisation of (a) an example receipt document image with GT features visualised with coloured borders and (b) its corresponding JSON file with GT features visualised with coloured borders.

[SELASIH, EKSLUSIF, SDN, BHD, 6/4/2018, 88, JALAN, IBRAHIM,
SULTAN, STULANG, LAUT, 80300, JOHOR, BAHRU, 270.00]

(a) Receipt GT feature list

[SELASIH, EKSLUSIE, SDN, BHD, 524101-D, OUTY, FREE, ZONE,
IJB, 88, JALAN, IBRAHIM SULTAN, STULANG, LAUT, 80300,
JOHOR, BAHRU, TEL, 07-2212BBB, FAX, 07-22168BB, GST, ID,
000752517120, INV, J1000B61B2-000000J100000B73B5, STAFF,
SJ10263, TRANS, 89282, DATE, 6/4/2016, 1934, DESCRIPTION,
AMOUNT, X000001720, LOI, WINSTON, ROUND, 3, SLO, 8, 90.00,
270.00, SR, TOTAL, RM, 270.00, NO., OF ETY, 3, GSTI, NET.MT,
GST, ANIUNT, SR, 6, 254, 72, 15.2B, 270.00, ELIGIBLE, PWP,
ACOUNT, IS, 270, GOODS, SOLD, PRE, MOT, REFUABLE, THANK, YOU]

(b) OCR output list

FIGURE 6.6: A visualisation of (a) the receipt GT feature list and (b) its corresponding OCR output list.

Evaluate OCR results

The evaluation of the OCR engine is performed in Module 5.0. The module takes the GT feature lists, the OCR output lists, and the engineered evaluation metric as input. The evaluation metric, discussed above, is utilised in order to compute an accuracy score for each receipt document image by taking the intersect of the corresponding GT feature list and the OCR output list. These OCR accuracies are then stored in the data store for use cases in the following subcomponents. Finally, an average base OCR accuracy is computed by taking the average of the base OCR accuracies achieved by all the receipt document images in the ICDAR 2019 SROIE data set.

6.2.2 Implementation of the Labelling subcomponent

The second InDIE framework subcomponent implemented is the Labelling subcomponent, comprising the three modules elaborated upon in §5.3.2. The subcomponent receives an array of selected enhancement techniques and corresponding hyperparameters from the user. Additionally, the subcomponent takes the stored receipt document images, the GT feature lists, and the base OCR accuracies for each receipt document image obtained in the previous subcomponent as input.

Select enhancement techniques

Module 6.0 guides the user towards selecting which set of document image enhancement techniques is appropriate for the data set at hand. The process is commenced by, once again, utilising the experimental sample sourced in the previous subcomponent. The experimental sample is, however, expanded by adding a few receipt document images which achieved a high OCR accuracy and a few which obtained a low OCR accuracy. The expansion of the experimental sample increases the probability of the user to identify the effects of the document image enhancement techniques, and whether it improves upon the base OCR accuracy, or deteriorates it. As discussed in §4.1, document image enhancement techniques do not always improve the image quality, accordingly, it is just as important to observe the effects of the transformations on receipt document images that are considered to be of high quality, than it is to observe the

effects on receipt document images considered to be of low quality. Consequently, ten receipt document images that obtained a base OCR accuracy of at least 0.90, and ten receipt document images that obtained a base OCR accuracy of at most 0.50, are added to the experimental sample.

The first document image enhancement technique considered is cropping (briefly discussed in §4.2.3). The intuition is that cropping out unnecessary white spaces ought to reduce the complexity of the image, thereby aiding the performance of the OCR engine. By visual inspection of the ICDAR 2019 SROIE experimental sample data set, it is identified that there are several receipt document images that may benefit from undergoing cropping. Consider the receipt document image visualised in Figure 6.7, where Figure 6.7(a) shows the original receipt document image with excess white spacing (black border shows original page size to visualise page size), and Figure 6.7(b) shows the cropped receipt document image with the white space removed (black border shows new page size). Before implementing the document image enhancement technique, the OCR engine achieved an OCR accuracy of 0.85, while after being cropped it improved to an OCR accuracy of 1.00. This equates to a 0.15 improvement in OCR performance. Since there are many receipt document images showcasing a similar composition as the one visualised in Figure 6.7, it is decided to include cropping as a possible document image enhancement technique for the ICDAR 2019 SROIE data set.

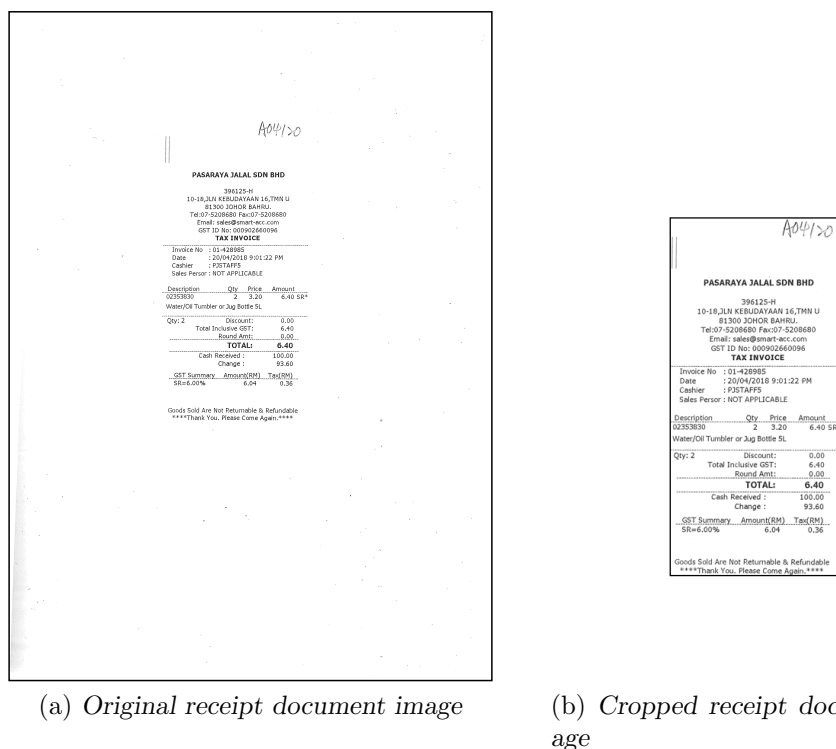


FIGURE 6.7: A visualisation of (a) an original receipt document image and (b) the receipt document image after being cropped.

The second document image enhancement technique considered is line removal (briefly discussed in §4.3.1). Printed lines for tables and/or structural reasons that are printed too close to characters tend to confuse OCR engines. Therefore, it is usually beneficial to remove these lines. By visual inspection of the ICDAR 2019 SROIE experimental sample data set, it is identified that there are limited lines printed on the receipt document images. This is as a result of the document type, as most receipts do not require many lines to structure the information

for reading purposes. Although lines are present on some receipt document images, they rarely protrude near important text. Consequently, it is decided not to include line removal as a possible document image enhancement technique for the ICDAR 2019 SROIE data set.

The third document image enhancement technique considered is document image binarisation (briefly discussed in §4.3.2). The intuition is that binarisation can remove shadows and overlapping background shapes from the surfaces of the receipt document images, reducing the complexity of the images. This can in turn (potentially) aid the performance of the OCR engine. By visual inspection of the ICDAR 2019 SROIE experimental sample data set, it is identified that there are several receipt document images that include shadows and background shapes, indicating that it may be beneficial to implement binarisation on some receipt document images. Consider the receipt document image visualised in Figure 6.8, where Figure 6.8(a) shows the original receipt document image comprising several dark shaded zones, and Figure 6.8(b) shows the receipt document image after global thresholding binarisation, without the presence of shadows. Before implementing the document image enhancement technique, the OCR engine achieved an OCR accuracy of 0.86, while it improved to an OCR accuracy of 1.00 after the implementation of binarisation. This equates to a 0.14 improvement in OCR performance. Accordingly, it is decided to include document image binarisation as a possible document image enhancement technique for the ICDAR 2019 SROIE data set.

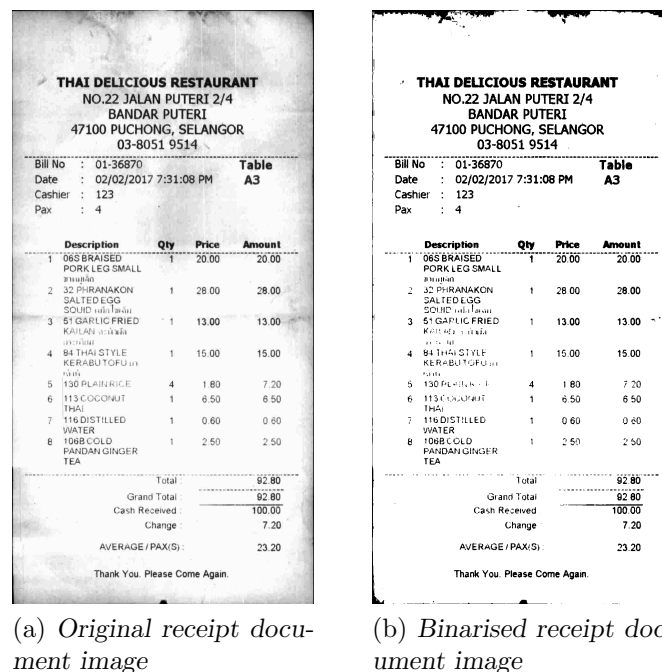


FIGURE 6.8: A visualisation of (a) an original receipt document image and (b) the receipt document image after being binarised.

The fourth document image enhancement technique considered is noise removal (briefly discussed in §4.3.3). Reducing noise on document images is a staple and widely utilised document image enhancement technique. Since the ICDAR 2019 SROIE data set is mostly scanned in with flatbed scanners, it is expected that noise will be present. By visual inspection of the ICDAR 2019 SROIE experimental sample data set, it is identified that there are several receipt document images containing noise, alluding to the high probability that it might be beneficial to include noise removal within the enhancement procedures. Consider the receipt document

image screenshot visualised in Figure 6.9 (enlarged for viewing purposes), where Figure 6.9(a) shows the original noisy receipt document image, and Figure 6.9(b) shows the receipt document image after a median filter noise removal technique is applied. Before the implementation of the document image enhancement technique, the OCR engine achieved an OCR accuracy of 0.69, while it improved to an OCR accuracy of 0.88 after the implementation of the median filter noise removal technique. This computes to a 0.19 improvement in OCR performance. Consequently, it is decided to include noise removal as a possible document image enhancement technique for the ICDAR 2019 SROIE data set.

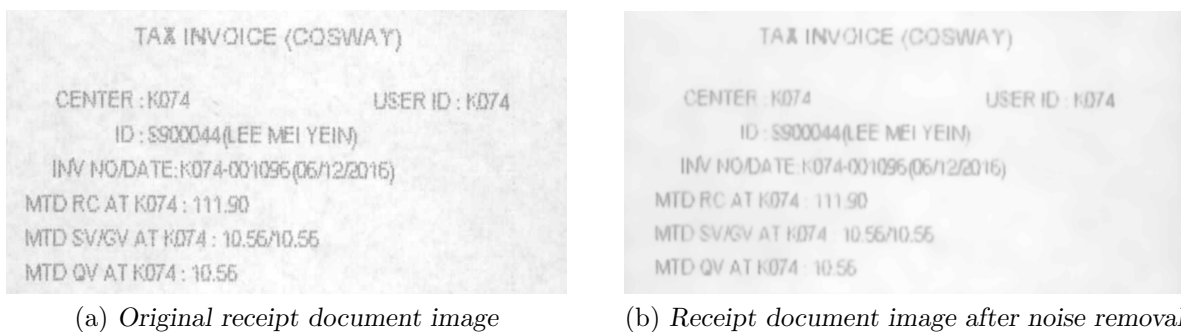


FIGURE 6.9: A visualisation of (a) an original receipt document image and (b) the receipt document image after noise removal.

The fifth and final document image enhancement technique considered is image sharpening (briefly discussed in §4.3.4). Most document image capturing equipment result in some image blur, therefore, image sharpening is used to increase the image quality by highlighting the edges and finer detail. By visual inspection of the ICDAR 2019 SROIE experimental sample data set, it is observed that there are several receipt document images that may benefit from the implementation of an image sharpening filter. Consider the receipt document image screenshot visualised in Figure 6.10 (enlarged for viewing purposes), where Figure 6.10(a) shows the original receipt document image, and Figure 6.10(b) shows the sharpened receipt document image. The filter used to convolve the document image is

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

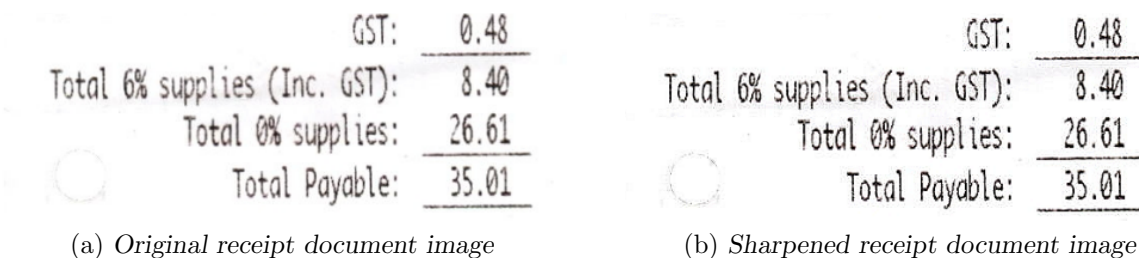
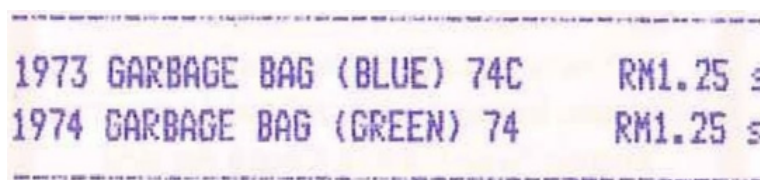


FIGURE 6.10: A visualisation of (a) an original receipt document image and (b) the receipt document image after being sharpened.

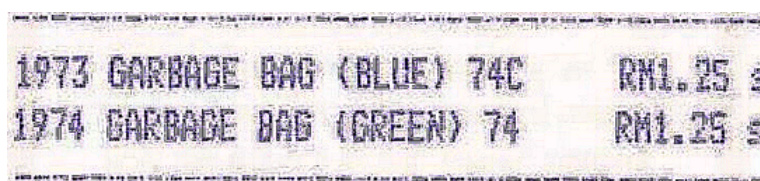
By means of visual inspection, it might not seem like the implementation of the image sharpening technique makes a notable difference, however, OCR engines are notably sensitive, even to the

smallest improvement in image quality. Before the implementation of image sharpening, the OCR engine achieved an OCR accuracy of 0.69, while it improved to an OCR accuracy of 0.88 after the implementation of the sharpening filter. This computes to a 0.19 improvement in OCR performance.

As discussed earlier, though, enhancement techniques can also deteriorate the document images. It is therefore also important to explore the examples where enhancement techniques did not result in improved OCR performance. In order to visualise the degradation that image sharpening can potentially introduce to an image, consider the screenshot visualised in Figure 6.11 (enlarged for viewing purposes), where the original receipt document image is shown in Figure 6.11(a) and the corresponding sharpened screenshot of the receipt document image is shown in Figure 6.11(b). Notice how the image sharpening technique amplified the imperfections around the printed characters, thereby distorting the character quality. Before implementing image sharpening, the OCR engine achieved an OCR accuracy of 0.71, however, it deteriorated to an OCR accuracy of 0.41 after the implementation. This results to a reduction in OCR accuracy of 0.30.



(a) Original receipt document image



(b) Over-sharpened receipt document image

FIGURE 6.11: A visualisation of (a) an original receipt document image and (b) the receipt document image after being sharpened.

The intuition is that the processes implemented in the later modules of the framework might be able to discern between the receipt document images that showcase OCR accuracy improvement from the ones that showcase deterioration. After exploring the results obtained by implementing image sharpening on selected images from the experimental sample data set, it is decided to include image sharpening as a possible document image enhancement technique for the ICDAR 2019 SROIE data set.

Create enhancement procedure categories

Module 7.0 facilitates the creation of the enhancement procedure categories which forms part of the list of possible receipt document image labels for the ICDAR 2019 SROIE data set. The enhancement procedure categories may comprise a single enhancement technique, a combination of enhancement techniques, or no enhancement techniques at all.

As mentioned in §5.3.2, a category comprising no enhancement techniques must always be included in the list of categories, ensuring that if all other enhancement procedures will result in a deterioration in OCR performance, that the document image ought to remain unaltered.

Accordingly, this is the first enhancement procedure to be included, called *Base image*. In Module 6.0, it was determined that cropping, binarisation, noise removal, and image sharpening are included in the list of possible document enhancement techniques. Implementing cropping showcased impressive results, as seen in the example visualised in Figure 6.7. It is also noted that there are several receipt document images with a similar structure as the visualised example. Therefore, the sole implementation of the cropping document enhancement technique is the second enhancement procedure, called *Cropping*. The implementation of a binarisation technique attained satisfactory results, as seen in the example visualised in Figure 6.8, for which an improvement of 0.14 was achieved. It is, however, important to note that the implementation of binarisation can also be markedly detrimental to some document images. This is an important caveat to be considered in the following modules. Nonetheless, the sole implementation of the binarisation document enhancement technique is selected as the third enhancement procedure, called *Binarisation*. Implementing noise reduction is identified as an effective document image enhancement technique for flatbed scanned document images, improving the OCR performance of the receipt document image, visualised in Figure 6.9, by 0.19. The sole implementation of the noise removal document enhancement technique is therefore selected as the fourth enhancement procedure, called *Noise removal*. The image sharpening filter showcased impressive results when an OCR improvement of 0.38 was attained on the receipt document image visualised in Figure 6.10. Although difficult for the human eye to see, the small improvements in image quality have a marketable influence on the performance of an OCR engine. Consequently, the sole implementation of the image sharpening filter is selected as the fifth enhancement procedure, called *Sharpening*.

As mentioned earlier, enhancement procedures may also include the sequential implementation of multiple enhancement techniques. It is deemed appropriate to create three additional enhancement categories, comprising the implementation of two sequential enhancement techniques. Consequently, the three additional enhancement techniques are as follows:

1. The implementation of cropping followed by noise removal, called *Cropping and noise removal*,
2. the implementation of cropping followed by image sharpening, called *Cropping and sharpening*, and
3. the implementation of binarisation followed by noise removal, called *Binarisation and noise removal*.

Therefore, eight enhancement procedures are engineered, one for implementing no enhancement technique, four for implementing a single enhancement technique, and three for implementing two sequential enhancement techniques. Consequently, each receipt document image will be assigned one of the eight engineered enhancement procedure categories as a label.

Assign enhancement procedure categories to images

The eighth module, and also the final module within the Labelling subcomponent, functions as a data labelling pipeline by assigning one of the eight previously engineered enhancement procedure categories to each receipt document image in the ICDAR 2019 SROIE data set. The module requires the list of enhancement procedure categories, the receipt document images, the GT feature lists, the selected OCR engine and its estimated hyperparameters, and the base OCR results, attained in Module 5.0, for each receipt document image.

The following procedure is repeated for each individual receipt document image: First, an enhancement procedure is applied to a receipt document image. Thereafter, the OCR recognition and output formatting steps of Module 3.0 is implemented, once again, whereby the EasyOCR engine, in combination with its previously estimated hyperparameters, is applied to the receipt document image. This is followed by the output formatting steps in order to produce the OCR output list. The OCR evaluation steps of Module 5.0 is then invoked, producing an OCR accuracy score for the receipt document image after being subjected to the specific enhancement procedure. These steps are repeated for each of the seven new enhancement procedures (the first category requires no enhancement; therefore, it is assigned the same OCR accuracy as the base OCR accuracy attained earlier), outputting eight OCR accuracy scores for the receipt document image, each corresponding to an enhancement procedure. The enhancement procedure which reflects the best OCR accuracy is then deemed the best category for the specific receipt document image and is therefore assigned its corresponding label.

6.2.3 Implementation of the Modelling subcomponent

The third subcomponent of the InDIE framework to be implemented is the Modelling subcomponent, comprising the three modules elaborated upon in §5.3.3. The subcomponent receives the selected model components with their corresponding estimated hyperparameters, a training set, a validation set, a testing set, and the assigned labels.

Select convolutional base

The convolutional base of the model, *i.e.* the first part of the model, ought to be utilised to extract feature maps from the receipt document images, usually not interpretable by humans. The three pre-trained models considered are the AlexNet [162], VGG-16 [262], and EfficientNet [278] architectures.

Since there is limited research on the capabilities of pre-trained models to extract knowledge from document images for document image enhancement purposes, the research performed on *document image classification* is used as a source to draw inspiration from. Document image classification encapsulates the task of classifying document images based on text contents and/or structural properties. According to Das *et al.* [60], a document structure can be realised through the use of a pattern classification system, *e.g.* deep learning techniques. The *Ryerson Vision Lab Complex Document Information Processing* (RVL-CDIP) data set [113] comprises scanned document images belonging to 16 classes. These classes include letters, forms, resumes, and memos. Numerous approaches and algorithms have been developed in pursuit of automating this complex task, including several transfer learning approaches, obtaining state-of-the-art results, as graphically illustrated in Figure 6.12.

In 2017, Afzal *et al.* [2] implemented a transfer learning approach, exploring the capabilities of several pre-trained CNN architectures, and found that the VGG-16 pre-trained model performed the best, obtaining an accuracy of 0.910 on the RVL-CDIP data set. Moreover, one year later, Das *et al.* [60] improved upon the work of Afzal *et al.* using the VGG-16 pre-trained model which achieved an accuracy of 0.922. The state-of-the-art results achieved by the application of the VGG-16 pre-trained model suggests the model's capabilities to transfer its learning experience from being trained on ImageNet to the domain of document image classification. Consequently, the intuition is that the feature maps extracted from the document images for the document image classification task might also be used to predict which enhancement procedure ought to

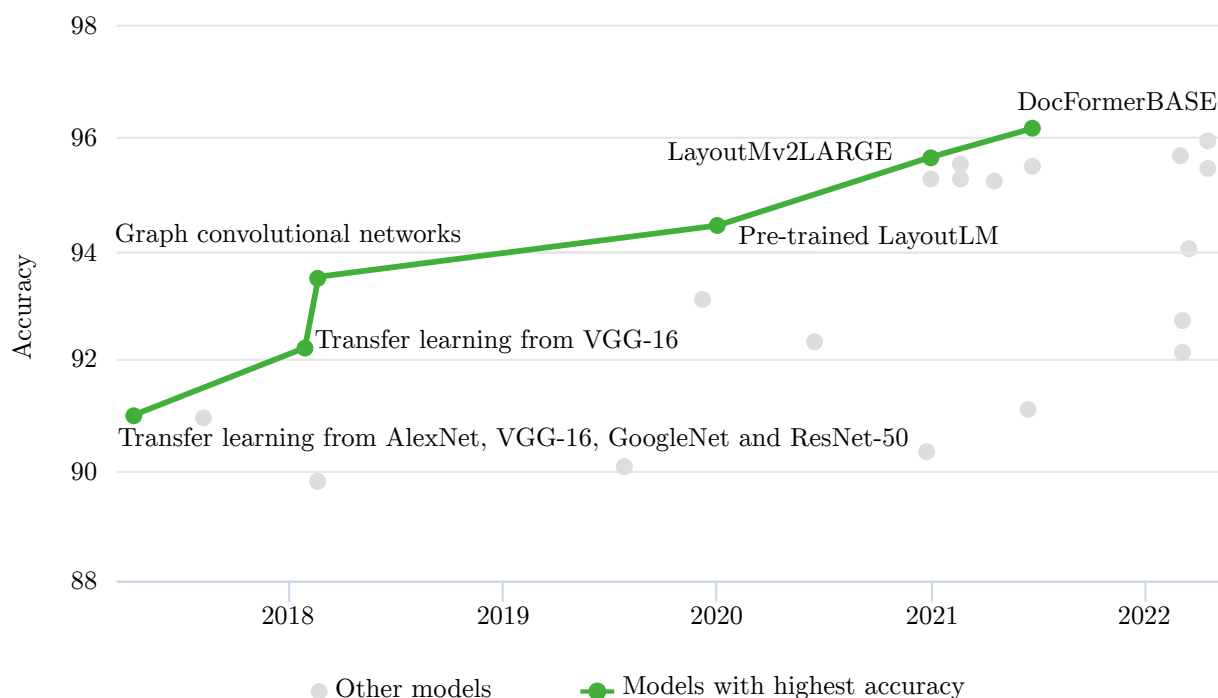


FIGURE 6.12: Graphical illustration of the top performing methods for document image classification on the RVL-CDIP data set [211].

be used on each of the ICDAR 2019 SROIE data set receipt document images. Therefore, the pre-trained VGG-16 model is selected as the convolutional base for the prediction model.

Build and train model

The building and training of the model is performed in three stages, namely: (1) The data preparation stage, where the data are transformed and parsed in an appropriate data sets for model training, (2) the model construction stage, where the different layers of the model is compiled together, and (3) the training stage, where the model is fitted to the prepared data sets.

The iterative nature of building a model to predict the enhancement procedures for ICDAR 2019 SROIE data set is initiated by splitting the data set into a training, validation, and testing set, based on a 60%, 20%, and 20% split, respectively. Thereafter, the training set and validation minority classes are oversampled in order to balance the data set. The receipt document images in all three data sets are resized to a 224×224 dimension with three colour channels, as recommended by Das *et al.* [60], since the VGG-16 model was initially trained on $224 \times 224 \times 3$ images. Thereafter, the training set and validation set data are augmented to increase the number of data instances by adding slightly modified copies of the original receipt document images. The augmentations include initial estimated parameters of rotations, namely 20° , shifting the width with a range of 0.2, shifting the height with a range of 0.2, adding a zoom range of 0.3, and permitting horizontal and vertical flip. The test set data are not augmented. Thereafter, the training set, validation set, and testing set data image pixel values are normalised by dividing each pixel by 255.

Next, the model is constructed, incorporating the convolutional base and a chosen classifier. The VGG-16 feature extraction layers are added as the convolutional base, requiring the removal of

the original VGG-16 classifier. The convolutional base layers may be set to frozen and/or unfrozen, depending on the extent of training desired. A global average pooling layer is added to further reduce the data, followed by a flatten layer. According to the results obtained by Das *et al.* [60], a classifier comprising multiple dense layers will result in the best accuracy when utilising the VGG-16 as a convolutional base. Accordingly, a classifier is constructed comprising three dense layers, the first of which has a starting number of 250 neurons, the second 50 neurons, and the third and final classification layer contains the number of neurons equal to the number of classes to be predicted. The first and second dense layers employ the ReLU activation function, while the third (*i.e.* final) classification layer employs a softmax activation function, as recommended by the literature (listed in Table 2.2). It is regarded as good practice to include dropout layers in order to attempt to reduce overfitting, as discussed in §2.5.1. A dropout layer is added between the first and second dense layers, and between the second and third dense layers, with initial dropout rates of 0.3 and 0.2, respectively. Afzal *et al.* [2] recommend a learning rate of between 0.01 and 0.0001. The Adam optimiser is selected with an estimated learning rate of 0.0001 for the initial training instance. Since the aim of the model is to predict categories, the categorical cross-entropy loss function is employed for model training. The chosen metric for model evaluation is AUC.

Finally, the model may be fitted to the training data set and evaluated in respect of the validation data set. Afzal *et al.* [2] recommend the number of training epochs to range between 40 and 80 epochs. Accordingly, the model is initially trained for 50 epochs.

Since the initial hyperparameters are merely estimations gathered from previous works in the domain of transfer learning for document images, several iterations of fine tuning are performed. During training, the training and validation curves are visualised and presented to the user. The validation curve is used to validate the training of the model and measure the performance of the model. The insight gained by inspecting the validation curve (*e.g.* whether the model is under- or overfitting or the possible need for additional epochs) is then used as a guide to fine-tune the hyperparameters of the model by means of a sensitivity analysis.

Upon further analysis of the various training and validation curves produced by the fine tuning stage of the training procedure, it was found that the task of accurately predicting which of the eight possible enhancement procedures would produce the best OCR performance for each receipt document images is deemed too complex for the model at hand. Accordingly, the problem is simplified by reducing the complex multiclass classification problem into a simpler binary classification task by only considering whether (or not) to implement a specific enhancement procedure. In order to select which enhancement procedure to select for the binary classification task, the results obtained after Module 8.0 is inspected. The enhancement procedure that, if correctly predicted and therefore implemented, can potentially result in the best overall OCR improvement is selected. In the case of the eight enhancement procedures created for the ICDAR 2019 SROIE data set, image sharpening achieved the best average OCR accuracy (if correctly predicted and implemented). Accordingly, Modules 8.0–10.0 are repeated, this time as a binary classification problem.

Evaluate model

In order to fairly evaluate the performance of the trained model, the model is tested in respect of the testing set (*i.e.* receipt document images comprising 20% of the overall ICDAR 2019 SROIE data set, unseen by the trained model) in the eleventh module of the InDIE framework.

6.2.4 Implementation of the Analysis subcomponent

The fourth and final InDIE framework subcomponent to be implemented is the Analysis subcomponent, comprising the three modules, *i.e.* Modules 12.0–14.0, as elaborated upon in §5.3.4. The subcomponent receives the test set receipt document images created in Module 9.0, the corresponding predicted classifications provided as output from Module 11.0, the corresponding GT feature lists produced by Module 2.0, and the OCR engine with its accompanying estimated hyperparameters as input. The implementation of Modules 12.0–14.0 is only briefly discussed, as the results obtained throughout the proof-of-concept implementation is elaborated upon later in this chapter.

Implement enhancement procedure predictions on test set images

Module 12.0 facilitates the implementation of the predicted classifications on the test set receipt document images. The test set receipt document images that received a prediction for Base image³ are left as is, whereas the remainder are transformed (and hopefully enhanced) with the image sharpening filter technique discussed earlier. The new and transformed test set is passed on as input to Module 13.0.

Implement OCR on enhanced test set

Module 13.0 receives the transformed and adapted test set receipt document images, and the OCR engine with its accompanying hyperparameters as input. OCR is implemented on the test set, and the OCR output is cleaned in a similar manner, as explained in the implementation of Module 3.0. The produced output is the new OCR output lists corresponding to the test set receipt document images.

Evaluate OCR results

In Module 14.0, the final module of the InDIE framework, the newly attained OCR output lists are used, in combination with the GT feature lists and the evaluation metric to compute the enhanced OCR accuracy of each receipt document image in the test set, whereafter the average enhanced OCR accuracy for the enhanced test set is obtained. The average enhanced OCR accuracy is then compared with the original test set average base OCR accuracy (*i.e.* the test set before any transformation) in order to determine whether the employment of the trained model improved the average OCR accuracy of the test set.

6.3 Results produced by InDIE framework implementation

A plethora of results were produced during the proof-of-concept implementation of the InDIE framework applied to the ICDAR 2019 SROIE data set. The insights gained through the analysis of these results were employed in order to guide the user towards making important design decisions, influencing the quality and accuracy of the trained model, and ultimately, the value that was added by the implementation of the InDIE framework. The remainder of this chapter is devoted to the visualisation, exploration, and in-depth discussions of the results obtained

³Enhancement procedure category indicating that document image ought to remain unaltered.

throughout the implementation of the four InDIE subcomponents on the ICDAR 2019 SROIE data set.

Initial base OCR results

Through the implementation of Module 5.0, *i.e.* the evaluation of the base OCR results, a base OCR accuracy was assigned to each receipt document image, according to the evaluation metric engineered in Module 4.0. As discussed earlier, the ICDAR 2019 SROIE data set is a well-known data set in the OCR document analysis academic committee. Accordingly, from the conception of the data set in 2019, several benchmark results have been attained which may be utilised in order to verify whether the implementation of the Configure subcomponent resulted in the expected base OCR performance.

In 2021, Correa *et al.* [54] released a comparative study analysing the base OCR performances of several OCR engines on the ICDAR 2019 SROIE data set. The OCR engines included Calamari, Kraken, OCRopus (employs an RNN model), Tesseract version 3, Tesseract version 4.1 (employing an LSTM model), Tesseract version 5 (employing an LSTM model), and Google Vision [24] (a proprietary and paid service). Although EasyOCR was not included due to its recent release in 2020, the results obtained by the Tesseract OCR engines may be used for comparison purposes as the Tesseract OCR engines and the EasyOCR engine are both open-source models, while Google Vision is a paid service. A summary of the average base OCR accuracy results obtained by Correa *et al.* and the EasyOCR results obtained by the InDIE framework implementation are provided in Table 6.1. The InDIE EasyOCR implementation achieved an average base OCR accuracy of 0.7695, while the Tesseract version 4.1 (released in 2019 and which employs an LSTM model) of Correa *et al.* achieved an average base OCR accuracy of 0.7842. It is important to note that the goal of this comparison is not to achieve a higher base OCR accuracy, but rather to showcase that the InDIE EasyOCR implementation was conducted in a manner (*e.g.* data cleaning, formatting, and estimations of hyperparameters) that produces expected and comparable results, verifiable by other works in the literature. The similar results showcase that the implementation of the EasyOCR engine is *on par* with other open-source OCR engine counterparts. Therefore, it is verified that the implementation of the first subcomponent of the InDIE framework achieved the desired outcome.

TABLE 6.1: Summary of average base OCR accuracy (word-level) results obtained by Correa *et al.* [54], and the EasyOCR results obtained by the InDIE framework implementation.

OCR Engines	Average base OCR accuracy
Calamari	0.5631
Kraken	0.3656
Tesseract version 3	0.1149
Tesseract version 4.1	0.7842
Tesseract version 5	0.7973
Google Vision	0.8608
InDIE: EasyOCR	0.7695

After verifying that the EasyOCR engine produced the expected results, valuable insights may be attained by visualising and analysing the base OCR accuracies assigned to the receipt document images. The chart visualised in Figure 6.13 displays the number of receipt document images that achieved a base OCR accuracy with respect to each of the ten accuracy bins. The lower base OCR accuracy bins are showcased in red, while the higher base OCR accuracy bins are

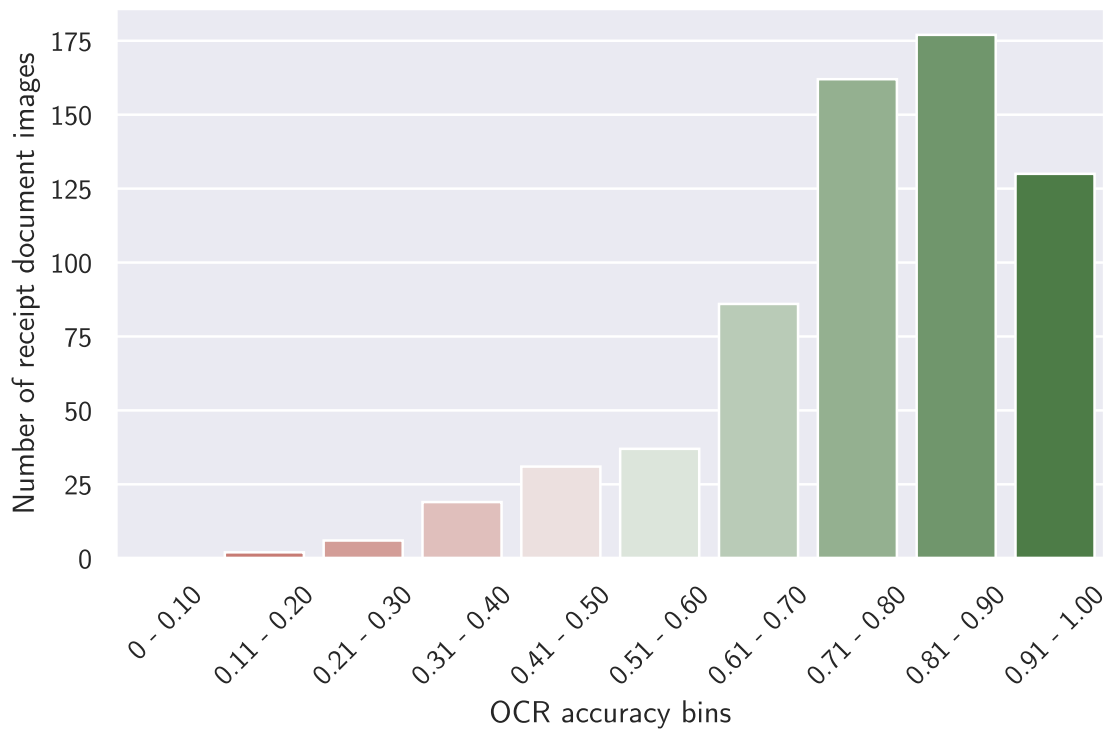


FIGURE 6.13: Graphical illustration of the base OCR accuracy results for the ICDAR 2019 SROIE data set.

showcased in green. The base OCR accuracy bin comprising the most receipt document images is the “0.81–0.90” bin, followed closely by the “0.71–0.80” bin. This reflects the average base OCR accuracy of 0.7695.

Enhancement procedure results

After the construction of the eight original enhancement procedure categories in Module 7.0, the receipt document images were labelled in Module 8.0 according to the enhancement procedure that resulted in the best OCR performance. The original results (before the binary relabelling) of the assigned labels are shown in Table 6.2, where the first column indicates the category name and the second column indicates the portion of the receipt document images assigned to the corresponding label (*i.e.* assigned if the enhancement procedure obtained the best OCR results).

It is found that 0.43 of the receipt document images were assigned the Base image category. This means that 0.43 of the receipt document images did not improve by any of the document enhancement techniques, and were therefore assigned the Base image category. The second most assigned enhancement procedure category was Sharpening, with 0.20 of the receipt document images experiencing the most pronounced OCR accuracy improvement after the implementation of the image sharpening filter. Although each category was assigned to several receipt document images, it was found that the enhancement procedure categories of Noise removal, Cropping & noise removal, Cropping & sharpening, and Binarisation & noise removal were all assigned to only a portion of 0.07, 0.04, 0.03, and 0.02 of the receipt document images, respectively.

Another factor to consider is the portion of images which would be improved upon (in terms of OCR accuracy) if each enhancement procedure was individually applied to the entire data set without any intelligence. The original results (before the binary relabelling) of the individual

TABLE 6.2: Summary of the original labelling distribution when considering all eight enhancement procedure categories.

Enhancement procedure category	Portion of image category assignment
Base image	0.43
Cropping	0.11
Binarisation	0.10
Noise removal	0.07
Sharpening	0.20
Cropping & noise removal	0.04
Cropping & sharpening	0.03
Binarisation & noise removal	0.02

application of enhancement procedures are shown in Table 6.3 according to which the first column indicates the category name that was individually implemented and the second column indicates the portion of the receipt document images that exhibited improved OCR accuracy. When inspecting the second column results, it was found that Sharpening improved the largest proportion of receipt document images (when applied to all), with 0.30 of the data set showcasing an improved OCR accuracy (*i.e.* in comparison with their corresponding base OCR accuracies).

TABLE 6.3: Summary of the portion of images which showcased an improved OCR accuracy after the individual application of enhancement procedures.

Single enhancement procedure category	Portion of entire data set which showed OCR improvement
Cropping	0.16
Binarisation	0.17
Noise removal	0.19
Sharpening	0.30
Cropping & noise removal	0.23
Cropping & sharpening	0.27
Binarisation & noise removal	0.16

The ICDAR 2019 SROIE data set is considered to be a markedly small data set. Accordingly, the portion according to which some of these enhancement procedures were assigned, were deemed too small, as it is hypothesised that the model would not be able to learn from the intrinsic patterns between all eight categories. Consequently, it was decided to remove all enhancement procedure categories assigned to less than 0.10 of the receipt document images. The few corresponding receipt document images were relabelled with the remaining enhancement procedure categories.

It was found that if the assigned labels are perfectly implemented (*i.e.* if each receipt document image was subjected to the enhancement procedure that would result in the best possible OCR improvement), the average best OCR accuracy for the ICDAR 2019 SROIE data set would be 0.8231. Therefore, if the transfer learning model of the Modelling subcomponent possesses the capability of predicting all the assigned labels exactly, followed by the implementation of the predicted enhancement procedures, the average OCR accuracy would be improved from 0.7695 (the average base OCR accuracy) to 0.8231 (the average best OCR accuracy), constituting an improvement of 0.054. This emphasises an important facet of OCR-related tasks. It is important

to note that even the smallest OCR improvement can have a significant impact on the real-world business case. Consider the example where the text of one million receipts (comprising an average of 15 GT feature strings each) are to be correctly recognised. An improvement of only 0.054 would result in an increase of correctly recognised GT strings by an additional 810 000 strings, a substantial rise in potential business value.

Graphically illustrated in Figure 6.14 is a chart showcasing both the computed base OCR accuracy distribution and the theoretical best OCR accuracy distribution, in terms of the ten accuracy bins. The results of the base receipt document images are shown in red, and the theoretical best receipt document images in green. Note the extent to which the red bars surpass the green bars in the ranges “0–0.10” to “0.71–0.80”, and the green bars surpass the red bars in the top performing ranges “0.81–0.91” and “0.91–1.00”. This is an indication that the correct implementation of the assigned enhancement procedures can result in a reduction of low OCR accuracy receipt document images and an increase in high OCR accuracy receipt document images.

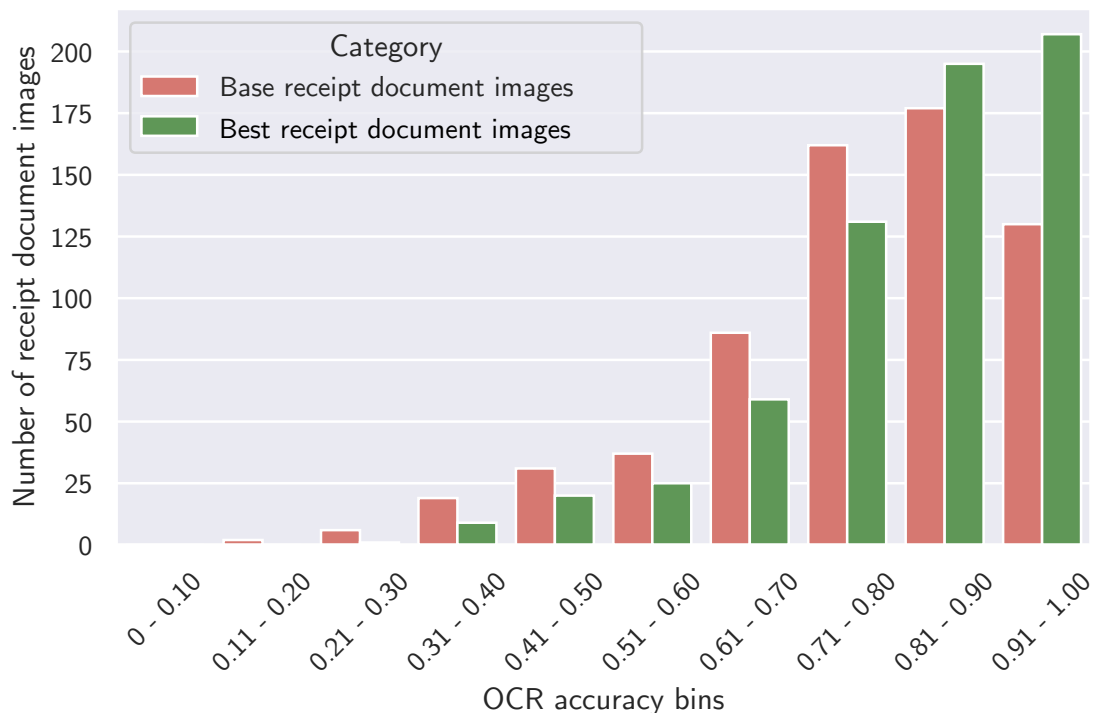


FIGURE 6.14: Graphical comparison of the number of receipt document images from the base OCR accuracy scenario and the enhanced OCR accuracy scenario, in terms of the ten accuracy bins.

Additional results were examined, specifically focussing on the computed OCR accuracies. though the assignment of the enhancement procedure categories, where each enhancement procedure was implemented on all the receipt document images in order to provide insights on the impact of solely implementing an enhancement procedure without any intelligence *versus* correctly predicting whether to implement the procedure or not. These results are presented in Table 6.4, where the first column indicates the enhancement procedure category and the second column indicates the average OCR accuracy if the corresponding enhancement procedure was applied to the entire receipt document images without any intelligence. It is found that none of the full implementations of any of the enhancement procedures could improve upon the average base OCR accuracy of 0.7695, with Sharpening achieving the closest results of 0.7481. The

TABLE 6.4: Summary of the OCR results obtained when applying individual enhancement procedure categories on the entire receipt data set.

Enhancement procedure category	Average full OCR accuracy
Base image	0.7695
Cropping	0.7399
Binarisation	0.6881
Noise removal	0.7476
Sharpening	0.7481
Cropping & noise removal	0.7400
Cropping & sharpening	0.7400
Binarisation & noise removal	0.6822

displayed results, once again, showcase the damage done by document enhancement techniques if not applied to the appropriate receipt document images.

Moreover, it was also examined which individual enhancement procedure would produce the best average OCR accuracy if it is assumed that a binary transfer learning model can precisely predict whether a specific individual enhancement procedure ought to be applied or not — *i.e.* only applied to the receipt document images if it would produce an improved OCR accuracy. These results are shown in Table 6.5. Since it is assumed that the theoretical transfer learning model can predict perfectly, it is expected that all the obtained average OCR accuracies ought to be an improvement on the average base OCR accuracy. The individual application of Sharpening, once again, obtained the best results with an average OCR accuracy of 0.8002, showing a maximum possible improvement of 0.0307 for a single enhancement procedure. When considering the example of the one million receipt document images, this would constitute an increase in the number of correctly recognised GT strings by 460 500 strings, still a considerable gain in potential business value.

TABLE 6.5: Summary of the average best OCR results obtained when applying individual enhancement procedure categories only on receipts which would showcase an improved OCR accuracy.

Single enhancement procedure category	Average Best OCR accuracy
Cropping	0.7823
Binarisation	0.7823
Noise removal	0.7838
Sharpening	0.8002
Cropping & noise removal	0.7880
Cropping & sharpening	0.7969
Binarisation & noise removal	0.7831

As mentioned in §6.2.3, the model training and validation curves showcased suboptimal training performance, and therefore a binary classifier was also considered in order to reduce the complexity of the task. After inspecting the results presented in Table 6.4 and Table 6.5, it was found that Sharpening showcased the best possible results when implementing a single enhancement procedure (other than the Base image enhancement procedure category). Consequently, the receipt document images were relabelled according to the binary classification of Base image (*i.e.* unaltered) or Sharpening.

Graphically illustrated in Figure 6.15 is a chart showcasing both the base OCR accuracy distribution and the theoretical best OCR accuracy distributions (for the individual application of the Sharpening enhancement procedure), in terms of the ten accuracy bins. The results of the base receipt document images are shown in red, and the Sharpened receipt document images (*i.e.* when only considering the Sharpening) in green. Once again, it may be noted how the red bars surpass the green bars in the ranges “0–0.10” to “0.71–0.80”, and the green bars surpass the red bars in the top performing ranges “0.81–0.91” and “0.91–1.00”, although the magnitude of the difference is slightly reduced when compared with the difference seen in Figure 6.14. This is another indication that the correct implementation of the Sharpening enhancement procedures can result in a reduction of low OCR accuracy receipt document images and an increase in high OCR accuracy receipt document images.

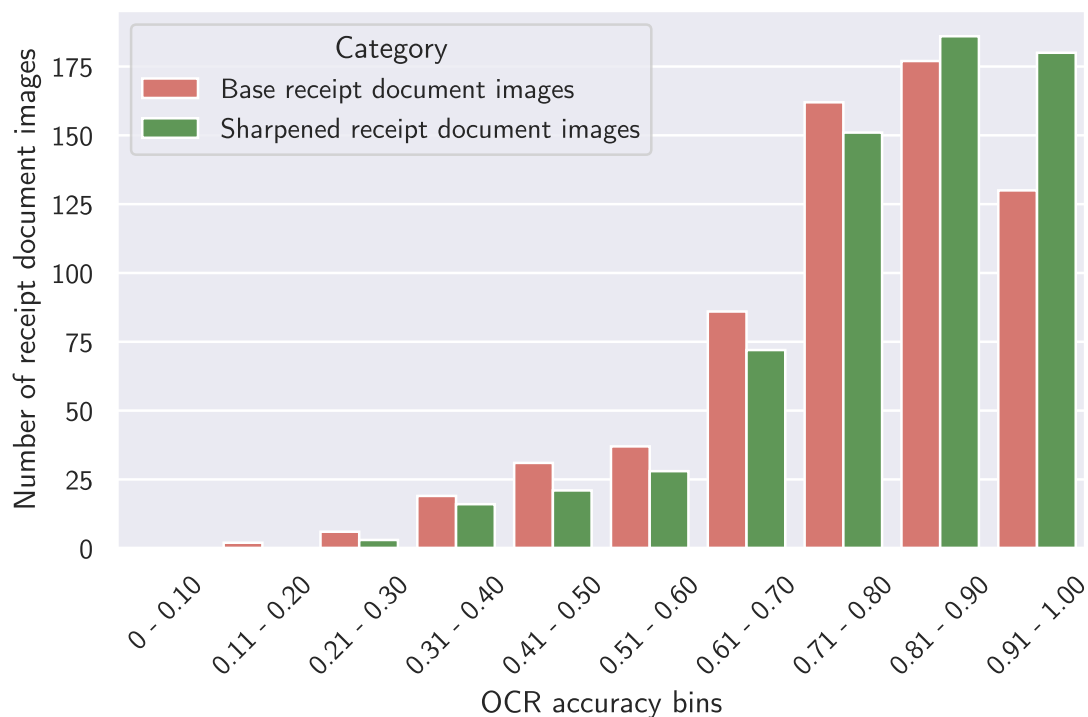


FIGURE 6.15: Graphical comparison of the number of receipt document images from the base OCR accuracy scenario and the enhanced (*i.e.* Base image or Sharpened) OCR accuracy scenario, in terms of the ten accuracy bins.

Model training and validation results

First, the model training and validation results when considering the five selected enhancement procedures are examined. After the assignment of the original five enhancement procedure categories in Module 8.0, the model was trained and fine-tuned several times. Recall that the first few training iterations were performed on the multiclass classification target feature, comprising Base image, Cropping, Binarisation, Noise removal, and Sharpening. The initial hyperparameters were estimated based on recommended document image classification hyperparameters identified in literature. After the implementation of a sensitivity analysis, whereby numerous rounds of fine-tuning were performed (guided by the literature reviewed in §2.1 to §2.6), it was found that the multiclass classification task struggled to learn from the provided data, producing validation AUC scores of less than 0.53. By inspecting the training and validation curves after

every training round, several attempts were made to increase the model's capability to learn more complex patterns, *e.g.* increasing the number of layers in the classifier, reducing dropout, and/or unfreezing several layers of the pre-trained VGG-16 convolutional base. Unfortunately, none of the aforementioned techniques introduced the model with the required capabilities to learn the complex patterns within the data. Consequently, the required complexity was reduced by relabelling the receipt document images according to a binary classification context where the image is either labelled as with the Base image category (*i.e.* no transformation) or the Sharpening category.

The training of the new task produced better results, showcasing that the model is indeed learning some intrinsic patterns embedded within the data. The training and validation curves of the first training iteration, *i.e.* before any fine tuning was performed, is illustrated in Figure 6.16, where the blue curve represents the training AUC curve, and the orange curve represents the validation AUC curve. For the first 30 epochs the training and validation curve continuously obtained an increasing AUC score as training continued. After 100 epochs, the training curve achieved a maximum AUC score of 0.71, however, when inspecting the validation curve, it was identified that the validation AUC curve reached a plateau around the 70 epoch mark, and slowly started to decrease until the last epoch. This was recognised as an indication that the model might be overfitting the training data.

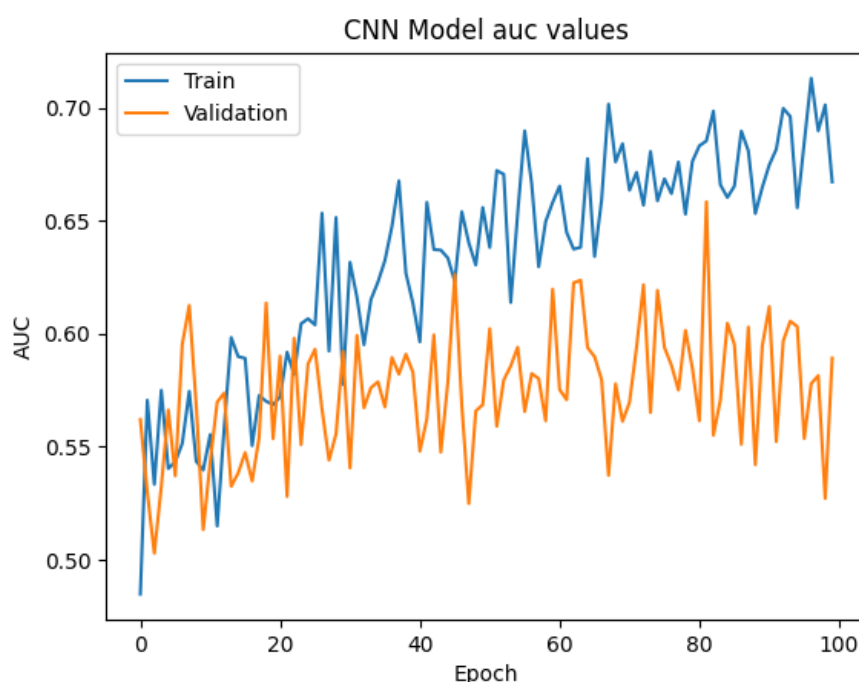


FIGURE 6.16: Visualisation of training (blue) and validation (orange) AUC curves for the training of the prediction model.

The possible occurrence of overfitting was verified by inspection of the loss curves visualised in Figure 6.17. For the first 60 epochs both the training and validation losses decreased as training continued, however, after approximately 70 epochs of training, the validation loss curve showcased a noteworthy spike which kept moving further away from the training loss curve, clearly indicating continuous overfitting. In order to attempt to reduce overfitting, the dropout values were increased during the fine tuning procedure.

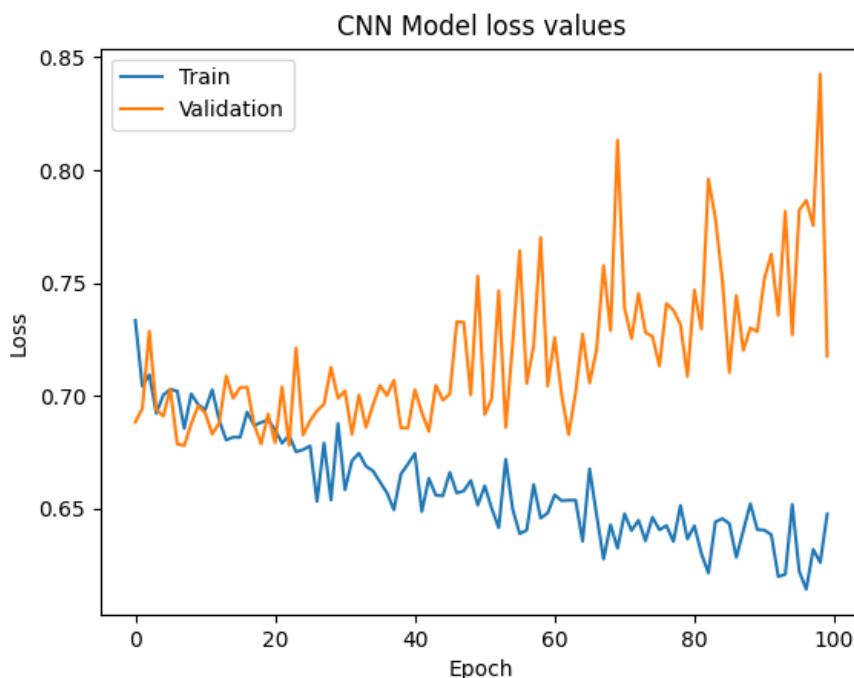


FIGURE 6.17: Visualisation of training (blue) and validation (orange) loss curves for the training of the prediction model.

After reducing the effects of overfitting, two layers of the pre-trained VGG-16 model were unfrozen, enabling the training of its weights. In order to train the unfrozen layers, the added classifier head was first trained for 50 epochs with the fully frozen pre-trained convolutional base. Thereafter, the last two layers of the convolutional base were unfrozen and trained in conjunction with the classifier for another 50 epochs. The training thereof resulted in significant overfitting. Accordingly, another dropout layer was added after the convolutional base with a dropout value of 0.5. The training and validation AUC curves are visualised in Figure 6.18 where the green line shows the start of the additional training of the two unfrozen convolutional base layers. There is a clear elevation in validation AUC score when the training on the two unfrozen layers commences, showcasing that an increase in AUC score can be obtained by unfreezing some of the convolutional base layers. The final AUC scores attained were 0.70 for the training curve, and 0.62 for the validation curve. Although these results are not particularly high AUC scores, it is found that the model is learning some information from the intrinsic patterns within the data, which is considered to be a valuable academic finding.

After the model was fine-tuned, it was evaluated on the test set. The test set confusion matrix is provided in Figure 6.19. It was found that 0.61 of the Base image category was predicted correctly, while only 0.57 of the Sharpening category was predicted correctly. It is important to remember that the Base image category ought to comprise all the receipt document images that would either deteriorate or remain in a neutral state if sharpened, while the Sharpening category ought to comprise all the document images that will improve if sharpened. Accordingly, when a receipt document image is incorrectly classified as Base image while its actual value is Sharpening, that no enhancement procedure will be implemented. Although the desired improvement will not be realised, no damage will be done by not enhancing the receipt document image. The receipt document image is merely left in its original state. In contrast, if the model classifies a receipt document image as Sharpening, while it has an actual value of Base image, this



FIGURE 6.18: Visualisation of training (blue) and validation (orange) AUC curves for the training of the prediction model where the initiation of unfreezing two convolutional base layers are indicated with the green vertical line.

may potentially deteriorate the receipt document image, as it might sharpen a receipt document image that will deteriorate when altered, reducing the average OCR accuracy. Therefore, the priority of the model ought to be to reduce false predictions of the Sharpening category (*i.e.* the false negatives). In the case of the test set, the attained false negative value is 0.39. It is recognised that the false negative value is still deemed as a relatively high value, however, it shows that the trained model has the capability to learn from the intrinsic patterns within the data, and to employ this attained knowledge to (slightly) reduce the false negative predictions.

		Actual	
		A	B
Predicted	A	0.61	0.43
	B	0.39	0.57

A: Base image
B: Sharpening

FIGURE 6.19: Test set confusion matrix results for the binary classification model.

Enhanced test set OCR results

The final results to be explored are obtained from the fourth and final subcomponent of the InDIE framework, *i.e.* the Analysis subcomponent, whereby the predicted classes were implemented on the receipt document images of the test set.

After implementing the model predictions, the average enhanced OCR accuracy (*i.e.* the test set OCR results which utilised the intelligent predictions) was computed and compared with the

corresponding test set's average base OCR accuracy (*i.e.* without any enhancement), the test set's average full Sharpening OCR accuracy (*i.e.* if Sharpening was applied to all the test set receipt document images), and the test set's average best OCR accuracy (*i.e.* if applied only to the receipt document images that would showcase an improved OCR accuracy). These results are shown in Table 6.6. The average base OCR accuracy for the test set is 0.7720. If one would implement the sharpening filter on all the test set receipt document images, the average OCR accuracy would fall to 0.7612, a decrease of 0.0108. If the sharpening filter is only implemented on the test set document images that would improve (*i.e.* the best possible case), the average OCR accuracy would be 0.8039. This means that if the transfer learning model classifies all the test set document images correctly, that the average OCR accuracy can be increased by 0.0319. Finally, after implementing the predictions of the trained model, an average enhanced OCR accuracy of 0.7827 was achieved, *i.e.* an increase of 0.0107. Considering the previously discussed business case of one million receipt document images with 15 GT feature strings each, this improvement would constitute an increase of 160 500 strings, still a noteworthy benefit to a real-world business case.

TABLE 6.6: Average OCR accuracy scenarios for different test set conditions.

Test set category	Average OCR accuracy
Average base OCR accuracy	0.7720
Average full Sharpened OCR accuracy	0.7612
Average best OCR accuracy	0.8039
Average enhanced OCR accuracy	0.7827

When comparing the newly attained average enhanced OCR accuracy (*i.e.* 0.7827) with the average full Sharpened OCR accuracy (*i.e.* 0.7612), an improvement of 0.0215 is achieved. In order to gain more insight into how the added machine intelligence aided in actualising this OCR improvement, it is important to analyse the number of receipt document images that either improved, remained unchanged, or deteriorated (in terms of OCR accuracy). Shown in Table 6.7 are detailed results of the aforementioned two categories in terms of their performance compared with the test set base OCR accuracies. If the Sharpening enhancement procedure is applied to all the test set receipt document images, a 0.30 portion improves in OCR accuracy while 0.38 of the images deteriorate in OCR accuracy, deriving an improved/deteriorated ratio of 0.7895. Since the ratio is considerably less than one, it can be inferred that the full implementation of the Sharpening enhancement procedure would deteriorate the average OCR accuracy. If the Sharpening enhancement procedure is intelligently predicted and implemented on the test set receipt document images (*i.e.* with the use of machine intelligence), a 0.18 portion improves in OCR accuracy while 0.16 of the images deteriorate in OCR accuracy, deriving an improved/deteriorated ratio of 1.1250. Since the ratio is greater than one, it is confirmed that the *intelligent* application of the Sharpening enhancement procedure derives an improvement of the average OCR accuracy.

In conclusion, it is found that *before* any form of machine intelligence was employed, the Sharpening enhancement procedure *deteriorated* the average OCR accuracy of the test set, with more receipt document images being damaged than enhanced. *After* the addition of machine intelligence, however, the implementation of the Sharpening enhancement procedure *improved* the average OCR accuracy of the test set, with more receipt document images being enhanced than damaged.

TABLE 6.7: *A comparison of the number of improved, unchanged, and deteriorated receipt document images when sharpening all the images and when only sharpening the predicted images.*

Impact category compared with the original base OCR	All images in test set		Only applied to predicted images in test set	
	Number	Ratio	Number	Ratio
Improved	38	0.30	22	0.18
Same	39	0.31	83	0.66
Deteriorated	48	0.38	20	0.16
Improved/deteriorated	0.7895		1.1250	

6.4 Chapter summary

In this chapter, a proof-of-concept implementation was executed by means of the InDIE framework. The chapter opened in §6.1 with the discussion of an appropriate benchmark data set for the proof-of-concept implementation, whereafter the ICDAR 2019 SROIE data set was selected and explored. This was followed by an in-depth discussion in §6.2 of the proof-of-consent implementation of the four InDIE subcomponents and all their underlying modules. The chapter was concluded in §6.3 with a thorough analysis of the results attained throughout the implementation of the proof-of-concept.

CHAPTER 7

Case study implementation

Contents

7.1 Data set background	129
7.2 Implementation of the InDIE framework	133
7.3 Results produced by InDIE framework implementation	139
7.4 Subject-matter expert validation	148
7.5 Chapter summary	150

In order to further illustrate the potential utility of the InDIE framework (introduced to the reader in Chapter 5 whereafter a proof-of-concept implementation was provided in Chapter 6), an instantiation is applied to a real-world case study, utilising data sourced by the industry partner attached to this project. The generic modules within the four subcomponents of the InDIE framework are populated with specific algorithms and user-defined settings in order to illustrate the working of the framework. First, the background of the case study is presented, as well as an exploration of the provided industry partner data associated with this study, whereby the difference between benchmark data sets and real-world data sets is showcased by means of visualisations and inspections. Subsequently, the implementation of the four subcomponents of the InDIE framework are briefly described. Thereafter, the results returned throughout the implementation are analysed, followed by a detailed discussion of the case study validation — aided by the critique and validation of two subject-matter experts. The chapter is concluded with a summary of its contents.

7.1 Data set background

The previous chapter showcased a proof-of-concept implementation of the InDIE framework by introducing machine intelligence into the selection and application of document enhancement techniques, thereby improving the average OCR accuracy of the ICDAR 2019 SROIE [227](#) data set. Although the data set is considered as a high quality benchmark data set in the realm of document analysis, real-world data sets (*i.e.* data sets engineered for real-time industry products and services) tend to be associated with additional and unforeseen complexities — spawned by factors that arise in everyday industry data pipelines (*e.g.* human-prone errors, system downtime, lost files, corrupt files, and the mixture of legacy and new code). Consequently, the InDIE framework ought to be employed on a real-world data set in order to showcase the potential

utility of the framework when the data set is not specifically prepared for document analysis purposes.

As mentioned in Chapter 1, the financial service provider industry allocates considerable resources (specifically labour) to the mundane task of annotating paper-based document information in order to be used in downstream products and services. An array of paper-based document classes may form part of these downstream products and services, namely bank statements, payslips, identification documents, and application forms, to name but a few. The need for automated document information extraction systems is therefore especially relevant in this sector.

In order for a financial service provider to provide a client with an unsecured loan, the client's last few months of transactional history, identification document, and payslip are required [41]. Traditionally, the client would be required to visit a branch of the financial service provider and to physically provide these documents for a consultant, who would then manually annotate the values in the presence of the client and input the data into the computer system for downstream analysis. The annotation of these paper-based documents is typically time consuming and also degrades the client's experience as the time utilised on annotations could rather have been used to service more pertinent client-specific needs [295]. If a client utilises a specific financial service provider as their *primary* financial provider (*i.e.* the client's monthly salary is deposited into this financial service provider's system), the financial service provider would have access to the client's transactional data, personal identification data, and salary data, constituting all the required information. If the client only utilises this financial service provider as a secondary bank (*i.e.* the client's salary would be deposited into a bank account hosted by another financial service provider), the financial service provider would be deprived from the client's salary information. Consequently, the bank consultant would still be required to manually annotate the payslip in front of the client. In order to streamline this process and reduce waste, the financial service provider would prefer access to the aforementioned documents in a computer-readable data format, whereby a computer system can automatically transform the pixel-based information into computer-readable data, enabling the computer to extract the required information.

At the time of writing, the industry partner of this thesis, *i.e.* Capitec Bank, has eighteen million active clients [40], however, a large number of these clients do not deposit their primary salary into their Capitec Bank account. Consequently, Capitec Bank cannot utilise the transactional history of these clients to derive their salary information. It is therefore required of these clients to visit a branch, during which a consultant can annotate the client's payslip information for ingestion by a computer system. In order to streamline the unsecured loan application, the industry partner is considering the implementation of an OCR system that can accurately recognise the pertinent information on client payslips, however, due to the degraded quality of payslip document images, open-source OCR engines seldom deliver the required average OCR accuracy performance. Accordingly, the implementation of an additional system that can intelligently enhance the payslip document images is necessitated in order to improve the average OCR accuracy attained when extracting the required information. Consequently, the situation faced by Capitec Bank is deemed as an appropriate real-world case study for the implementation of the InDIE framework.

Capitec Bank has provided 2000 payslip document images and its corresponding annotations in a CSV file. The payslip document images originates from previous unsecured loan application branch visits from Capitec Bank clients, whereby the client would provide the consultant with the paper-based payslip, enabling the consultant to first scan the payslip with a flatbed scanner, whereafter the consultant would annotate the required information into the computer system. The consultants were required to annotate, if available, the following eleven feature strings: *Em-*

ployee name, employee surname, employee code, employee occupation, company name, company address, company postal code, base salary, gross salary, total deductions, and net salary.

As mentioned before, real-world data sets are not compiled to the same degree of high quality (or for the specific document analysis purpose) when compared with a benchmark data set. Since Capitec Bank has more than 840 branches across South Africa [40], each housing several consultants (specifically for receiving unsecured loan applications), it is an eventuality that the quality and standard of annotations would vary to a certain degree. Consequently, after an initial exploration of the provided payslip data set was carried out, several human-prone errors were found. It was observed that contrary standards were applied throughout the annotation of various GT features.

Consider the case in which the GT feature of a client’s “employee code” is “00be7762”. Where one consultant would annotate the employee code as printed on the payslip, another consultant might annotate it as “be7762”. This type of human-prone error is a regular occurrence throughout the data, and is spawned from the notion that removing and/or only partly annotating certain GT features which would (according to the consult) not be detrimental to the specific business goal at that time. Consider the case where the GT feature of “gross salary” of a client is “R9 899.96”, but it is annotated as “R9 900.00”. In the business case of annotating the value for an unsecured loan application, rounding up the gross salary by a mere four cents would not have a significant influence on whether the application is successful or not. In the case of measuring the OCR accuracy of the payslip document image, however, it would constitute as an erroneous GT feature as the GT feature value does not match the value printed on the payslip (in terms of a word-level evaluation metric). Consider the case in which the GT feature “workplace address” of a client comprises “PO BOX 141”. Where most consultants would annotate it as-is, it was found that some consultants would input an additional blank space between the character “P” and “O”, influencing the splitting of strings in the text formatting steps of the InDIE framework, and thereby falsely reducing the measured OCR accuracy.

Recall the finding that there are no missing values in the ICDAR 2019 SROIE annotation data (§6.2.1). Created as a data set for academic document analysis purposes, the ICDAR 2019 SROIE data set comprises exceptionally high quality annotated data, however, document analysis is (as expected) not considered a priority when annotating payslips for unsecured loan application purposes. It was observed that most GT features contained a plethora of missing values, as shown in Table 7.1. The GT feature with the most missing values, *i.e.* company address, has 1 556 missing values, which equates to 78% of the total payslips.

TABLE 7.1: Total missing values for each GT feature for the payslip data set.

GT feature name	Missing values	GT feature name	Missing values
Employee name	30	Company postal code	488
Employee surname	8	Base salary	1
Employee code	333	Gross salary	1
Employee occupation	23	Total deductions	23
Company name	165	Net salary	2
Company address	1 556		

After exploring a subset of the payslip document images, it was found that the majority of the images may be classified as having substandard and degraded quality. The degradations vary from a single quality issue to a combination of quality issues, namely: Shadows on certain sections of the background, multiple fold lines intersecting textual information, an array of skewed document scan angles, coffee stains, dark watermarks over important text strings, a

variety of font types and sizes, and the frequent occurrence of table and/or structural printed lines close to textual information.

Five distinct payslips are visualised in Figure 7.1 in order to illustrate the aforementioned data quality issues pertaining to this data set. It is important to note that these provided payslip document images are available online [69] and not sourced from the Capitec Bank payslip data set, as all payslip document images and corresponding annotations from the case study data set are deemed as confidential, as stipulated in a signed non-disclosure agreement between the author and Capitec Bank. A clear distinction between the ICDAR 2019 SROIE data set and the payslip data set is the discernable presence of tables and structural lines printed on the payslips. An additional observation is that payslip document images also comprise notably more textual information (when compared with the receipt document images), resulting in increased strings on the document images.

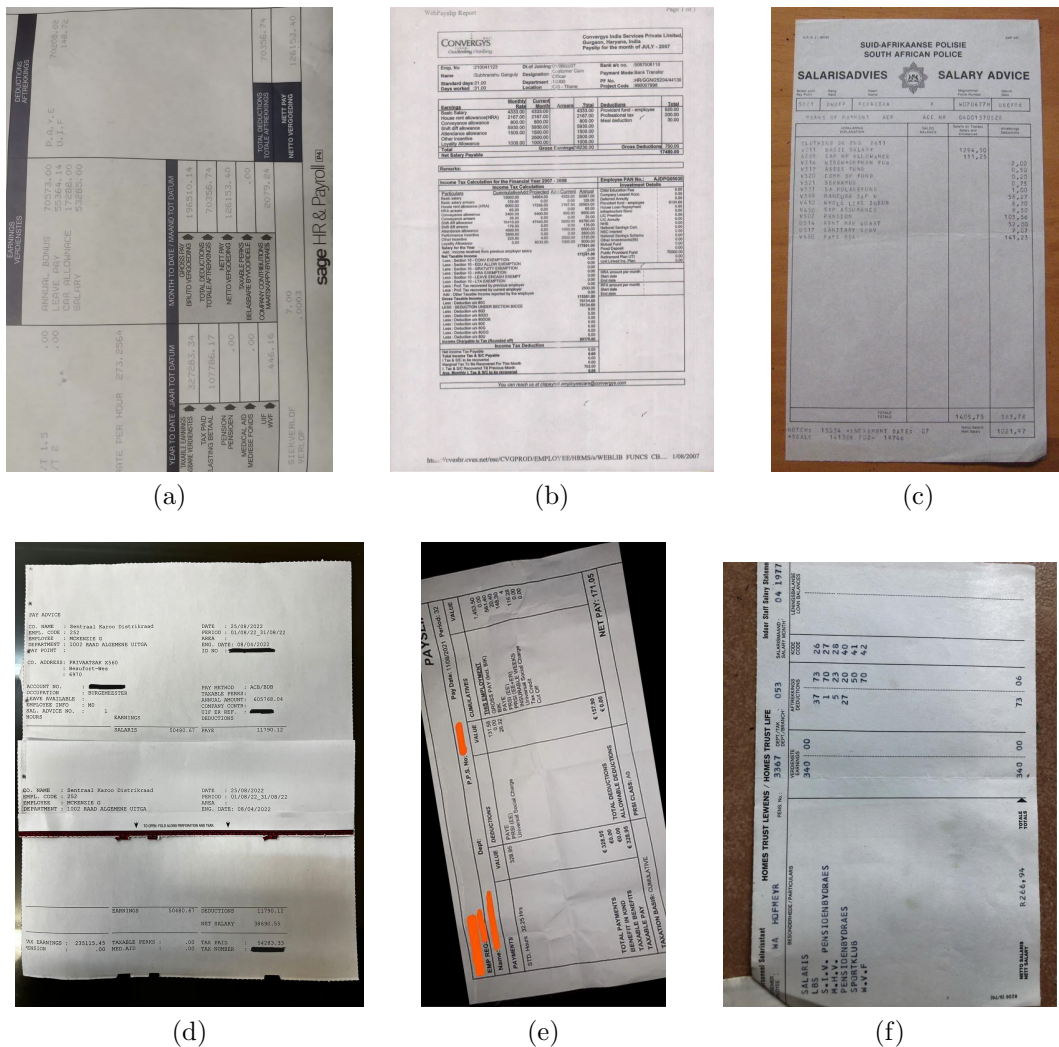


FIGURE 7.1: Examples of six generic payslip document images [69].

Consider the payslip presented in Figure 7.1(a). Almost half of the printed strings are grouped within predetermined text blocks, specifically made for those features. Moreover, it is also observed that several textual features are printed with a particularly light-toned ink, increasing the complexity of the recognition thereof. Another noteworthy observation is the usage of dark-

shaded backgrounds for column or row headings. This may potentially hinder the OCR engine, since most text are printed in black, but some headings are printed in white. Consider the payslip visualised in Figure 7.1(b), where an excessive number of printed strings are present on the payslip document image. The number of used strings results in the usage of a noteworthy small font size. As the font size of the characters decreases, the quality also decreases. This might hinder the OCR engine towards discriminating the character' shapes from one another, resulting in erroneous recognitions. The payslip document image shown in Figure 7.1(d) represents a commonly occurring case according to which the scanned payslip image comprises several paper-based pieces. This results in various shadows and unwanted lines on the payslip document image, unnecessarily increasing the complexity of the payslip document image. When inspecting the payslip presented in Figure 7.1(e), a skew and incorrectly oriented payslip document image is observed. The characteristics of this payslip are prevalent throughout the payslip data set, according to which several payslips are recognised to be either skew and/or incorrectly orientated.

7.2 Implementation of the InDIE framework

The Capitec Bank payslip data set is a real-world data set employed in industry. It comprises semi-annotated payslip document images recognised as having a substandard and degraded quality. Accordingly, in order to explore the utility of the InDIE framework when implemented on a real-world data set, the Capitec Bank payslip data set is deemed appropriate for this case study. Similar to the proof-of-concept implementation in §6.2, the aim in this section is to attempt to improve the average OCR accuracy of the provided real-world payslip document image data set by intelligently predicting which enhancement procedure to implement for each payslip document image. The remainder of this section comprises brief discussions of the case study implementation selections and estimations for each of the fourteen modules of the InDIE framework.

Filter features

As mentioned previously, eleven pertinent payslip features were annotated by in-branch consultants as part of unsecured loan applications by clients. The four factors discussed in §5.3.1 are explored in fulfilment of this module. First, the GT features are ranked according to their importance to the business goals at hand — *i.e.* extracting specific textual information for unsecured loan application purposes. Due to this specialised business domain of unsecured loan applications, it is required to ascertain GT feature rankings that reflect the true business value of the features. Accordingly, a suitable subject-matter expert, employed by the industry partner, is consulted on the importance of recognising certain GT features [295]. The GT feature rankings, as recommended by the subject-matter expert, are provided in Table 7.2. Net salary and company name are ranked as the most important features, while company address and employee code are ranked as the least important features.

Second, an analysis of the general proximity of the eleven GT features is performed. It is found that the information pertaining to the employee (and the company that employs them) is generally located on the upper third of the payslips. The base salary, gross salary, and total deductions are generally located in the middle third of the payslip, while the net salary is predominantly located on the lower third of the payslip.

Third, the annotation accuracy of each GT feature is inspected. By examining the missing data information provided in Table 7.1, it is found that the GT features employee code, company

TABLE 7.2: *GT features ranked according to the guidance by subject-matter expert.*

Rank	GT feature name	Rank	GT feature name
1	Net salary	7	Employee surname
2	Company name	8	Employee name
3	Occupation	9	Company postal code
4	Base salary	10	Company address
5	Total deductions	11	Employee code
6	Gross salary		

name, company address, and company postal code all have more than 15% of their values missing, significantly degrading the value of these GT feature annotations. Moreover, it is found that the annotations of monetary-type GT features (*i.e.* net salary, base salary, gross salary, total deductions) sometimes deviate slightly from the actual strings printed on the payslip document images. The most detrimental human-prone error is the rounding of monetary values, thereby completely deteriorating the potential OCR accuracy evaluation of the GT feature. Additionally, it is recognised that some employee name annotations comprise solely the initials of the clients, while the printed string is the full client name. Finally, with regards to annotation format, it is recognised that several monetary GT features deviate from a common standard, varying by including/excluding currency symbols, and utilising a “,” decimal separator instead of a “.” decimal separator.

After exploring and considering all of the aforementioned factors, it is concluded that the following eight GT features are selected for the case study data set: Net salary, base salary, gross salary, total deductions, company name, occupation, employee name, and employee surname.

Clean text and image data

The first text-based data cleaning procedure is the removal of all data entries that contain a missing value within one of their corresponding GT features. After the removal of these entries (and their corresponding payslip document images), 1 835 data entries and payslip document images remain.

Next, the GT feature strings are transformed into appropriate and standardised formats in order to be easily compared with the corresponding OCR output. It is found that all alphabetic characters are captured in uppercase format, and are left as-is. In order to mitigate the damage incurred by the monetary-type GT features due to rounding errors, it is decided to filter out all decimals (*i.e.* cents), thereby restoring the precision of the OCR accuracy, whilst having a negligible effect on the business goals. Monetary-type GT feature annotations for which a “,”-symbol is annotated as the decimal separator are correspondingly replaced by a “.”-symbol. As a final step, the eight captured GT features for each individual data entry are formatted as a list of strings. This is achieved by separating the information contained within a feature annotation into multiple strings through splitting the annotation content at each blank space and joining all the strings into a comma-separated list.

Since it is common for payslips to be printed in a landscape format, correctly orientating the images is a crucial step in the data wrangling process of document images. Accordingly, all the payslip document images are wrangled through the implementation of both (1) the skew correction operation and (2) the text orientation correction operation. A Hough transformer-based approach [193] is implemented for correcting any undesired angle in respect of the payslip document images. For the orientation correction, the word comparison method is implemented,

whereby the EasyOCR engine [141] is employed with respect to the four different angled payslip document image variants. After a manual inspection is performed, it is found that the two operations corrected the skew and/or incorrectly orientated payslip document images.

Implement OCR

Upon inspection of the payslip document images, it is found that the composition of the images is deemed complex. The payslip document images comprise various logos, background watermarks, complex font types and sizes, many printed lines overlapping with alphabetic strings with thick fonts, and a constant occurrence of shaded areas and fold lines. Therefore, it is required to select an OCR engine with the flexibility to be adapted to the required conditions. Taking the aforementioned into consideration, the EasyOCR engine is selected as the OCR engine for this case study implementation.

A small experimental sample of 30 payslip document images is created and utilised to fine-tune the hyperparameters of the EasyOCR engine. Several hyperparameters are selected by the user. First, the detection language is set to English. Next, the GPU option is set to false, as the industry partner device does not have a GPU, and the data ought to remain on the provided device as it contains markedly sensitive and confidential data. After performing a sensitivity analysis on the experimental sample, it is found that the detection text boxes ought to be fine-tuned for the payslip data set. The `add_margin` hyperparameter is tuned from its default value of 0.1 to 0.3, while the `width_ths` hyperparameter is increased from its default value of 0.5 to 0.65. The increase of the `width_ths` hyperparameter is due to the faulty merging of text boxes for payslip document images comprising characters with notably small font.

After the estimation of suitable hyperparameter values, the EasyOCR engine is applied to all of the payslip document images, producing corresponding OCR outputs (in a paragraph form). This is followed by the formatting of the OCR outputs, whereby punctuation marks and currency symbols are removed from the obtained strings. Thereafter, all alphabetic characters are transformed into uppercase characters. The single paragraph OCR output is then divided into a list of strings, separating the strings when a blank space is found in the paragraph. This enables the GT feature lists to be directly compared with the corresponding OCR output lists, as both are similarly standardised and appropriately formatted.

Engineer evaluation metric

The engineering of an appropriate evaluation metric is a crucial step in the InDIE framework implementation, as the downstream usage thereof influences the assignment of the subsequent engineered enhancement procedures. The receipt and payslip document types both reside in the domain of financial documents, requiring higher average OCR accuracies, as previously mentioned. Moreover, financial documents comprise several monetary-type GT features, thereby warranting a word-based evaluation metric.

Upon taking the aforementioned into consideration, a word-based evaluation metric is engineered, whereby each string within the GT feature lists holds equal weighting in the calculation of the OCR accuracy. Therefore, the evaluation metric for the payslip data set case study is alike to the evaluation metric engineered for the ICDAR 2019 SROIE data set proof-of-concept implementation.

Evaluate OCR results

The base OCR accuracy of a single payslip document image is computed by taking the intersection of the OCR output list and the GT feature output list, and dividing it with the total number of GT feature strings. Thereafter, the average base OCR is computed by taking the average of all the base OCR accuracies of the payslip document images, yielding the base performance of the EasyOCR engine on the payslip data set.

Select enhancement techniques

In this module, the expected impact from the implementation of several document enhancement techniques are tested by utilising the previously compiled experimental sample. The experimental sample is expanded upon by adding 15 high achieving OCR accuracy payslip document images and 15 low achieving OCR accuracy payslip document images. As previously mentioned, the addition of these payslip document images aids towards showcasing the possible improvement (or deterioration) caused by the various document enhancement techniques. The considered enhancement techniques include cropping, line removal, binarisation, noise removal, sharpening, and a sequential combination of the mentioned document image techniques.

First, the effects of cropping is evaluated. After inspecting the experimental sample, it is observed that the majority of payslip document images does not have an abundance of excess white space. This is a consequence of the plethora of information contained in a generic payslip, thereby requiring vastly more space for structuring and presenting the information than a typical document. Moreover, the usage of tables and printed lines results in a natural border on the edges of a document page, as showcased in Figure 7.1(a). Consequently, most payslip document images contain limited space for cropping purposes. After implementing the document enhancement technique on the experimental sample, it is found that the majority of the payslip document images deteriorated as a result of erroneous cropping, with only a few that improved (in terms of OCR accuracy). It is therefore decided not to include cropping as a potential document enhancement technique.

While line removal was disregarded in the ICDAR 2019 SROIE data set proof-of-concept implementation, it is expected to be a prominent method for the enhancement of payslip document images. By visually inspecting the experimental sample, it is observed that the overwhelming majority of the payslip document images contain either printed tables or several printed structural lines. After implementing line removal, it is found that several payslip document images showcased a considerable improvement when compared with its base OCR accuracy, with several payslip document images improving by more than 0.30 in terms of OCR accuracy. Simultaneously, several payslip document images also deteriorated after the removal of lines, with a few showcasing a marked deterioration of 0.45. Intelligently identifying which payslip document image requires line removal would be beneficial towards improving the average OCR accuracy. Therefore, line removal is included as a potential document enhancement technique.

The document enhancement techniques of binarisation, noise removal, and sharpening all performed similarly on the experimental sample payslip document images, with a range of respectable improvements and notable deteriorations. Binarisation, noise removal, and sharpening are therefore included as potential document enhancement techniques.

Create enhancement procedure categories

Upon considering the aforementioned document enhancement techniques, eight enhancement procedure categories are created, as listed in Table 7.3. The first enhancement procedure is called Base image, once again a requirement for ensuring that if all other enhancement procedures result in a deterioration of OCR performance, the document image ought to remain unaltered. The next four enhancement procedures, *i.e.* category 2–5, all constitute the implementation of a single enhancement technique. Categories 6–8 all comprise the sequential implementation of line removal, and then either noise removal, binarisation, or sharpening, respectively.

TABLE 7.3: *Constructed enhancement procedures for the payslip data set case study.*

Category	Enhancement procedure
1	Base image
2	Line removal
3	Binarisation
4	Noise removal
5	Sharpening
6	Line removal & noise removal
7	Line removal & binarisation
8	Line removal & sharpening

Assign enhancement procedure categories to images

After the creation of the eight enhancement procedures, each payslip document image is subjected to eight image transformations, each corresponding to their own payslip document image variant. Thereafter, OCR is implemented in respect of each variant, and the OCR output formatted. This enables the evaluation of each enhancement procedure for each payslip document image. The enhancement procedure corresponding to the best OCR accuracy is then deemed the best category for the specific payslip document image, and is therefore assigned its corresponding label.

Upon further analysis of the attained assignment results (elaborated upon later in this chapter), it was found that several enhancement procedures were either not assigned to sufficient payslip document images in order to be reasonably separated into the training, validation, and testing sets, or that the improvement showcased by the implementation of the enhancement technique did not produce the desired results. Consequently, it is decided to only consider three enhancement procedure categories, namely Base image, Line removal, and Sharpening.

Select convolutional base

Referring back to the exploration of document image classification discussed in §6.2.3, it was found that at the current state of pre-trained CNN models, the VGG-16 model has some capabilities of extracting feature maps that represent the intrinsic patterns between document images and their assigned labels. Accordingly, it is suggested that the VGG-16 pre-trained model may transfer its learning experience from being trained on ImageNet to the domain of document image classification. Similarly, the same argument is employed to select an appropriate convolutional base for the payslip data set model. Consequently, the pre-trained VGG-16 model is selected as the convolutional base for the prediction model.

Build and train model

First, the payslip data are separated into three data sets, *i.e.* the training set, the validation set, and the testing set, based on a 60%, 20%, 20% split. Thereafter, the training set and validation minority classes (*i.e.* Line removal and Sharpening) are oversampled in order to balance the data set. Although one may technically reduce or enlarge the input images, Das *et al.* [60] recommend that the payslip document images within each data set to be resized to $224 \times 224 \times 3$, since the VGG-16 model was trained upon these specifications. This is followed by augmenting the training and validation data sets with rotations of 20° , shifting the width by a range of 0.2, shifting the height by a range of 0.2, adding a zoom range of 0.3, and permitting the horizontal and vertical flipping of images. Each of the payslip document images is then normalised by dividing each pixel value by 255.

The VGG-16 feature extraction layers (*i.e.* without the VGG-16 classifier) forms the convolutional base of the prediction model. For the first training round, all convolutional layers are kept frozen, making their weights fixed and untrainable. A global average pooling layer is added to further reduce the data, followed by a flatten layer. Three dense layers are added as a new trainable classifier (as recommended by Das *et al.* [60]), with 500 neurons for the first layer, 250 neurons for the second layer, and three neurons (equal to the number of classes) for the third layer. The first two dense layers employ the ReLU activation function, while the third employs a softmax activation function. Dropout layers are added between the flatten layer and the first dense layer, and between all the dense layers, with initial dropout values of 0.5, 0.3, and 0.2, respectively. The Adam optimiser is selected with an estimated learning rate of 0.0001. Moreover, the categorical cross-entropy is employed as the loss function for this model, with an AUC evaluation metric. As recommended by Afzal *et al.* [2], the initial number of training epochs is set to 50. Several rounds of training are performed where the training and validation curves are utilised to fine-tune the aforementioned hyperparameters.

Evaluate model

After the prediction model is fine-tuned to the user's satisfaction (and the model's capabilities), the predictive power of the model is evaluated in respect of the unseen payslip document images contained within the previously separated test set. The results obtained by this analysis are visualised and elaborated upon later in this chapter.

Implement enhancement procedure predictions on test set images

The twelfth module of the InDIE framework facilitates the implementation of the predicted enhancement procedure categories on the test set payslip document images. Each payslip document image in the test set is transformed according to its corresponding enhancement procedure prediction, unless it received the Base image prediction, in which case the image is left unaltered. The newly transformed test set payslip document images are then stored and passed on to the next module for OCR processing.

Implement OCR on enhanced test set

The newly transformed test set payslip document images are subjected to the previously selected EasyOCR engine and its accompanying estimated hyperparameters for text recognition. It is important to use precisely the same OCR engine hyperparameters in order to fairly compare

the OCR performances. After an OCR output is produced for each test set payslip document image, the OCR output is standardised and cleaned according to the aforementioned list format (for comparison purposes).

Evaluate final OCR results

The fourteenth and final module of the InDIE framework facilitates the final evaluation of the OCR performance after machine intelligence was incorporated into the enhancement of the payslip document images. The OCR output lists are compared with the corresponding GT feature list, resulting in a new OCR accuracy for each of the test set payslip document images. The new average enhanced OCR accuracy for the test set is computed, whereafter is it compared with the average base OCR accuracy for the test set. The analysis of these results are elaborated upon in the next section of this chapter.

7.3 Results produced by InDIE framework implementation

Copious results were produced throughout the case study implementation of the InDIE framework. These results were explored so as to gain valuable insights which were, in turn, employed in order to carry out design decisions and estimations. The choices made by the user ultimately influenced the final performance of the InDIE framework, therefore, the results ought to be scrutinised and explored in order to verify the decisions made throughout the execution of the case study. The remainder of this section is devoted to the visualisation, exploration, and in-depth discussion of the results obtained throughout the implementation of the InDIE framework on the payslip data set.

Initial base OCR results

Module 5.0 (*i.e.* the evaluation of the base OCR results) facilitated the assignment of a base OCR accuracy for each payslip document image, calculated by utilising the evaluation metric designed in Module 4.0. The average base OCR accuracy attained by the EasyOCR engine for the payslip data set was 0.7375. Recall that in the proof-of-concept implementation that an average base OCR accuracy of 0.7695 was achieved. This is an indication that EasyOCR engine might struggle with greater difficulty under the more complex and degraded quality of the real-life payslip data set.

Valuable insights may be obtained by visualising these computed accuracies. The results are illustrated in Figure 7.2, where the payslip document images that achieved specific base OCR accuracies are binned together in increments of 0.1. The red-coloured bars indicate the payslip accuracy bins that achieved poor results, while bars coloured green indicated more favourable results. When comparing the visualisation of base OCR accuracies in Figure 7.2 with the visualisation of base OCR accuracies of the ICDAR 2019 SROIE data set in Figure 6.13, it is observed that there are proportionally more payslip document images in the lower end of the chart than there were receipt document images in the proof-of-concept results. Accordingly, this verifies the comparatively poorer performance of the case study data set results when compared with the proof-of-concept data set result. An unsurprising result given the complexity associated with real-world data sets.

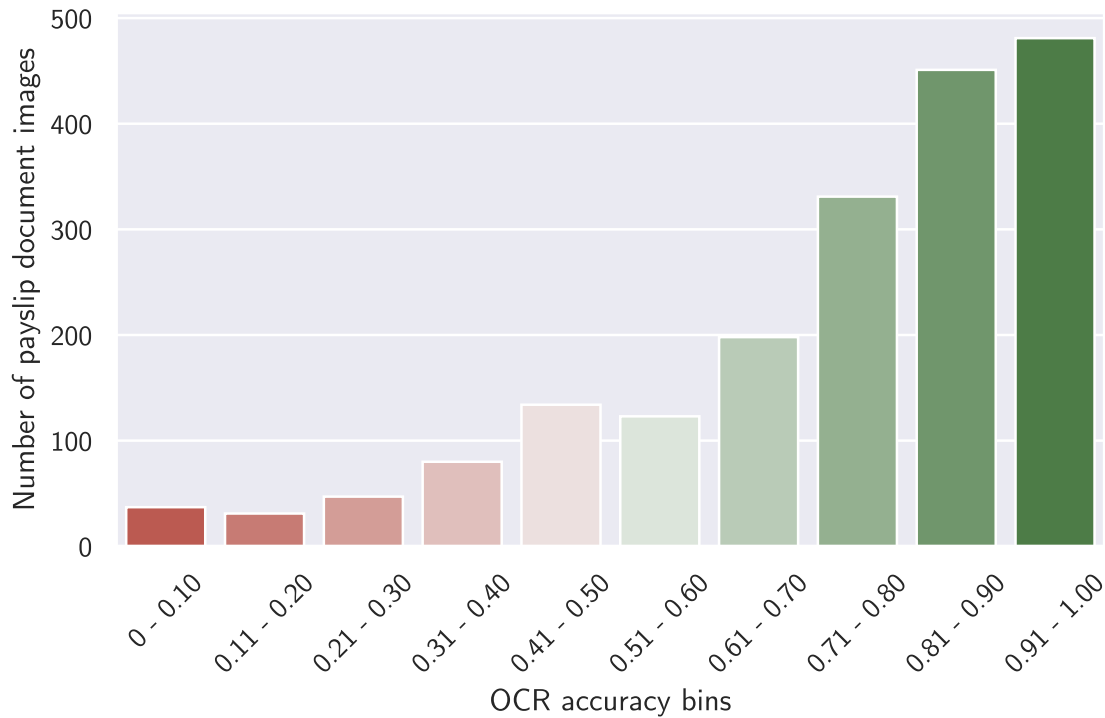


FIGURE 7.2: Graphical illustration of the base OCR accuracy results for the payslip data set.

Enhancement procedure results

In order to determine which enhancement procedures ought to be included in the final modelling stage of the case study it is important to analyse the portion of the payslip image data set to which each enhancement procedure was assigned and the corresponding impact on the average OCR accuracy.

Through the implementation of Module 8.0, each of the payslip document images were labelled according to the best performing enhancement procedure category (in terms of OCR accuracy). The distribution of the considered labels are presented in Table 7.4, where the first column indicates the category name and the second column indicates the portion of the payslip document images assigned to the corresponding label (*i.e.* assigned if the enhancement procedure obtained the best OCR result across all of the categories).

The first noteworthy observation is that the Base image category is deemed the best enhancement procedure for 0.60 of the payslip document images, constituting the majority of the payslip document images. This indicates that one might need to prepare for the possibility that the desired average OCR accuracy improvement might be lower than expected, since only 0.40 of the payslip images can showcase an improved OCR accuracy when altered with the enhancement procedures created in the Labelling subcomponent. Upon further analysis of the obtained results presented in the table, it is apparent that two enhancement procedure categories, *i.e.* Line removal and Sharpening, are assigned to considerably more payslip document images than their counterparts, accounting for 0.13 and 0.12 of the payslip document images, respectively, while the remaining enhancement procedures each account for less than 0.04 of the entire payslip image data set. Consequently, there is a possibility that the transfer learning model might find it difficult to learn the intrinsic patterns of these enhancement procedures, which were assigned

TABLE 7.4: Summary of the original labelling distribution when considering all eight enhancement procedure categories.

Enhancement procedure category	Portion of image category assignment
Base image	0.60
Line removal	0.13
Binarisation	0.03
Noise removal	0.04
Sharpening	0.12
Line removal & noise removal	0.03
Line removal & binarisation	0.02
Line removal & sharpening	0.03

to such small portions of the payslip data set, since so few training examples would be provided to the model.

Another factor to consider is the portion of images that would be improved upon (in terms of OCR accuracy) if each enhancement procedure was individually applied to the entire payslip data set without any added machine intelligence. The original results of the individual application of each enhancement procedure are shown in Table 7.5, where the first column indicates the enhancement category name that was individually implemented and the second column indicates the portion of the payslip document images that showcased an improved OCR accuracy (when compared with their corresponding base OCR accuracies). When inspecting the table, it was found that two enhancement procedures, *i.e.* Sharpening and Line removal & sharpening, achieved the best results, with both improving 0.21 of the payslip document images. Following closely are Line removal & noise removal with a portion of 0.20 which improved, and Line removal with a portion of 0.17 which improved.

It is interesting to note that both enhancement procedures that incorporated binarisation significantly underperformed when compared with the remainder of the enhancement procedures, indicating that it might not be beneficial to consider binarisation for this particular payslip image data set. Another noteworthy observation is that although the individual application of Line removal resulted in a portion of 0.17 of the payslip document images improving and the individual application of Noise removal resulted in an a portion of 0.13 of the payslip document images improving, the enhancement procedure category combining Line removal & noise removal resulted in an improvement of 0.20 of the payslip document images. This is an indication that it can be beneficial to sequentially apply document image enhancement techniques.

TABLE 7.5: Summary of the portion of images which showcased an improved OCR accuracy after the application of enhancement procedures.

Enhancement procedure category	Portion of entire data set which showed OCR improvement
Line removal	0.17
Binarisation	0.05
Noise removal	0.13
Sharpening	0.21
Line removal & noise removal	0.20
Line removal & binarisation	0.09
Line removal & sharpening	0.21

The enhancement procedures of Binarisation, Noise removal, Line removal & noise removal, Line removal & binarisation, and Line removal & sharpening were all assigned to 0.04 (or less) of the payslip document images — a portion of 0.04 constituting merely 73 of the 1 835 payslip document images. Consequently, the enhancement procedure categories of Binarisation, Noise removal, Line removal & binarisation, and Line removal & sharpening are removed, and the few corresponding payslip document images relabelled with the best remaining enhancement procedure category.

It was found that if all of the assigned enhancement procedures are precisely predicted by a transfer learning model, and subsequently implemented, the average OCR accuracy of the payslip data set would be 0.7936 (referred to as the average best OCR accuracy). This equates to improving the average base OCR accuracy from 0.7375 to the average best OCR accuracy of 0.7936, constituting a theoretical improvement of 0.0561. Through exploring the GT feature lists engineered in Module 2.0, it is found that the average payslip comprises around ten GT feature strings. Accordingly, with the cleaned payslip data set comprising 1 835 images, the total number of sought after GT feature strings was computed as 18 350. An average OCR accuracy improvement of merely 0.0561 would result in an additional 1 030 GT features, a substantial improvement in potential business value.

A comparison of the base OCR accuracy distribution with the theoretical best OCR accuracy distribution, in terms of ten accuracy bins, is graphically illustrated in Figure 7.3. The results of the base payslip document images are shown in red, and the theoretical best payslip document images in green. The most considerable difference between the two sets of results is the significant increase of payslip document images in the “0.91–1.00” accuracy bin. It can be observed that the correct application of the enhancement procedures improved the OCR accuracies of several payslip document images from the lower accuracy bins to the “0.91–1.00” accuracy bin.

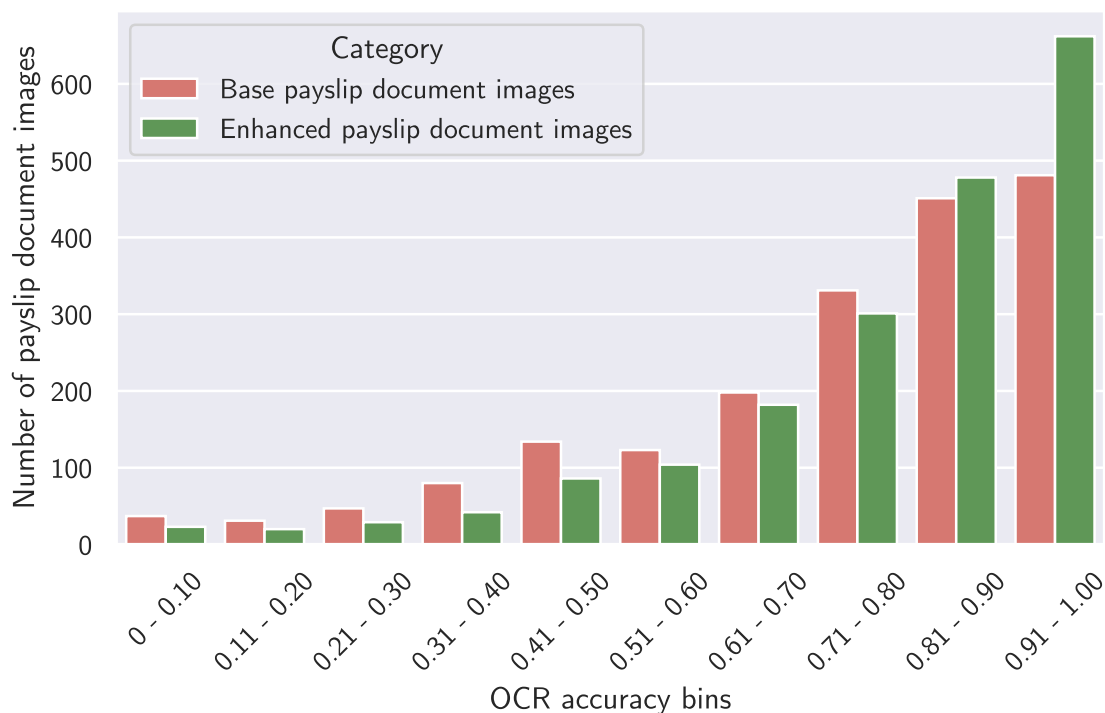


FIGURE 7.3: Graphical comparison of the number of payslip document images from the base OCR accuracy scenario and the enhanced OCR accuracy scenario, in terms of the ten accuracy bins.

Consequently, visual inspection of the presented chart verifies that the correct implementation of the assigned enhancement procedures results in a reduction of low OCR accuracy payslip document images and an increase in high OCR accuracy payslip document images.

In order to gain insights into the impact of implementing an enhancement procedure without any added machine intelligence *versus* correctly predicting whether to implement the procedure or not, each enhancement procedure was individually implemented on all the payslip document images and corresponding average OCR accuracies were computed. These results are provided in Table 7.6, according to which the first column indicates the enhancement procedure category and the second column indicates the average OCR accuracy if the corresponding enhancement procedure was applied to all of the payslip document images. A noteworthy finding was that only one enhancement procedure improved upon the average base OCR accuracy, *i.e.* the implementation of Line removal, improving from 0.7375 (*i.e.* the average base OCR) to 0.7425. By far, the full implementation of binarisation deteriorated the average OCR accuracy the most, decreasing to 0.5494, once again showcasing the detrimental effect that an enhancement procedure can have when not applied to the appropriate payslip document images.

TABLE 7.6: Summary of the OCR results obtained when applying enhancement procedure categories on the entire payslip data set.

Enhancement procedure category	Average full OCR accuracy
Base image	0.7375
Line removal	0.7415
Binarisation	0.5494
Noise removal	0.7168
Sharpening	0.7200
Line removal & noise removal	0.7247
Line removal & binarisation	0.5536
Line removal & sharpening	0.7160

Moreover, it was also examined which individual enhancement procedure would produce the best average OCR accuracy if it is assumed that a transfer learning model can precisely predict whether a specific individual enhancement procedure ought to be applied or not — *i.e.* only applied to the payslip document images if it would produce an improved OCR accuracy. These results are shown in Table 7.7. Since it is assumed that the theoretical transfer learning model can predict with perfect accuracy, it is expected that all the obtained average OCR accuracies ought to be an improvement on the average base OCR accuracy. The individual application of the Line removal & sharpening, once again, obtained the best results with an average best OCR accuracy of 0.7660, while Line removal & noise removal came second, and Sharpening third.

Upon considering all of the results explored in the aforementioned tables and figures, it may be concluded that in order to provide the machine learning model with adequate training, validation, and testing data, that only enhancement procedures that are assigned to a sufficient portion of the data while simultaneously producing acceptable OCR improvements may be considered. Therefore, only three enhancement procedures were included, namely Base image, Line removal, and Sharpening. Accordingly, the payslip document images were relabelled by only considering these three enhancement procedures.

The newly attained labelling results are presented in Table 7.8. The Base image enhancement category was assigned to the majority (a 0.68 portion) of the available payslip document images, with Line removal assigned to a 0.20 portion of the available payslip document images, and

TABLE 7.7: Summary of the average best OCR results obtained when applying enhancement procedure categories only on payslips that would showcase an improved OCR accuracy.

Enhancement procedure category	Average best OCR accuracy
Line removal	0.7604
Binarisation	0.7437
Noise removal	0.7518
Sharpening	0.7624
Line removal & noise removal	0.7644
Line removal & binarisation	0.7507
Line removal & sharpening	0.7660

Sharpening only to a 0.12 portion of the available payslip document images. It was observed that after the relabelling procedure, Line removal increased by an additional 0.07 portion of the available payslip document images when compared with the original assignment, while the portion assigned by the Sharpening enhancement procedure remained unchanged.

TABLE 7.8: Newly attained enhancement procedure label results.

Enhancement procedure category	Portion of images assigned per category
Base image	0.68
Line removal	0.20
Sharpening	0.12

After the newly assigned labels, it was found that if, theoretically, the transfer learning model predicts all the assigned labels to precision (*i.e.* the best possible scenario), an average best OCR accuracy of 0.7772 may be obtained. This is a stark improvement when compared with the original average base OCR accuracy of 0.7375.

A comparison of the base OCR accuracy distribution with the new theoretical best OCR accuracy distribution (when only considering the three selected enhancement procedures), in terms of ten accuracy bins, is graphically illustrated in Figure 7.4. It can be observed that the number of payslip document images within bins “0–0.10” to “0.71–0.80” are reduced after the implementation of the enhancement procedures. Note the significant gains observed within the “0.91–1.00” bin, indicating that the positive effects of an enhancement procedure not only improves the OCR accuracy of most payslip document images from one bin to the next, but in most cases it improves the OCR accuracy from its current bin to the best performing bin.

Model training and validation results

After the assignment of the enhancement procedure categories in Module 8.0, the model was trained and fine-tuned several times. The training and validation curves of the final training round is presented in Figure 7.5, where the blue curve represents the training AUC curve, and the orange curve represents the validation curve. The hyperparameters employed for the final training round is as follows: A batch size of 32, image dimensions of $224 \times 224 \times 3$, a learning rate of 0.0001, 50 epochs, a fully frozen VGG-16 convolutional layer, a dropout layer after the global pooling layer with a dropout rate of 0.2, a dense layer with 250 neurons, followed by a dropout layer with a dropout rate of 0.5, a dense layer with 100 neurons, and a dropout layer

with a dropout rate of 0.5 before the final dense layer comprising three neurons and a softmax activation function.

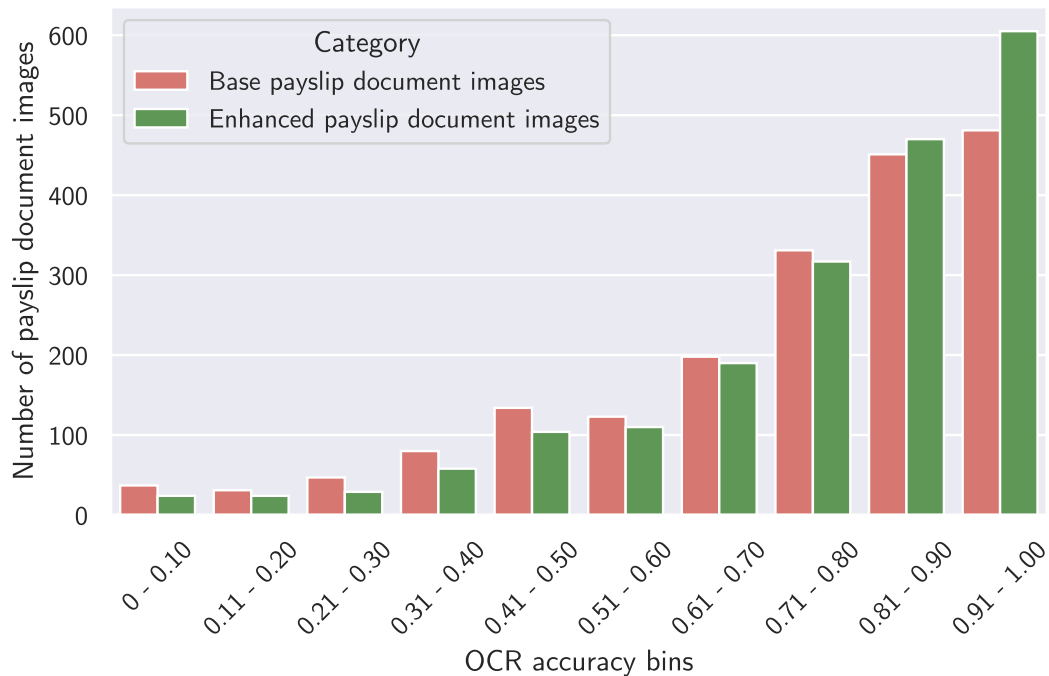


FIGURE 7.4: Graphical comparison of the distribution of payslip document images from the base OCR accuracy scenario and the enhanced OCR accuracy scenario, in terms of the ten accuracy bins.

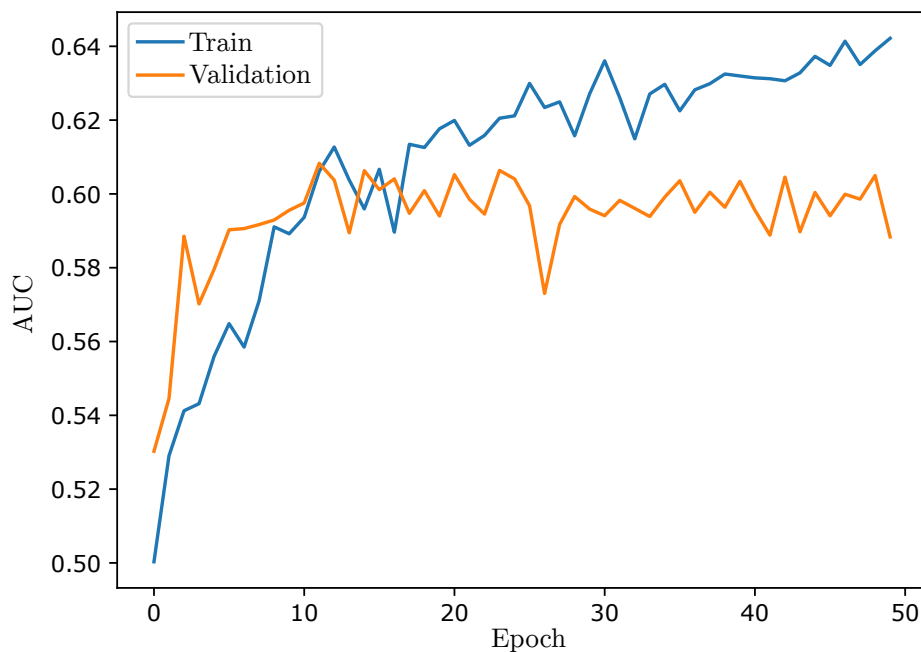


FIGURE 7.5: Visualisation of training (blue) and validation (orange) AUC curves for the training of the prediction model.

After inspecting the AUC curves, it was found that both the training and validation curve showcase a steady increase from epoch 0 to around epoch 15, whereat the two curves converge. After epoch 15, however, the validation curve stabilises at around an AUC score of 0.61 while the training AUC score continues to increase to an AUC score of 0.65. This is an indication that the model started to overfit on the training data and therefore training was terminated.

Multiple rounds of training were performed whereby unfrozen convolutional layers were introduced to the training of the model. Significant overfitting was observed, which was followed by the dropout rates being increased. The added regularisation did not, however, prevent the model from overfitting. Consequently, all convolutional layers were kept frozen.

After the model was fine-tuned, it was evaluated on the test set. The test set confusion matrix is presented in Figure 7.6. It was found that 0.57 of Base image category was predicted correctly, 0.52 of Line removal category was predicted correctly, and 0.44 of Sharpening was predicted correctly. It was previously construed that although misclassifying an image as Base image results in a loss of potential improvement, the OCR performance is at least not undermined by merely leaving the image in its original state. Misclassifying an enhancement procedure as another implementable enhancement procedure, however, might be detrimental to the OCR accuracy of an image as there is no *prior* knowledge whether the implemented document enhancement technique will improve or deteriorate the OCR accuracy. Accordingly, it ought to be a priority of the model to reduce false positives predicted as Line removal or Sharpening.

In the case of the payslip test set, 0.14 of the payslip document images holding an actual label of Sharpening was falsely predicted to be Line removal. Moreover, 0.32 of the payslip document images holding an actual label of Line removal was falsely predicted to be Sharpening. Although it is recognised that it would be desired to reduce these false negative results even further, it shows that the trained model has the capability to learn from the intrinsic patterns within the data, and thereby reduce the (potential damaging) false negative predictions.

		Actual		
		A	B	C
Predicted	A	0.57	0.16	0.42
	B	0.16	0.52	0.14
	C	0.28	0.32	0.44

A: Base image
B: Line Removal
C: Sharpening

FIGURE 7.6: Visualisation of the confusion matrix for the payslip test set.

Enhanced test set OCR results

The final results to be explored were obtained from the fourth and final subcomponent of the InDIE framework, *i.e.* the Analysis subcomponent, whereby the predicted classes were implemented on the payslip document images of the test set.

Module 12.0 facilitated the implementation of the predicted classifications of the test set payslip document images. Thereafter, in Module 13.0, the transformed (and hopefully enhanced) test set payslip images underwent OCR and output formatting, producing corresponding enhanced OCR output lists. The newly attained OCR output lists were then subjected to OCR evaluation (employing the engineered evaluation metric), producing new enhanced OCR accuracy values for each payslip document image within the test set. Lastly, the new average enhanced OCR accuracy was computed.

Presented in Table 7.9 are the original average base OCR accuracy for the test set, the average full Line removal OCR accuracy (*i.e.* if Line removal was applied to the entire test set), the average full Sharpening OCR accuracy (*i.e.* if Sharpening was applied to the entire test set), and the newly attained average enhanced OCR accuracy (*i.e.* the test set OCR results which utilised the intelligent predictions). The average base OCR accuracy for the test set was 0.7382. If one would implement Line removal on the entire test set, the average OCR accuracy would actually improve to 0.7402, showcasing the value of removing printed lines from payslip document images. If Sharpening was applied to the entire test set, the average OCR accuracy would fall to 0.7200, a decrease of 0.0182. If the test set payslip images are enhanced precisely as labelled (*i.e.* the model predicts 100% correctly), the average OCR accuracy would increase to 0.7744, *i.e.* an increase of 0.0362. Finally, after the implementation of the trained transfer learning model predictions, a new average enhanced OCR accuracy of 0.7444 was achieved. This increase constitutes an improvement of 0.0062. Considering the 1 835 payslip document images available, the implementation of the InDIE framework would increase the number of correctly recognised GT feature strings with an additional 114 strings, which would previously not be recognisable by the OCR engine. Considering that the industry partner receives several several thousands of unsecured loan applications every month, this small increase in recognition can result in a significant increase in business value.

TABLE 7.9: Average OCR accuracy scenarios for different test set conditions.

Test set category	Average OCR accuracy
Average base OCR accuracy	0.7382
Average full Line removal OCR accuracy	0.7406
Average full Sharpened OCR accuracy	0.7200
Average best OCR accuracy	0.7744
Average enhanced OCR accuracy	0.7444

A clear OCR accuracy improvement was achieved when comparing the newly computed average enhanced OCR accuracy (*i.e.* 0.7444) with the average full OCR accuracies attained by applying either Line removal or Sharpening to the entire test set. In order to gain more insight into why this improvement was realised, it is important to scrutinise the number of payslip document images that either improved, remained unchanged, or deteriorated (compared with the base OCR accuracies).

Presented in Table 7.10 are detailed results of the three aforementioned categories. Upon inspecting the portion of the test set that improved, it was found that the intelligently enhanced images produced the smallest number of improved payslip document images, with only 59 show-

casing improved OCR accuracies. Although this seems to contrast the average OCR accuracy improvement, it is found that the overall improvement does not stem from enhancing more payslip document images, but rather from reducing the number of payslip document images that were erroneously altered. After applying Line removal to the entire test set, a 0.16 portion of the payslip document images deteriorated. After applying Sharpening to the entire test set, a considerable 0.30 portion of the payslip document images deteriorated. After altering only the payslip document images according to their intelligently predicted enhancement procedures, only a 0.11 portion of the payslip document images deteriorated — a significant reduction in the portion of deteriorated payslip document images.

This is reflected in the improved/deteriorated ratio, where a higher score is representative of the desired outcome. After applying Line removal on the entire test set, an improved/deteriorated ratio of 1.0169 was attained, applying Sharpening on the entire test set produced an improved/deteriorated ratio of 0.6759, and the intelligently predicted application achieved an impressive improved/deteriorated ratio of 1.4390 — a clear indicator that the utilisation of the transfer learning model outperformed the rudimentary application of document enhancement techniques.

TABLE 7.10: A comparison of the number of improved, unchanged, and deteriorated payslip document images when Line removal is applied to all the images, Sharpening is applied to all the images, and when only the intelligently predicted images are enhanced.

Impact category compared with the original base OCR	Line removal on all images		Sharpening on all images		Only predicted images	
	Number	Ratio	Number	Ratio	Number	Ratio
Improved	60	0.16	73	0.20	59	0.16
Same	249	0.68	187	0.51	268	0.73
Deteriorated	59	0.16	108	0.30	41	0.11
Improved/deteriorated		1.0169		0.6759		1.4390

In conclusion, when considering the aforementioned analysis of results obtained from the implementation of the InDIE framework with respect to a real-world data set, it was found that the addition of machine intelligence reduced the detrimental effects of document enhancement techniques by intelligently predicting which payslip document images require which type of enhancement. Therefore, the average OCR performance of the selected OCR engine was improved with the utilisation of machine intelligence.

7.4 Subject-matter expert validation

During the implementation of the InDIE framework in respect of the real-world payslip data set, it was found that the utilisation of machine intelligence to enhance the payslip document images can lead to improved OCR performance, even if the original intent for the data set was not for the purpose of document analysis — thereby numerically validating the practical workability of the framework. In pursuit of an additional measure to attain credibility and foster confidence in the utility of the framework when applied to a real-world environment, the proposed framework was presented to two key Capitec Bank representatives.

Interviews were conducted with (1) Mr David Gouvias [97] and (2) Mr Johan Olivier [207]. Gouvias holds an Honours in Computer Science and Mathematics from the University of Cape Town and is currently a senior data scientist at Capitec Bank. Gouvias has more than 30 years

of experience as a systems architect, steering software programs into production for various reputable companies such as *Sanlam*, *Discovery Health*, and *inQuba*. Olivier, currently a senior machine learning engineer at Capitec Bank, holds a BCom in Actuarial Sciences, Mathematics, and Statistics, and is board certified by the Institute and Faculty of Actuaries. Olivier has eight years of data science and machine learning engineering experience, developing and implementing various deep learning solutions into production. Both Gouvias and Olivier have developed software programs exploiting the power of OCR technology, and is therefore deemed as qualified and appropriate subject-matter experts.

The proposed InDIE framework was presented to the subject-matter experts in a structured manner. This included the detailing of the identified opportunity, the design and development of the framework architecture and modules within, the proof-of-concept instantiation with its corresponding Python scripts and results, and the case study demonstration with its corresponding Python scripts and obtained results. This enabled the Capitec Bank representatives to ascertain the practical workability of the framework, while additionally confirming the validity of the results obtained.

It is imperative for the architecture and composition of the InDIE framework to undergo scrutiny from industry representatives in order to uphold the stated utility of implementing the framework on real-world problems. The presentation was well received by the Capitec Bank representatives, with an overwhelming positive response to the potential utility of the framework for productionising real-world software solutions. Olivier expressed that by employing the three design principles of a well-designed software product and the six phases of the CRISP-DM reference model (discussed in §5.1) as inspiration for the architecture of the InDIE framework, renders the research conducted during this project an understandable and scientifically sound guide for any industry practitioner who desires to implement/improve their digitalisation processes.

“If I was tasked to improve the OCR performance of a real-world computerised system, I can see myself easily using the InDIE framework, especially as a logical starting point, adapting the modules and utilising more advanced enhancement techniques for the problem at hand.” — Johan Olivier

According to Gouvais, there is novelty in the approach adopted by the author, viewing the problem in a (pleasantly) unexpected way. Continuing, Gouvais stated that the flexible nature of the framework (*i.e.* the inter-changeability of the modules, as alluded to in §5.2) showcases the potential value attainable from an industry perspective.

With regards to the implementation of the Modelling subcomponent on the real-world payslip case study (discussed in §7.2), Olivier confirmed that the estimation and selection of model algorithms, model architecture, and hyperparameter settings were logical and acceptable. Although the model encountered some difficulty to learn from the provided image data (referring to the relatively low validation AUC score), Gouvias suspects that the model training can benefit greatly if a slightly larger data set is considered. Gouvias was impressed by the ability of the pre-trained VGG-16 convolutional base to transfer some of its learning experience from being trained on ImageNet to the realm of payslip document images. Additionally, Gouvais was surprised by the finding that training the classifier on slightly more than a thousand instances resulted in the model being capable of discriminating between the three considered classes with moderate accuracy.

Considering the newly attained average OCR test set performance (showcased in §7.3), Gouvais and Olivier both stated in agreement that any improvement at all is considered to be an impressive result.

“OCR problems scale. Therefore, albeit a rather small improvement in terms of the average accuracy gain, this is a major victory when considering the complexity of the underlying problem and the limited resources employed” — David Gouvias

7.5 Chapter summary

In this chapter, a real-world case study was executed by means of the InDIE framework. The Chapter opened by introducing the reader to the case study background and data set in §7.1. Thereafter, in §7.2, the implementation of the various InDIE framework modules were briefly discussed, followed by an in-depth analysis of all the obtained results. Finally, an expert-validation was provided in §7.4, whereby the methodology and results were scrutinised by two subject-matter experts.

CHAPTER 8

Conclusion

Contents

8.1	Thesis summary	151
8.2	Thesis contributions	153
8.3	Suggestions for future work	154

This chapter opens with a summary of the contents of this thesis in §8.1. This is followed by an appraisal of the contributions made in the thesis in §8.2. The chapter is concluded in §8.3 with a number of suggestions with respect to potential future work that may stem from the research conducted in this thesis.

8.1 Thesis summary

In addition to this final concluding chapter, the thesis comprised seven chapters aimed at addressing the problem statement and fulfilling the stated objectives. In Chapter 1, the introductory chapter of this thesis, a brief background of the problem is provided. In particular, the reader was introduced to the interrelation of the machine learning subfields of deep learning and computer vision. This was then expanded upon by the exploration of the utility of computer vision in the proliferating era of digitalisation, whereby the technology of OCR was introduced to the reader. Several modern implementations of OCR were discussed, whereafter the problem faced by the industry partner of this thesis was stated. Finally, the scope and objectives of the study were detailed, along with a brief outline of the thesis organisation, data collection and management, ethical considerations, and reporting of results.

Chapter 2, 3, and 4 were devoted to a review of the pertinent literature, in fulfilment of Objective I(a), Objective I(b), and Objective I(c), respectively. In Chapter 2, a literature review was presented detailing the principle concepts and terminology found in the machine learning paradigm of deep learning. This chapter opened with a discussion on ANN fundamentals which explored the working of FNNs, RNNs, LSTMs, and CNNs. This was followed by considering several prominent CNN architectures. In particular, the discussion expanded upon the architectures of AlexNet, VGG-16, and EfficientNet, showcasing the different compositions that a CNN architecture may describe. Thereafter, the concept and implementation of transfer learning (specifically in the realm of pre-trained CNN models) were discussed.

In Chapter 3, a further review was conducted on the pertinent literature pertaining to the major phases of OCR systems, the engineering of OCR evaluation metrics, and two prominent OCR

engines. This chapter was primarily focussed on providing the background information that is necessary to understand the underlying processes employed when implementing an OCR engine in pursuit of transforming pixel-based textual information into computer-readable data. First, the origins of OCR and typical challenges faced by OCR systems were discussed. Thereafter, the major phases of an OCR engine were explored. This included discussions on segmentation, feature extraction, and character classification. This was followed by considering different OCR evaluation metrics. In particular, the difference between character-based and word-based evaluation metrics were highlighted. Finally, the working and advantages of two prominent OCR engines considered within this thesis were explored, namely Tesseract and EasyOCR.

Chapter 4 was devoted to the pertinent literature relating to document image preprocessing techniques for improving OCR performance. The chapter opened with introducing the reader to the utility of implementing preprocessing techniques on document images in pursuit of improved OCR performance. In particular, typical challenges faced by OCR systems were presented *via* the consideration of a degraded document image. The implementation of several document image enhancement techniques remedied the document image, thereby facilitating an OCR engine to improve its recognition of the textual information on the considered document image. Thereafter, different document image enhancement techniques in the paradigms of geometric transformations and pixel transformations were explored, showcasing the potential benefit of introducing these techniques to degraded document images.

The InDIE framework was introduced to the reader in Chapter 5 through a discourse on the top-down description of the proposed framework, in fulfilment of Objective II. The chapter opened with an overview of the major steps involved in a generic data mining process, serving as a preliminary discussion for exploring the typical composition of a modular data mining framework. Thereafter, the modular design was adopted as the foundational building blocks for the proposed InDIE framework structure. After presenting the modular framework to the reader, each of the 14 modules, constituting the InDIE framework, were explored in detail.

Chapter 6 comprised a discussion on the verification of the InDIE framework proposed in this thesis through a proof-of-concept instantiation, in fulfilment of Objective III. First, the open-source and prominent benchmark receipt document image data set, which was selected for the proof-of-concept implementation, was discussed in detail. In particular, several receipt document images were visualised and scrutinised, followed by the inspection of the GT features which were contained within JSON files. Thereafter, the generic modules within the InDIE framework were populated with specific algorithms and user-defined settings in order to illustrate the utility of the InDIE framework. The four subcomponents of the InDIE framework were initialised and implemented on the selected benchmark data set. The results generated throughout the employment of the InDIE framework were evaluated and analysed. It was concluded that by utilising the InDIE framework it is (to some degree) possible to intelligently predict which enhancement procedure to implement on which receipt document image, resulting in an improvement of the average OCR accuracy attained by the selected OCR engine. Accordingly, the proof-of-concept instantiation was deemed a success, verifying that the implementation of the InDIE framework can result in an improved OCR performance.

In Chapter 7, a case study was performed in order to validate whether the InDIE framework can produce the desired outcome when implemented on a real-world data set provided by the industry partner of this thesis, in fulfilment of Objective IV. The chapter opened with an exploration of the industry partner data set, whereby the document images and the corresponding GT features were examined. Thereafter, the generic InDIE framework was populated with specific algorithms and user-defined settings in order to illustrate the working of the framework. The implementation of the four subcomponent of the InDIE framework was briefly discussed, followed

by an in-depth analysis of the acquired results. It was found that the addition of machine intelligence was successful in reducing the detrimental effects of the document enhancement techniques by automatically predicting document specific enhancement techniques. Therefore, the attained results validated the utility of the InDIE framework. In conclusion of the chapter, the proposed model was subjected to face validation by subject-matter experts, who deemed the results achieved by the model as satisfactory.

The present chapter serves as the conclusion of this thesis and includes sensible follow-up work which may stem from the research performed in this thesis, in fulfilment of Objective V.

8.2 Thesis contributions

The main contributions of this thesis are five-fold. This section contains a documentation and appraisal of these contributions.

Contribution I *The design and development of a generic framework for intelligently enhancing document images in pursuit of improved OCR performance.*

The overarching aim in this project was to design, develop, and demonstrate the practical workability of a generic framework capable of intelligently enhancing document images in pursuit of improved OCR performance. This was achieved through the process of data mining by incorporating various ideas, techniques and methods from three separate fields of research, namely deep learning, OCR, and document image enhancement, reviewed in Chapters 2, 3, and 4, respectively. An architectural design of the modular InDIE framework was formally presented and described in detail in Chapter 5. Being modular by design enables any module to be exchanged, modified or deleted in accordance with new findings in the literature without inhibiting the functional working of other modules of the framework.

Contribution II *Instantiation of the proposed framework as a proof-of-concept demonstrator involving a prominent document analysis benchmark data set.*

The design of the aforementioned intelligent document enhancement framework was not limited to a conceptual level only. The focus of Chapter 6 was to verify the potential utility of the InDIE framework. To this end, the framework was successfully applied to the ICDAR 2019 SROIE data set, a prominent document analysis benchmark data set, in the form of a computerised proof-of-concept demonstrator. The ICDAR 2019 SROIE data set comprised degraded receipt document images which experienced a reduction in average OCR performance (*i.e.* from 0.7720 to 0.7612 in respect of the test set) if the Sharpening enhancement procedure is applied universally. Accordingly, the ICDAR 2019 SROIE data set can be regarded as a suitable data set for verifying the utility of the framework. During this instantiation, details were provided to describe how the various modules constituting the framework may be realised. The framework was shown to be effective in intelligently minimising the incorrect implementation of the Sharpening enhancement procedure on the receipt document images, thereby achieving an improved average OCR performance (*i.e.* from 0.7720 to 0.7827 in respect of the test set).

Contribution III *Application of the proposed framework to a real-world case study.*

Although the execution of the proof-of-concept demonstrator with respect to the benchmark data set was successful, it does not reflect all the challenges and unforeseen complexities that accompany real-world document image data sets. The focus of Chapter 7 was to

validate the ability of the InDIE framework to provide utility on a real-world data set by performing an instantiation of the InDIE framework on a payslip document image data set provided by the industry partner of this project. The payslip document image data set and its corresponding annotations were not collected for the intent of being used in the realm of document analysis, and is therefore a suitable data set for validating the framework. The framework was shown to be effective in intelligently minimising the incorrect implementation of both the Line removal and Sharpening enhancement procedures on the payslip document images, thereby achieving an improved average OCR performance (*i.e.* from 0.7382 to 0.7444 in respect of the test set).

Contribution IV *A demonstration of transfer learning’s utility in respect of an ImageNet pre-trained VGG-16 architecture applied to document image enhancement.*

Based on the research conducted by Afzal *et al.* [2] in 2017 and Das *et al.* [60] in 2018, it was found that the pre-trained VGG-16 model exhibits the capabilities to transfer its learning experience from being trained on ImageNet to the domain of document image classification. Consequently, the intuition was that the feature maps extracted for achieving success in the realm of document image classification might achieve similar success in the realm of intelligent document image enhancement. Through the successful instantiations of the proof-of-concept demonstration and the real-world case study, it was found that the transfer learning model could learn intrinsic patterns within the document image data, proving the intuition to be correct. This is considered to be a valuable and novel finding, opening further exploration of transfer learning approaches within this field.

Contribution V *Suggestion of potential future projects emanating from thesis contributions.*

An abundance of research has been conducted in the realm of document image analysis, however, the recent increase in computational capabilities has enabled the application of new deep learning and computer vision approaches. Although the proof-of-concept and case study instantiations verified and validated the potential utility of the InDIE framework, further analysis and exploration is required to improve upon the obtained results. Consequently, the author formulated several proposals for future work (elaborated upon in the remainder of this chapter) which may potentially elevate the utility of the framework.

8.3 Suggestions for future work

This final section contains suggestions for ten avenues of further investigation as possible follow-up work on the contributions of this thesis.

Proposal I *Procure a larger data set for model training, validation, and testing.*

Although the utilisation of a pre-trained CNN model significantly reduces the amount of data required to train the prediction model (from millions of training instances to only thousands), additional data are still required for all the considered classes to be fairly represented when separated into training, validation, and testing sets. As discussed in §7.3, several enhancement procedure categories were removed from consideration as they would not have been adequately represented in the downstream processes. Acquiring more data would enable the user to consider additional enhancement procedures which were previously excluded as they were not assigned to a sufficient number of document images, rendering them inconsequential in respect of the validation and testing sets. Moreover,

the amount of available training data is especially important when the considered data image domain significantly differs from the image data on which the pre-trained model was trained, as it might be beneficial to retrain some of the pre-trained model's final layers on the new data. The domain of annotated document images differs notably from the original ImageNet data set which most pre-trained models were trained on. Therefore, if more data are obtained successfully, it might be possible to retrain more of these layers, enabling the model to produce more accurate predictions.

Proposal II *Consider the inclusion of a confidence score for the OCR evaluation metric.*

The two methods of OCR evaluation considered within this project, discussed in §3.3, included character-level and word-level accuracy. Both of these methods, however, have some accompanying drawbacks, as a character-level evaluation metric cannot be used for monetary entities, while word-level evaluation metrics consider an entire entity as incorrect even though only a single character may have been recognised incorrectly. Some modern OCR engines can provide the user with a *confidence score* for both on a character-level and/or a word-level. Internal scoring mechanisms are utilised in order to compute these confidence scores, whereafter a confidence threshold is selected for determining whether the recognised text is considered to be correct or incorrect. This internal computation may be incorporated into the engineering of the OCR evaluation metric in the first subcomponent of the InDIE framework, thereby possibly providing a more valuable evaluation metric. This, in-turn, might result in document images being assigned other enhancement procedures as their target labels (*i.e.* the implementation of Module 8.0), since the employment of a newly engineered evaluation metric might result in a new best performing enhancement procedure. Therefore, the quality of the training, validation, and testing data sets might improve, possibly resulting in a more accurate prediction model.

Proposal III *Expand and diversify the validation suite on data sets from various domains.*

The data sets considered in both the proof-of-concept instantiations in Chapter 6 and the case study in Chapter 7 varied in image composition and quality of annotations. Both data sets stem, however, from the domain of financial documents analysis and therefore comprises similar monetary GT features. Due to limited computing power and availability of acceptable data sets, however, only one proof-of-concept and one case study were implemented. In order to further support the InDIE framework's generic nature, it may be valuable to apply it to several additional data sets, comprising a greater variation of document type and accompanying annotations. Attempting to intelligently enhance newspaper article images, progress report images, memo images, journal article images, and technical drawings are all examples of possible additional use cases of the InDIE framework.

Proposal IV *Conduct a comparative study in respect of the base average OCR accuracy by employing different OCR engines for the various case studies.*

Both the proof-of-concept instantiation of Chapter 6 and the case study implementation of Chapter 7 employed only a single pre-selected OCR engine. Due to limited computing power available, the author was constrained to select a single OCR engine based on literature and the complexity of the image data at hand, rather than experimentally exploring the significance of different OCR engine implementations. While it is typically beneficial to consult the literature for additional insights and guidance, each image data set is problem-specific and may deviate from the expected norm. In order to further increase the attained OCR performance, it may be beneficial to explore the different OCR performances of multiple open-source OCR engines, as some OCR engines might yield better results in specific domains.

Proposal V *Conduct a comparative study in respect of the predictive model by employing different convolutional bases for the various case studies.*

As described in §6.2.3, the pre-trained VGG-16 CNN model may be utilised as a convolutional base for the domain of document image enhancement as the usage of the VGG-16 model showcased state-of-the-art results in the similar domain of document image classification. Although the pre-trained VGG-16 CNN model yielded positive results in respect of both the proof-of-concept and case study instantiations, not much is known regarding the capabilities of recently developed pre-trained models. Consequently, an in-depth exploration of whether the utilisation of these modern pre-trained models as convolutional bases can add value to the prediction of enhancement procedure techniques.

Proposal VI *Compare the performance of the predictive model by employing different classifier heads for the various case studies.*

The current version of the proposed InDIE framework employs a classifier comprising several dense and dropout layers. This selection was made based on the recommendations reported in the literature pertaining to transfer learning in the realm of document image classification. Although the dense layer classifier achieved positive results in both the proof-of-concept and the case study instantiations, it may prove beneficial to determine experimentally whether it is the best classifier for the domain of intelligent document image enhancement.

Proposal VII *Extend the set of considered enhancement techniques by including more complex and modernised algorithms.*

As explored in Chapter 4, there are various geometric and pixel transformations that can be applied to a document image in order to reduce the impact of image degradations. Although the current InDIE framework considers multiple document image enhancement techniques, they are mostly computationally simple and straightforward to implement. Since the focus of the project was placed on the design, development and demonstration of the practical workability of the InDIE framework, only the most popular and basic document enhancement techniques were considered. Although the implementation of the considered techniques showcased noteworthy improvements on the proof-of-concept and case study instantiations, it may be of interest to investigate the effects of more complex document enhancement techniques — the aim being to create additional and improved enhancement procedures which is expected to minimise the number of images assigned to the Basic image category (as discussed in §5.3.2).

Proposal VIII *Investigate which pre-trained convolutional base filters are most important for extracting the intrinsic patterns within the provided data sets.*

Through the successful execution of the proof-of-concept instantiation in Chapter 6, it was reported that the pre-trained VGG-16 model possesses the capability to transfer its learning experience from being trained on ImageNet to the domain of intelligent document image enhancement. Accordingly, some of the general feature filters trained for the ImageNet data set are also valuable feature filters for extracting document image features. Investigating which of these general feature filters are most important for extracting the intrinsic patterns within the provided document image data may add significant value to the research and its contributions.

Proposal IX *Applying the framework to document images with handwritten characters.*

The current implementation of the InDIE framework only considers the processing of document images comprising printed textual information. The InDIE framework was con-

structured in such a manner that it may be easily adapted to different problem settings. Instead of solely considering OCR engines that can only recognise printed text, OCR engines trained on handwritten text may be included, with the addition of new text cleaning procedures tailored for handwritten text. Accordingly, document images containing handwritten textual information may then be used as input data. The Labelling, Modelling, and Analysis subcomponents of the framework may then be used in a similar manner as it was employed in the aforementioned instantiations.

Proposal X *Applying the framework to document images captured with mobile phone cameras.*

The InDIE framework was presented and validated in this project in the context of document images captured with flatbed scanners. It is assumed that document image enhancement procedures required for images captured with mobile phone cameras are considered markedly more difficult attributable to the capturing angle (and lighting conditions). In order to further strengthen the generic nature of the InDIE framework, it would be valuable to source an appropriate image data set — captured with mobile phone cameras — and apply the InDIE framework to it. Some minor adjustments may have to be made to the InDIE framework in order to facilitate this change, including the addition of various other document enhancement techniques tailored for reducing the effects of perspective distortions.

References

- [1] ABRAHAM A, 2005, *Artificial neural networks*, John Wiley & Sons, Oklahoma (OK).
- [2] AFZAL MZ, KOLSCH A, AHMED S & LIWICKI M, 2017, *Cutting the error by half: Investigation of very deep cnn and advanced training strategies for document image classification*, Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp. 883–888.
- [3] AIT-MOHAND K, PAQUET T & RAGOT N, 2014, *Combining structure and parameter adaptation of HMMs for printed text recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **36(9)**, pp. 1716–1732.
- [4] AKHIL S, 2016, *An overview of tesseract OCR engine*, BTech Final year project, Department of Computer Science and Engineering National Institute of Technology, Calicut.
- [5] ALAMY, 2022, *Death certificate stock photos and images*, [Online], [Cited October 2022], Available from <https://www.alamy.com/stock-photo/death-certificate.html>.
- [6] ALBAWI S, MOHAMMED TA & AL-ZAWI S, 2017, *Understanding of a convolutional neural network*, Proceedings of the International Conference on Engineering and Technology, Antalya, pp. 1–6.
- [7] ALGINAHI Y, 2010, *Preprocessing techniques in character recognition*, Character Recognition, **1**, pp. 1–19.
- [8] ALMUSAWI OAR, 2018, *A survey on optical character recognition system*, Misan Journal of Academic Studies, **17(33)**.
- [9] ALZUBAIDI L, ZHANG J, HUMAIDI AJ, AL-DUJAILI A, DUAN Y, AL-SHAMMA O, SANTAMARIA J, FADHEL MA, AL-AMIDIE M & FARHAN L, 2021, *Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions*, Journal of Big Data, **8(1)**, pp. 1–74.
- [10] AL-AMEEN Z, MUTTAR A & AL-BADRANI G, 2019, *Improving the sharpness of digital image using an amended unsharp mask filter*, International Journal of Image, Graphics & Signal Processing, **11(3)**, pp. 1–9.
- [11] ANDRYCHOWICZ M, DENIL M, COLMENAREJO SG, HOFFMAN MW, PFAU D, SCHAU T, SHILLINGFORD B & DE FREITAS N, 2016, *Learning to learn by gradient descent by gradient descent*, Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, pp. 3988–3996.
- [12] ANTONIADIS P, 2022, *Activation functions: Sigmoid vs tanh*, [Online], [Cited July 2022], Available from <https://www.baeldung.com/cs/sigmoid-vs-tanh-functions>.
- [13] ARDIZZONE E, BRUNO A & MAZZOLA G, 2013, *Saliency based image cropping*, Lecture Notes in Computer Science, **8156(14)**, pp. 773–782.

- [14] ARVIND K, KUMAR J & RAMAKRISHNAN A, 2007, *Line removal and restoration of hand-written strokes*, Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, pp. 208–214.
- [15] ATALASOFT, 2019, *How to remove lines to help OCR*, [Online], [Cited July 2022], Available from <https://www.atalasoft.com/KB2/KB/50179/HOWTO-How-to-remove-lines-to-help-OCR>.
- [16] ATHIMETHPHAT M, 2011, *A review on global binarization algorithms for degraded document images*, Australian Journal of Technology, **14(3)**, pp. 188–195.
- [17] AWALGAONKAR N, BARTAKKE P & CHAUGULE R, 2021, *Automatic license plate recognition system using SSD*, Proceedings of the International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation, Goa, pp. 394–399.
- [18] AZEVEDO A & SANTOS MF, 2008, *KDD, SEMMA and CRISP-DM: A parallel overview*, Proceedings of the European Conference on Data Mining, Amsterdam, pp. 182–185.
- [19] BAEK Y, LEE B, HAN D, YUN S & LEE H, 2019, *Character region awareness for text detection*, Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach (CA), pp. 9365–9374.
- [20] BAGDANOV A & KANAI J, 1997, *Projection profile based skew estimation algorithm for JBIG compressed images*, Proceedings of the 4th International Conference on Document Analysis and Recognition, Ulm, pp. 401–405.
- [21] BAHETI P, 2022, *Twelve types of neural network activation functions*, [Online], [Cited March 2022], Available from [https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=get%20started%20%3A\)-,What%20is%20a%20Neural%20Network%20Activation%20Function%3F,prediction%20using%20simpler%20mathematical%20operations.](https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=get%20started%20%3A)-,What%20is%20a%20Neural%20Network%20Activation%20Function%3F,prediction%20using%20simpler%20mathematical%20operations.)
- [22] BENGIO Y, SIMARD P & FRASCONI P, 1994, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, **5(2)**, pp. 157–166.
- [23] BENTO C, 2021, *Stochastic gradient descent explained in real life*, [Online], [Cited March 2022], Available from <https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32#:~:text=Compared%20to%20Gradient%20Descent%2C%20Stochastic,updates%20have%20a%20higher%20variance.>
- [24] BISONG E, “Google AutoML: Cloud vision”, in: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Berkeley (CA): Apress, 2019, pp. 581–598.
- [25] BLESSER B, SHILLMAN R, KUKLINSKI T, COX C, EDEN M & VENTURA J, 1974, *A theoretical approach for character recognition based on phenomenological attributes*, International Journal of Man-Machine Studies, **6(6)**, pp. 701–714.
- [26] BLOOMBERG D, 2001, *Leptonica image processing and analysis library*, [Online], [Cited April 2022], Available from <http://www.leptonica.org/>.
- [27] BOKSER M, 1992, *Omnidocument technologies*, Proceedings of the IEEE, **80(7)**, pp. 1066–1078.
- [28] BOROVNIKOV E, 2014, *A survey of modern optical character recognition techniques*, arXiv preprint arXiv:1412.4183.
- [29] BRE F, GIMENEZ JM & FACHINOTTI VD, 2018, *Prediction of wind pressure coefficients on building surfaces using artificial neural networks*, Energy and Buildings, **158**, pp. 1429–1441.

- [30] BREUEL TM, 2017, *High performance text recognition using a hybrid convolutional-lstm implementation*, Proceedings of the 14th International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp. 11–16.
- [31] BROSAN T & SUN D, 2002, *Inspection and grading of agricultural and food products by computer vision systems — A review*, Computers and Electronics in Agriculture, **36(2-3)**, pp. 193–213.
- [32] BROWNLEE J, 2017, *Difference between classification and regression in machine learning*, [Online], [Cited October 2022], Available from <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>.
- [33] BROWNLEE J, 2019, *Understand the impact of learning rate on neural network performance*, [Online], [Cited October 2022], Available from <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:text=Specifically%2C%20the%20learning%20rate%20is,is%20adapted%20to%20the%20problem.>
- [34] BROWNRIGG DR, 1984, *The weighted median filter*, Communications of the ACM, **27(8)**, pp. 807–818.
- [35] BULAT A & TZIMIROPOULOS G, 2016, *Human pose estimation via convolutional part heatmap regression*, Proceedings of the European Conference on Computer Vision, Amsterdam, pp. 717–732.
- [36] BULLINARIA JA, 2013, *Recurrent neural networks*, Neural Computation: Lecture, **12**, pp. 1–20.
- [37] BUSINESSTECH, 2020, *South Africa's five biggest banks compared: earnings vs customers vs market cap vs reach*, [Online], [Cited March 2021], Available from <https://businesstech.co.za/news/banking/431754/south-africas-5-biggest-banks-compared-earnings-vs-customers-vs-market-cap-vs-reach/>.
- [38] CAMBRIDGE IN COLOUR, 2020, *Sharpening: Unsharp mask*, [Online], [Cited July 2022], Available from <https://www.cambridgeincolour.com/tutorials/unsharp-mask.htm>.
- [39] CAMPBELL M, HOANE AJ & HSU F, 2002, *Deep blue*, Artificial intelligence, **134(1-2)**, pp. 57–83.
- [40] CAPITEC, 2022, *How Capitec became SA's biggest digital bank*, [Online], [Cited September 2022], Available from <https://www.capitecbank.co.za/blog/articles/experiences/how-capitec-became-sas-biggest-digital-bank/#:~:text=You%20have%20to%20cater%20to,Capitec%20branches%20across%20South%20Africa.>
- [41] CAPITEC, 2022, *Personal loan*, [Online], [Cited October 2022], Available from <https://www.capitecbank.co.za/personal/credit/personal-loan/>.
- [42] CAPITEC, 2020, *The credit boot camp part 1: The basics of credit*, [Online], [Cited March 2021], Available from <https://www.capitecbank.co.za/bank-better-live-better/articles/good-for-credit/understanding-the-basics-of-credit-is-important/>.
- [43] CH'NG CK & CHAN CS, 2017, *Total-text: A comprehensive dataset for scene text detection and recognition*, Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp. 935–942.
- [44] CHAI T & DRAXLER RR, 2014, *Root mean square error or mean absolute error*, Geoscientific Model Development Discussions, **7(1)**, pp. 1525–1534.

- [45] CHAKI J & DEY N, 2018, *A beginner's guide to image preprocessing techniques*, CRC Press, Boca Raton (FL).
- [46] CHANDRASEKAR R, 2014, *Elementary? Question answering, IBM's Watson, and the Jeopardy! challenge*, *Resonance*, **19(3)**, pp. 222–241.
- [47] CHAUDHURI A & CHAUDHURI S, 1997, *Robust detection of skew in document images*, *IEEE Transactions on Image Processing*, **6(2)**, pp. 344–349.
- [48] CHAUHAN S, SHARMA E & DOEGAR A, 2016, *Binarization techniques for degraded document images — A review*, *Proceedings of the 5th International Conference on Reliability, Infocom Technologies and Optimization, Noida*, pp. 163–166.
- [49] CHICCO D, 2017, *Ten quick tips for machine learning in computational biology*, *BioData Mining*, **10(1)**, pp. 1–17.
- [50] CHOWDHURY A, KAUTZ E, YENER B & LEWIS D, 2016, *Image driven machine learning methods for microstructure recognition*, *Computational Materials Science*, **123**, pp. 176–187.
- [51] CIARDIELLO G, SCAFURO G, DEGRANDI M, SPADA M & ROCCOTELLI M, 1988, *An experimental system for office document handling and text recognition*, *Proceedings of the 9th International Conference on Pattern Recognition, Cambridge*, pp. 739–743.
- [52] CLAUSNER C, ANTONACOPOULOS A & SCHACHER S, 2020, *Efficient and effective OCR engine training*, *International Journal on Document Analysis and Recognition (IJDAR)*, **23(1)**, pp. 73–88.
- [53] CLAUSNER C, PLETSCHACHER S & ANTONACOPOULOS A, 2020, *Flexible character accuracy measure for reading-order-independent evaluation*, *Pattern Recognition Letters*, **131**, pp. 390–397.
- [54] CORREA IL, DREWS PLJ & RODRIGUES RN, 2021, *Combination of optical character recognition engines for documents containing sparse text and alphanumeric codes*, *Proceedings of the 34th Conference on Graphics, Patterns and Images (SIBGRAPI), Rio de Janeiro*, pp. 299–306.
- [55] CORTÉS E & SÁNCHEZ S, 2021, *Deep learning transfer with AlexNet for chest X-ray COVID-19 recognition*, *IEEE Latin America Transactions*, **19(6)**, pp. 944–951.
- [56] CRAWFORD M, KHOSHGOFTAAR TM, PRUSA JD, RICHTER AN & AL NAJADA H, 2015, *Survey of review spam detection using machine learning techniques*, *Journal of Big Data*, **2(1)**, pp. 1–24.
- [57] CUNNINGHAM P, CORD M & DELANY SJ, “Supervised learning”, in: *Machine learning techniques for multimedia*, Heidelberg: Springer, 2008, pp. 21–49.
- [58] DA SILVA IN, SPATTI DH, FLAUZINO RA, LIBONI LHB & DOS REIS ALVES SF, 2017, *Artificial neural networks*, Cham: Springer International Publishing, **39**.
- [59] DANIELSSON P.-E, 1980, *Euclidean distance mapping*, *Computer Graphics and Image Processing*, **14(3)**, pp. 227–248.
- [60] DAS A, ROY S, BHATTACHARYA U & PARUI SK, 2018, *Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks*, *Proceedings of the 24th International Conference on Pattern Recognition (ICPR), Sydney*, pp. 3180–3185.
- [61] DENG G & CAHILL L, 1993, *An adaptive Gaussian filter for noise reduction and edge detection*, *Proceedings of the 4th Conference Record Nuclear Science Symposium and Medical Imaging Conference, San Francisco (CA)*, pp. 1615–1619.

- [62] DENG J, DONG W, SOCHER R, LI L, LI K & FEI-FEI L, 2009, *Imagenet: A large-scale hierarchical image database*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami (FL), pp. 248–255.
- [63] DEY R, BALABANTARAY RC, MOHANTY S, SINGH D, KARUPPIAH M & SAMANTA D, 2022, *Approach for preprocessing in offline optical character recognition (OCR)*, Proceedings of the Interdisciplinary Research in Technology and Management (IRTM), Kolkata, pp. 1–6.
- [64] DICKSON B, 2019, *What are artificial neural networks (ANN)?*, [Online], [Cited March 2021], Available from <https://bdtechtalks.com/2019/08/05/what-is-artificial-neural-network-ann/>.
- [65] DOCPARSER, 2018, *Improve OCR accuracy with advanced image preprocessing*, [Online], [Cited June 2022], Available from <https://docparser.com/blog/improve-ocr-accuracy/>.
- [66] DONGES N, 2020, *Gradient descent in machine learning: A basic introduction*, [Online], [Cited October 2022], Available from <https://builtin.com/data-science/gradient-descent>.
- [67] DONGRE VJ & MANKAR VH, *A review of research on devnagari character recognition*, International Journal of Computer Applications, **975**, pp. 8887–8888.
- [68] DUDA RO & HART PE, 1972, *Use of the Hough transformation to detect lines and curves in pictures*, Communications of the ACM, **15(1)**, pp. 11–15.
- [69] DURIAN L, 2022, *Personal finance*, [Online], [Cited September 2021], Available from https://www.reddit.com/r/personalfinance/comments/p39z1j/please_help_me_understand_what_all_this_stuff_on/.
- [70] DYKSTRA EW, 2017, *Image inarisation: Introduction*, [Online], [Cited July 2022], Available from <https://craftofcoding.wordpress.com/2017/02/13/image-binarization-1-introduction/>.
- [71] ELHEDDA W, MEHRI M & MAHJOUB MA, 2017, *A comparative study of filtering approaches applied to color archival document images*, Proceedings of the International Arab Conference on Information Technology, Yasmine Hammamet.
- [72] ELSHEIKH A, YACOUT S & OUALI M, 2019, *Bidirectional handshaking LSTM for remaining useful life prediction*, Neurocomputing, **323**, pp. 148–156.
- [73] ELSKEN T, METZEN JH & HUTTER F, 2019, *Neural architecture search: A survey*, The Journal of Machine Learning Research, **20(1)**, pp. 1997–2017.
- [74] ERKAN U, GÖKREM L & ENGINOĞLU S, 2018, *Different applied median filter in salt and pepper noise*, Computers & Electrical Engineering, **70**, pp. 789–798.
- [75] FANG S, XIE H, WANG Y, MAO Z & ZHANG Y, 2021, *Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville (TS), pp. 7098–7107.
- [76] FARAHMAND A, SARRAFZADEH H & SHANBEHZADEH J, 2013, *Document image noises and removal methods*, Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, pp. 436–440.
- [77] FINLAYSON G, HORDLEY S, SCHAEFER G & TIAN GY, 2005, *Illuminant and device invariant colour using histogram equalisation*, Pattern Recognition, **38(2)**, pp. 179–190.

- [78] FISCHER T & KRAUSS C, 2018, *Deep learning with long short-term memory networks for financial market predictions*, European Journal of Operational Research, **270(2)**, pp. 654–669.
- [79] FIVEKO, 2022, *Gaussian blur — Noise reduction filter in image processing*, [Online], [Cited July 2022], Available from <https://fiveko.com/gaussian-blur-filter/>.
- [80] GAO H, MAO J, ZHOU J, HUANG Z, WANG L & XU W, 2015, *Are you talking to a machine? Dataset and methods for multilingual image question*, Advances in Neural Information Processing Systems, **28**, pp. 1–8.
- [81] GAO L, HUANG Y, DEJEAN H, MEUNIER J, YAN Q, FANG Y, KLEBER F & LANG E, 2019, *ICDAR 2019 competition on table detection and recognition (CTDAR)*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1510–1515.
- [82] GARDNER MW & DORLING S, 1998, *Artificial neural networks (the multilayer perceptron) — A review of applications in the atmospheric sciences*, Atmospheric Environment, **32(14-15)**, pp. 2627–2636.
- [83] GATOS B, DANATSAS D, PRATIKAKIS I & PERANTONIS SJ, 2005, *Automatic table detection in document images*, Proceedings of the International Conference on Pattern Recognition and Image Analysis, Bath, pp. 609–618.
- [84] GERS FA & SCHMIDHUBER J, 2000, *Recurrent nets that time and count*, Proceedings of the International Joint Conference on Neural Networks, Como, pp. 189–194.
- [85] GERS FA, SCHMIDHUBER J & CUMMINS F, 2000, *Learning to forget: Continual prediction with LSTM*, Neural Computation, **12(10)**, pp. 2451–2471.
- [86] GHAHRAMANI Z, 2003, *Unsupervised learning*, Proceedings of the Summer School on Machine Learning, Canberra, pp. 72–112.
- [87] GHARDE SS, BAVISKAR PV & ADHIYA K, 2013, *Identification of handwritten simple mathematical equation based on svm and projection histogram*, International Journal of Soft Computing and Engineering (IJSCE), **3(2)**, pp. 425–429.
- [88] GOMEZ L & KARATZAS D, 2016, *A fine-grained approach to scene text script identification*, Proceedings of the 12th IAPR workshop on document analysis systems (DAS), Santorini, pp. 192–197.
- [89] GONÇALVES GR, DINIZ MA, LAROCA R, MENOTTI D & SCHWARTZ WR, 2018, *Real-time automatic license plate recognition through deep multi-task networks*, Proceedings of the 31th Conference on Graphics, Patterns and Images (SIBGRAPI), Parana, pp. 110–117.
- [90] GOODFELLOW I, BENGIO Y & COURVILLE A, 2016, *Deep learning*, MIT Press, Cambridge (MA).
- [91] GOOGLE, 2022, *Google colaboraroty*, [Online], [Cited September 2022], Available from <https://colab.research.google.com/>.
- [92] GOOGLE, 2006, *TesseractOCR*, [Online], [Cited June 2022], Available from <https://github.com/tesseract-ocr/>.
- [93] GOOGLECODE, 1996, *ISRI OCR evaluation tools*, [Online], [Cited June 2022], Available from <https://code.google.com/archive/p/isri-ocr-evaluation-tools/>.
- [94] GOOGLECODE, 1999, *Text digitisation*, [Online], [Cited June 2022], Available from <https://sites.google.com/site/textdigitisation/qualitymeasures/computingerrrates>.

- [95] GoSHOPPI, 2022, *Convolutional neural networks*, [Online], [Cited May 2022], Available from <https://www.sneakernews36.top/ProductDetail.aspx?iid=377204068&pr=39.88>.
- [96] GOSWAMI R & SHARMA O, 2013, *A review on character recognition techniques*, International Journal of Computer Applications, **83(7)**, pp. 18–23.
- [97] GOUVIAS D, 2022, Current Senior Data Scientist at Capitec Bank, [Personal Communication], Contactable at davidgouvias@capitecbank.co.za.
- [98] GOYAL M, 2011, *Morphological image processing*, International Journal of Computer Science Trends and Technology, **2(4)**, pp. 59–60.
- [99] GRAVES A, “Supervised sequence labelling”, in: *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, 2012, pp. 5–13.
- [100] GRAVES A, ECK D, BERINGER N & SCHMIDHUBER J, 2004, *Biologically plausible speech recognition with LSTM neural nets*, Proceedings of the International Workshop on Biologically Inspired Approaches to Advanced Information Technology, Lausanne, pp. 127–136.
- [101] GRAVES A, FERNÁNDEZ S, GOMEZ F & SCHMIDHUBER J, 2006, *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*, Proceedings of the 23th International Conference on Machine Learning, Pittsburgh (PA), pp. 369–376.
- [102] GRAVES A, MOHAMED A & HINTON G, 2013, *Speech recognition with deep recurrent neural networks*, Proceedings of the 38th International Conference on Acoustics, Speech and Signal Processing, Vancouver, pp. 6645–6649.
- [103] GRAVES A & SCHMIDHUBER J, 2005, *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*, Neural Networks, **18(5-6)**, pp. 602–610.
- [104] GREAT LEARNING TEAM, 2020, *Introduction to image pre-processing*, [Online], [Cited June 2022], Available from <https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/#IntroductiontoImagePre-Processing>.
- [105] GREFF K, SRIVASTAVA RK, KOUTNIK J, STEUNEBRINK BR & SCHMIDHUBER J, 2016, *LSTM: A search space odyssey*, Transactions on Neural Networks and Learning Systems, **28(10)**, pp. 2222–2232.
- [106] GUO Y, LIU Y, OERLEMANS A, LAO S, WU S & LEW MS, 2016, *Deep learning for visual understanding: A review*, Neurocomputing, **187**, pp. 27–48.
- [107] GUPTA A, 2019, *A comprehensive guide on deep learning optimizers*, [Online], [Cited October 2022], Available from <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=An%20optimizer%20is%20a%20function,loss%20and%20improve%20the%20accuracy.>
- [108] GUPTA S, 2021, *Regression versus classification in machine learning: What is the difference*, [Online], [Cited October 2022], Available from <https://www.springboard.com/blog/data-science/regression-vs-classification/#:~:text=The%20most%20significant%20difference%20between,types%20of%20machine%20learning%20algorithms.>
- [109] GUYON I, HARALICK RM, HULL JJ & PHILLIPS IT, “Data sets for OCR and document image understanding research”, in: *Handbook of Character Recognition and Document Image Analysis*, World Scientific, 1997, pp. 779–799.
- [110] HAENLEIN M & KAPLAN A, 2019, *A brief history of artificial intelligence: On the past, present, and future of artificial intelligence*, California Management Review, **61(4)**, pp. 5–14.

- [111] HAMAD KA & MEHMET K, 2016, *A detailed analysis of optical character recognition technology*, International Journal of Applied Mathematics Electronics and Computers, pp. 244–249.
- [112] HAN KTM & UYYANONVARA B, 2016, *A survey of blob detection algorithms for biomedical images*, Proceedings of the 7th International Conference of Information and Communication Technology for Embedded Systems, Chennai, pp. 57–60.
- [113] HARLEY AW, UFKES A & DERPANIS KG, 2015, *Evaluation of deep convolutional nets for document image classification and retrieval*, Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR), Tunis, pp. 991–995.
- [114] HASAN MS, 2017, *An application of pre-trained CNN for image classification*, Proceedings of the 20th International Conference of Computer and Information Technology, Dhaka, pp. 1–6.
- [115] HASHIZUME A, YEH P.-S & ROSENFELD A, 1986, *A method of detecting the orientation of aligned components*, Pattern Recognition Letters, **4(2)**, pp. 125–132.
- [116] HASSOUN MH, 1995, *Fundamentals of artificial neural networks*, MIT Press, Cambridge (MA).
- [117] HAZIRBAS C, MA L, DOMOKOS C & CREMERS D, 2016, *Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture*, Proceedings of the 13th Asian Conference on Computer Vision, Perth, pp. 213–228.
- [118] HE K, ZHANG X, REN S & SUN J, 2016, *Deep residual learning for image recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas (NV), pp. 770–778.
- [119] HE Y, LIN J, LIU Z, WANG H, LI L.-J & HAN S, 2018, *Amc: Automl for model compression and acceleration on mobile devices*, Proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, pp. 784–800.
- [120] HEIJMANS HJ, 1999, *Connected morphological operators for binary images*, Computer Vision and Image Understanding, **73(1)**, pp. 99–120.
- [121] HINDS SC, FISHER JL & D’AMATO DP, 1990, *A document skew detection method using run-length encoding and the Hough transform*, Proceedings of the 10th International Conference on Pattern Recognition, Atlantic City (NJ), pp. 464–468.
- [122] HO Y & WOOKEY S, 2019, *The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling*, IEEE Access, **8**, pp. 4806–4813.
- [123] HOCHREITER S, BENGIO Y, FRASCONI P & SCHMIDHUBER J, 2001, *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies*.
- [124] HOCHREITER S & SCHMIDHUBER J, 1997, *Long short-term memory*, Neural Computation, **9(8)**, pp. 1735–1780.
- [125] HOLLEY R, 2009, *How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs*, D-Lib Magazine, **15(3/4)**, pp. 1–13.
- [126] HUANG T, YANG G & TANG G, 1979, *A fast two-dimensional median filtering algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing, **27(1)**, pp. 13–18.
- [127] HUANG Y, CHENG Y, BAPNA A, FIRAT O, CHEN D, CHEN M, LEE H, NGIAM J, LE QV & WU Y, 2019, *Gpipe: Efficient training of giant neural networks using pipeline parallelism*, Advances in Neural Information Processing Systems, **32**, pp. 1–10.

- [128] HUANG Z, CHEN K, HE J, BAI X, KARATZAS D, LU S & JAWAHAR C, 2019, *Competition on scanned receipt ocr and information extraction*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1516–1520.
- [129] HUANG Z, XU W & YU K, 2015, *Bidirectional LSTM-CRF models for sequence tagging*, arXiv preprint arXiv:1508.01991.
- [130] HUANG Z, JIA X & GUO Y, 2019, *State-of-the-art model for music object recognition with deep learning*, Applied Sciences, **9(13)**, pp. 2645.
- [131] HUSSEIN RS, ELKHIDIR AA & ELNOURANI MG, 2015, *Optical character recognition of arabic handwritten characters using neural network*, Proceedings of the International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), Khartoum, pp. 456–461.
- [132] HUVAL B, WANG T, TANDON S, KISKE J, SONG W, PAZHAYAMPALLIL J, ANDRILUKA M, RAJPURKAR P, MIGIMATSU T & CHENG-YUE R, 2015, *An empirical evaluation of deep learning on highway driving*, arXiv preprint arXiv:1504.01716.
- [133] ICDAR, 2019, *The international conference on document analysis and recognition*, [Online], [Cited September 2022], Available from <https://icdar2019.org/competitions-2/>.
- [134] ICDAR, 2021, *The international conference on document analysis and recognition*, [Online], [Cited September 2022], Available from <https://www.icdar.org/>.
- [135] ILLINGWORTH J & KITTLER J, 1988, *A survey of the Hough transform*, Computer Vision, Graphics, and Image Processing, **44(1)**, pp. 87–116.
- [136] IMPEDOVO D & PIRLO G, 2014, *Zoning methods for handwritten character recognition: A survey*, Pattern Recognition, **47(3)**, pp. 969–981.
- [137] IOFFE S & SZEGEDY C, 2015, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, Proceedings of the 32th International Conference on Machine Learning, Lille, pp. 448–456.
- [138] ISHITANI Y, 1993, *Document skew detection based on local region complexity*, Proceedings of the 2th International Conference on Document Analysis and Recognition (ICDAR), Tsukuba, pp. 49–52.
- [139] ISMAIL FAWAZ H, LUCAS B, FORESTIER G, PELLETIER C, SCHMIDT DF, WEBER J, WEBB GI, IDOUMGHAR L, MULLER P.-A & PETITJEAN F, 2020, *Inceptiontime: Finding alexnet for time series classification*, Data Mining and Knowledge Discovery, **34(6)**, pp. 1936–1962.
- [140] IVAN R, 2017, *What is machine learning?*, [Online], [Cited March 2021], Available from <https://www.cognizantsoftvision.com/blog/what-is-machine-learning/>.
- [141] JAIDENAI, 2020, *EasyOCR*, [Online], [Cited June 2022], Available from <https://github.com/JaidedAI/EasyOCR>.
- [142] JAVATPOINT, 2020, *Gradient descent in machine learning*, [Online], [Cited October 2022], Available from <https://www.javatpoint.com/gradient-descent-in-machine-learning>.
- [143] JIANG X, BUNKE H & WIDMER-KLJAJO D, 1999, *Skew detection of document images by focused nearest-neighbor clustering*, Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR), Bangalore, pp. 629–632.

- [144] JIMENO YEPES A, ZHONG P & BURDICK D, 2021, *ICDAR 2021 competition on scientific literature parsing*, Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR), Lausanne, pp. 605–617.
- [145] KAFLE K & KANAN C, 2017, *Visual question answering: Datasets, algorithms, and future challenges*, Computer Vision and Image Understanding, **163**, pp. 3–20.
- [146] KAHAN S, PAVLIDIS T & BAIRD HS, 1987, *On the recognition of printed characters of any font and size*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **(2)**, pp. 274–288.
- [147] KANJO E, YOUNIS EM & ANG CS, 2019, *Deep learning analysis of mobile physiological, environmental and location sensor data for emotion detection*, Information Fusion, **49**, pp. 46–56.
- [148] KARPATY A, 2015, *The unreasonable effectiveness of recurrent neural networks*, [Online], [Cited March 2022], Available from <http://karpathy.github.io/2015/05/21/rn-effectiveness/>.
- [149] KAUR S & KAUR M, 2018, *Image sharpening using basic enhancement techniques*, International Journal of Research, Engineering, Science, and Management, **1(12)**, pp. 122–126.
- [150] KENDALL K & KENDALL J, 2013, *Systems analysis and design*, 9th Edition, Pearson, Upper Saddle River (NJ).
- [151] KHAN A, SOHAIL A, ZAHOORA U & QURESHI AS, 2020, *A survey of the recent architectures of deep convolutional neural networks*, Artificial Intelligence Review, **53(8)**, pp. 5455–5516.
- [152] KHIRIRAT S, FEYZMAHDAVIAN HR & JOHANSSON M, 2017, *Mini-batch gradient descent: Faster convergence under data sparsity*, Proceedings of the 56th Annual Conference on Decision and Control (CDC), Melbourne, pp. 2880–2887.
- [153] KIM G, GOVINDARAJU V & SRIHARI SN, 1999, *An architecture for handwritten text recognition systems*, International Journal on Document Analysis and Recognition, **2(1)**, pp. 37–44.
- [154] KIM P, “Convolutional neural network”, in: *MATLAB Deep Learning*, Springer, 2017, pp. 121–147.
- [155] KINGMA DP & BA J, 2014, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980.
- [156] KIRYATI N, ELGAR Y & BRUCKSTEIN AM, 1991, *A probabilistic Hough transform*, Pattern Recognition, **24(4)**, pp. 303–316.
- [157] KOCHALE M TSA, KHEMARIYA A & TIWARI MA, *Real-time automatic vehicle (license) recognition identification system with the Help of Opencv & Easyocr model*, International Journal of Research, Science, Technology, and Management, **23(3)**, pp. 11–15.
- [158] KOIDL K, 2013, *Loss functions in classification tasks*, School of Computer Science and Statistic Trinity College, Dublin, pp. 1–5.
- [159] KONONENKO I, 2001, *Machine learning for medical diagnosis: History, state of the art and perspective*, Artificial Intelligence in Medicine, **23(1)**, pp. 89–109.
- [160] KRENKER A, BEŠTER J & KOS A, 2011, *Introduction to the artificial neural networks*, Artificial Neural Networks: Methodological Advances and Biomedical Applications, pp. 1–18.

- [161] KRISHNASWAMY RANGARAJAN A & PURUSHOTHAMAN R, 2020, *Disease classification in eggplant using pre-trained VGG16 and MSVM*, Scientific Reports, **10(1)**, pp. 1–11.
- [162] KRIZHEVSKY A, SUTSKEVER I & HINTON GE, 2012, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems, **25**, pp. 1097–1105.
- [163] KSHETRY RL, 2021, *Image preprocessing and modified adaptive thresholding for improving OCR*, arXiv preprint arXiv:2111.14075.
- [164] KUMAR H & SHIVAKUMARA P, 2007, *Skew detection technique for binary document images based on hough transform*, International Journal of Computer and Information Engineering, **1(8)**, pp. 2401–2407.
- [165] LA MANNA S, COLIA A & SPERDUTI A, 1999, *Optical font recognition for multi-font OCR and document processing*, Proceedings of the 10th International Workshop on Database and Expert Systems Applications, Vienna, pp. 549–553.
- [166] LAMPROPOULOS AS & TSIHRINTZIS GA, 2015, *Machine learning paradigms*, Applications in Recommender Systems, pp. 31–55.
- [167] LAROCA R, SEVERO E, ZANLORENSI LA, OLIVEIRA LS, GONÇALVES GR, SCHWARTZ WR & MENOTTI D, 2018, *A robust real-time automatic license plate recognition based on the YOLO detector*, Proceedings of the International Joint Conference on Neural Networks, Rio de Janeiro, pp. 1–10.
- [168] LECUN Y & BENGIO Y, 1995, *Convolutional networks for images, speech, and time series*, The Handbook of Brain Theory and Neural Networks, **3361(10)**, pp. 1995.
- [169] LECUN Y, BOTTOU L, BENGIO Y & HAFFNER P, 1998, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, **86(11)**, pp. 2278–2324.
- [170] LEVENSHTAIN VI, 1966, *Binary codes capable of correcting deletions, insertions, and reversals*, Proceedings of the 8th Soviet Physics Doklady, Oakland (CA), pp. 707–710.
- [171] LI J & LIU D, 2021, *Information bottleneck theory on convolutional neural networks*, Neural Processing Letters, **53(2)**, pp. 1385–1400.
- [172] LI Z, LIU F, YANG W, PENG S & ZHOU J, 2021, *A survey of convolutional neural networks: analysis, applications, and prospects*, IEEE Transactions on Neural Networks and Learning Systems, pp. 1–15.
- [173] LIN M, CHEN Q & YAN S, 2013, *Network in network*, arXiv preprint arXiv:1312.4400.
- [174] LIPTON ZC, BERKOWITZ J & ELKAN C, 2015, *A critical review of recurrent neural networks for sequence learning*, arXiv preprint arXiv:1506.00019.
- [175] LIU H, JIN S & ZHANG C, 2018, *Connectionist temporal classification with maximum entropy regularization*, Proceedings of the 32th International Conference on Neural Information Processing Systems, Montreal, pp. 839–849.
- [176] LU S, LU Z & ZHANG Y, 2019, *Pathological brain detection based on AlexNet and transfer learning*, Journal of Computational Science, **30**, pp. 41–47.
- [177] MAGESHWARAN R, 2021, *Review of best open-source OCR tools*, [Online], [Cited June 2022], Available from <https://medium.com/technovators/review-of-best-open-source-ocr-tools-fc839a20e61f>.
- [178] MAHDAVI M, ZANIBBI R, MOUCHERE H, VIARD-GAUDIN C & GARAIN U, 2019, *ICDAR 2019 CROHME TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1533–1538.

- [179] MARCELINO P, 2018, *Transfer learning from pre-trained models*, [Online], [Cited May 2022], Available from <https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751#:~:text=Transfer%20learning%20is%20a%20popular,when%20solving%20a%20different%20problem>.
- [180] AL-MASRI A, 2019, *How does back-propagation in artificial neural networks work*, [Online], [Cited March 2022], Available from <https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7>.
- [181] MEHRI M, HEROUX P, MULLOT R, MOREUX J.-P, COUASNON B & BARRETT B, 2019, *ICDAR 2019 competition on historical book analysis-hba2019*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1488–1493.
- [182] MEHROTRA K, MOHAN CK & RANKA S, 1997, *Elements of artificial neural networks*, MIT Press, Cambridge (MA).
- [183] MEYER MH & WEBB PH, 2005, *Modular, layered architecture: The necessary foundation for effective mass customisation in software*, International Journal of Mass Customisation, **1(1)**, pp. 14–36.
- [184] MICHALAK H & OKARMA K, 2020, *Robust combined binarization method of non-uniformly illuminated document images for alphanumerical character recognition*, Sensors, **20(10)**, pp. 2914–2915.
- [185] MIN S, LEE B & YOON S, 2017, *Deep learning in bioinformatics*, Briefings in Bioinformatics, **18(5)**, pp. 851–869.
- [186] MISHRA D, 2020, *Translational invariance vs translational equivariance*, [Online], [Cited April 2022], Available from <https://towardsdatascience.com/translational-invariance-vs-translational-equivariance-f9fbc8fca63a>.
- [187] MITCHELL TM, 1997, *Artificial neural networks*, Machine Learning, **45**, pp. 81–127.
- [188] MITHE R, INDALKAR S & DIVEKAR N, 2013, *Optical character recognition*, International journal of Recent Technology and Engineering (IJRTE), **2(1)**, pp. 72–75.
- [189] MITTAL R & GARG A, 2020, *Text extraction using OCR: A systematic review*, Proceedings of the 2th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, pp. 357–362.
- [190] MORAVEC K, 2002, *A grayscale reader for camera images of xerox data glyphs*, Proceedings of the 13th British Machine Vision Conference, Cardiff, pp. 1–10.
- [191] MOTA JS, OKIMOTO MV, CANEDO ED & MOTA JS, 2021, *Google summer of code gender diversity: An analysis of the last 4 editions*, Computer on the Beach, **12**, pp. 117–124.
- [192] MURZOVA A, 2020, *Otsu's thresholding with OpenCV*, [Online], [Cited July 2022], Available from <https://learnopencv.com/otsu-thresholding-with-opencv/>.
- [193] MUTHUKRISHNAN, 2020, *Skew detection and correction of document images using Hough transform*, [Online], [Cited June 2022], Available from <https://muthu.co/skew-detection-and-correction-of-document-images-using-hough-transform/>.
- [194] NAGY G, NARTKER TA & RICE SV, 1999, *Optical character recognition: An illustrated guide to the frontier*, Proceedings of the 7th Document Recognition and Retrieval, San José (CA), pp. 58–69.
- [195] NAIR V & HINTON GE, 2010, *Rectified linear units improve restricted boltzmann machines*, Proceedings of the 27th International Conference on Machine Learning, Haifa, pp. 807–814.

- [196] NAJAFIRAGHEB N, UNIVERSITY H, BANDARE-E-ABBAS I, HATAM A & HARIFI A, 2016, *A survey of feature extraction techniques in OCR*, Proceedings of the International Conference on New Research Achievements in Electrical and Computer Engineering.
- [197] NATHANCY, 2016, *How to remove all lines and borders in an image while keeping text programmatically?*, [Online], [Cited July 2022], Available from <https://stackoverflow.com/questions/33949831/how-to-remove-all-lines-and-borders-in-an-image-while-keeping-text-programmatica>.
- [198] NAZ S, UMAR AI, AHMAD R, AHMED SB, SHIRAZI SH, SIDDIQI I & RAZZAK MI, 2016, *Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks*, Neurocomputing, **177**, pp. 228–241.
- [199] NEAL RM, 1992, *Connectionist learning of belief networks*, Artificial Intelligence, **56(1)**, pp. 71–113.
- [200] NEUDECKER C, BAIERER K, GERBER M, CHRISTIAN C, APOSTOLOS A & STEFAN P, 2021, *A survey of OCR evaluation tools and metrics*, Proceedings of the 6th International Workshop on Historical Document Imaging and Processing, Lausanne, pp. 13–18.
- [201] NEWELL A, SHAW JC & SIMON HA, 1959, *Report on a general problem solving program*, Proceedings of the International Federation for Information Processing, Paris, p. 64.
- [202] NG A, 2013, *Machine learning and AI via brain simulations*, [Online], [Cited July 2022], Available from <http://datascienceassn.org/sites/default/files/Machine%20Learning%20and%20AI%20via%20Brain%20Simulations.pdf>.
- [203] NIBLACK W, 1985, *An introduction to digital image processing*, Strandberg Publishing Company, Copenhagen.
- [204] NUGRAHA BT & SU S.-F, 2017, *Towards self-driving car using convolutional neural network and road lane detector*, Proceedings of the 2th International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology, Jakarta, pp. 65–69.
- [205] NWANKPA CE, IJOMAH W, GACHAGAN A & MARSHALL S, 2021, *Activation functions: Comparison of trends in practice and research for deep learning*, Proceedings of the 6th International Conference on Computational Sciences and Technology, Online, pp. 124–133.
- [206] O’GORMAN L, 1993, *The document spectrum for page layout analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **15(11)**, pp. 1162–1173.
- [207] OLIVIER J, 2022, Current Senior Machine Learning Engineer at Capitec Bank, [Personal Communication], Contactable at johanolivier@capitecbank.co.za.
- [208] OPENCV, 2022, *Extract horizontal and vertical lines by using morphological operations*, [Online], [Cited July 2022], Available from https://docs.opencv.org/4.x/dd/dd7/tutorial_morph_lines_detection.html.
- [209] OTSU N, 1979, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man, and Cybernetics, **9(1)**, pp. 62–66.
- [210] PANT P, 2019, *CNN: Understanding edge detection with an example*, [Online], [Cited March 2022], Available from <https://pradeeppant.com/2019/11/21/cnn-understanding-edge-detection-with-an-example.html>.
- [211] PAPERS WITH CODE, 2022, *Document image classification on RVL-CDIP*, [Online], [Cited September 2022], Available from <https://paperswithcode.com/sota/document-image-classification-on-rvl-cdip>.

- [212] PASZKE A, GROSS S, MASSA F, LERER A, BRADBURY J, CHANAN G, KILLEEN T, LIN Z, GIMELSHEIN N & ANTIGA L, 2019, *Pytorch: An imperative style, high-performance deep learning library*, Advances in Neural Information Processing systems, **33**, pp. 8026–8037.
- [213] PEREZ S, 2012, *Uber simplifies sign up process: Just hold up your credit card & you're in*, [Online], [Cited March 2021], Available from <https://techcrunch.com/2012/04/04/uber-speeds-up-sign-up-process-just-hold-up-your-credit-card-youre-in/>.
- [214] PERONA P & MALIK J, 1990, *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **12(7)**, pp. 629–639.
- [215] PHILLIPS IT, CHEN S, HA J & HARALICK RM, 1993, *English document database design and implementation methodology*, Proceedings of the Annual Symposium on Document Analysis and Retrieval, Las Vegas (NV), pp. 65–104.
- [216] PLETSCHACHER S & ANTONACOPOULOS A, 2010, *The page (page analysis and ground-truth elements) format framework*, Proceedings of the 20th International Conference on Pattern Recognition, Istanbul, pp. 257–260.
- [217] RANI S, CS VS & VASUDEV T, 2016, *An unsupervised classification technique for detection of flipped orientations in document images*, International Journal of Electrical and Computer Engineering, **6(5)**, pp. 2140.
- [218] RANJAN A, BEHERA VNJ & REZA M, “OCR using computer vision and machine learning”, in: *Machine Learning Algorithms for Industrial Applications*, Springer, 2021, pp. 83–105.
- [219] RASAMOELINA AD, ADJAILIA F & SINČÁK P, 2020, *A review of activation function for artificial neural network*, Proceedings of the 18th World Symposium on Applied Machine Intelligence and Informatics (SAMII), Harlany, pp. 281–286.
- [220] REFAEY MA, 2015, *Ruled lines detection and removal in grey level handwritten image documents*, Proceedings of the 6th International Conference on Information and Communication Systems (ICICS), Amman, pp. 218–221.
- [221] REHMAN A, KURNIAWAN F & SABA T, 2011, *An automatic approach for line detection and removal without smash-up characters*, The Imaging Science Journal, **59(3)**, pp. 177–182.
- [222] REHMAN A & SABA T, 2011, *Document skew estimation and correction: analysis of techniques, common problems and possible solutions*, Applied Artificial Intelligence, **25(9)**, pp. 769–787.
- [223] RICE SV, JENKINS FR & NARTKER TA, 1995, *The fourth annual test of OCR accuracy*, (Unpublished) Technical Report 95.
- [224] RICE SV, 1996, *Measuring the accuracy of page-reading systems*, University of Nevada, Las Vegas (NV).
- [225] RICHTER M, 2019, *Digitize your receipts using computer vision*, [Online], [Cited October 2022], Available from <https://www.inovex.de/de/blog/digitize-receipts-computer-vision/>.
- [226] RIGAUD C, DOUCET A, COUSTATY M & MOREUX J.-P, 2019, *ICDAR 2019 competition on post-OCR text correction*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1588–1593.
- [227] ROBUST READING COMPETITION, 2019, *Overview — ICDAR 2019 Robust reading challenge on scanned receipts OCR and information extraction*, [Online], [Cited September 2022], Available from <https://rrc.cvc.uab.es/?ch=13>.

- [228] RODRIGUES I, 2020, *Gaussian blur — Noise reduction filter in image processing*, [Online], [Cited July 2022], Available from <https://towardsdatascience.com/crisp-dm-methodology-leader-in-data-mining-and-big-data-467efd3d3781>.
- [229] ROE E, 2021, *Aged document binarization using the U-Net architecture*, [Online], [Cited July 2022], Available from https://medium.com/@er_95882/aged-document-binarization-using-the-u-net-architecture-fa171cba6bd2.
- [230] ROSENBROCK A, 2021, *Improving OCR results with basic image processing*, [Online], [Cited June 2022], Available from <https://pyimagesearch.com/2021/11/22/improving-ocr-results-with-basic-image-processing/>.
- [231] RSIP VISION, 2021, *Image features for classification*, [Online], [Cited July 2022], Available from [https://www.rsipvision.com/image-features-for-classification/#:~:text=Ideally%2C%20features%20should%20be%20invariant,magnification\)%20or%20changing%20acquisition%20angle.](https://www.rsipvision.com/image-features-for-classification/#:~:text=Ideally%2C%20features%20should%20be%20invariant,magnification)%20or%20changing%20acquisition%20angle.)
- [232] RUDER S, 2016, *An overview of gradient descent optimization algorithms*, [Online], [Cited October 2022], Available from <https://runder.io/optimizing-gradient-descent/>.
- [233] RUMELHART DE, HINTON GE & WILLIAMS RJ, 1986, *Learning representations by back-propagating errors*, *Nature*, **323(6088)**, pp. 533–536.
- [234] RUSSO F, 2002, *An image enhancement technique combining sharpening and noise reduction*, *IEEE Transactions on Instrumentation and Measurement*, **51(4)**, pp. 824–828.
- [235] RYU S, KIM S, CHOI J, YU H & LEE GG, 2017, *Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems*, *Pattern Recognition Letters*, **88**, pp. 26–32.
- [236] SADRI J & CHERIET M, 2009, *A new approach for skew correction of documents based on particle swarm optimization*, *Proceedings of the 10th International Conference on Document Analysis and Recognition*, Barcelona, pp. 1066–1070.
- [237] SAGHEER A & KOTB M, 2019, *Time series forecasting of petroleum production using deep LSTM recurrent networks*, *Neurocomputing*, **323**, pp. 203–213.
- [238] SAHA S, 2018, *A comprehensive guide to convolutional neural networks — The ELI5 way*, [Online], [Cited April 2022], Available from <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [239] SAHOO PK, SOLTANI S & WONG AK, 1988, *A survey of thresholding techniques*, *Computer Vision, Graphics, and Image Processing*, **41(2)**, pp. 233–260.
- [240] SAHU M & DASH R, “A survey on deep learning: convolution neural network (CNN)”, in: *Intelligent and Cloud Computing*, Springer, 2021, pp. 317–325.
- [241] SAI Y, JINXIA R & ZHONGXIA L, 2009, *Learning of neural networks based on weighted mean squares error function*, *Proceedings of the International Symposium on Computational Intelligence and Design*, Changsha, pp. 241–244.
- [242] SAID N, AHMAD K, RIEGLER M, POGORELOV K, HASSAN L, AHMAD N & CONCI N, 2019, *Natural disasters detection in social media and satellite imagery: a survey*, *Multimedia Tools and Applications*, **78(22)**, pp. 31267–31302.
- [243] SAINI R, DOBSON D, MORREY J, LIWICKI M & LIWICKI FS, 2019, *ICDAR 2019 historical document reading challenge on large structured Chinese family records*, *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, pp. 1499–1504.

- [244] SANDLER M, HOWARD A, ZHU M, ZHMOGINOV A & CHEN L.-C, 2018, *Mobilenetv2: Inverted residuals and linear bottlenecks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City (UT), pp. 4510–4520.
- [245] SARFRAZ M & RASHEED Z, 2008, *Skew estimation and correction of text using bounding box*, Proceedings of the 5th International Conference on Computer Graphics, Imaging and Visualisation, Washington (DC), pp. 259–264.
- [246] SAUVOLA J & PIETIKÄINEN M, 2000, *Adaptive document image binarization*, Pattern Recognition, **33**(2), pp. 225–236.
- [247] SAYGIN AP, CICEKLI I & AKMAN V, 2000, *Turing test: 50 years later*, Minds and Machines, **10**(4), pp. 463–518.
- [248] SCHEIDL H, FIEL S & SABLATNIG R, 2018, *Word beam search: A connectionist temporal classification decoding algorithm*, Proceedings of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, pp. 253–258.
- [249] SCHUSTER M & PALIWAL KK, 1997, *Bidirectional recurrent neural networks*, IEEE Transactions on Signal Processing, **45**(11), pp. 2673–2681.
- [250] SENTHILKUMARAN N & VAITHEGI S, 2016, *Image segmentation by using thresholding techniques for medical images*, Computer Science & Engineering: An International Journal, **6**(1), pp. 1–13.
- [251] SHANE J, 2018, *Neural networks, explained*, [Online], [Cited February 2022], Available from <https://physicsworld.com/a/neural-networks-explained/#:~:text=What%20are%20neural%20networks%3F,programmed%20with%20rules%20to%20follow.>
- [252] SHANTHI T & SABEENIAN R, 2019, *Modified Alexnet architecture for classification of diabetic retinopathy images*, Computers & Electrical Engineering, **76**, pp. 56–64.
- [253] SHARMA S, 2017, *Epoch vs batch size vs iterations*, [Online], [Cited October 2022], Available from <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>.
- [254] SHEARER C, 2000, *The CRISP-DM model: The new blueprint for data mining*, Journal of Data Warehousing, **5**(4), pp. 13–22.
- [255] SHELKE MS, DESHMUKH PR & SHANDILYA VK, 2017, *A review on imbalanced data handling using undersampling and oversampling technique*, Internasional Journal on Recent Trends in Engineering Research, **3**(4), pp. 444–449.
- [256] SHI B, BAI X & YAO C, 2016, *An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **39**(11), pp. 2298–2304.
- [257] SHINDE AA & CHOUGULE D, 2012, *Text pre-processing and text segmentation for OCR*, International Journal of Computer Science Engineering and Technology, **2**(1), pp. 810–812.
- [258] SHOBHA RANI N & VASUDEV T, 2018, *An efficient technique for detection and removal of lines with text stroke crossings in document images*, Proceedings of the International Conference on Cognition and Recognition, Zhengzhou, pp. 83–97.
- [259] SIDDIQUI S, SALMAN A, MALIK I, SHAFAIT F, MIAN A, SHORTIS M & HARVEY E, 2017, *Automatic fish species classification in underwater videos: Exploiting pretrained deep neural network models to compensate for limited labelled data*, Journal of Marine Science, **75**, pp. 374–387.

- [260] SIDOROV G, VELASQUEZ F, STAMATATOS E, GELBUKH A & CHANONA-HERNÁNDEZ L, 2014, *Syntactic n-grams as machine learning features for natural language processing*, Expert Systems with Applications, **41(3)**, pp. 853–860.
- [261] SILVER D, SCHRITTWIESER J, SIMONYAN K, ANTONOGLU I, HUANG A, GUEZ A, HUBERT T, BAKER L, LAI M & BOLTON A, 2017, *Mastering the game of go without human knowledge*, Nature, **550(7676)**, pp. 354–359.
- [262] SIMONYAN K & ZISSERMAN A, 2015, *Very deep convolutional networks for large-scale image recognition*, Proceedings of the 3th, San Diego (CA), pp. 1–14.
- [263] SINGH A, PANG G, TOH M, HUANG J, GALUBA W & HASSNER T, 2021, *TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville (TN), pp. 8802–8812.
- [264] SINGH J & BHUSHAN B, 2019, *Real-time indian license plate detection using deep neural networks and Optical character recognition using LSTM Tesseract*, Proceedings of the International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Noida, pp. 347–352.
- [265] SIRFZ, 2021, *A Python wrapper for the tesseract-ocr API*, [Online], [Cited June 2022], Available from <https://github.com/sirfz/tesseractocr>.
- [266] SMARTENGINES, 2020, *Banking & insurance*, [Online], [Cited March 2021], Available from <https://smartengines.com/industries/banking-insurance/>.
- [267] SMITH R, 1995, *A simple and efficient skew detection algorithm via text row accumulation*, Proceedings of the International Conference on Document Analysis and Recognition, Montreal, pp. 1145–1148.
- [268] SMITH R, 2007, *An overview of the Tesseract OCR engine*, Proceedings of the 9th Conference on Document Analysis and Recognition (ICDAR), Parana, pp. 629–633.
- [269] SMITH RW, 1987, *The extraction and recognition of text from multimedia document images*, PhD thesis, University of Bristol.
- [270] SPITZ AL, 1997, *Determination of the script and language content of document images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **19(3)**, pp. 235–245.
- [271] SRIHARI SN & GOVINDARAJU V, 1989, *Analysis of textual images using the Hough transform*, Machine Vision and Applications, **2(3)**, pp. 141–153.
- [272] SRIHARI SN, SHEKHAWAT A & LAM SW, “Optical character recognition (OCR)”, in: *Encyclopedia of Computer Science*, 2003, pp. 1326–1333.
- [273] SU B, LU S & LIM TC, 2012, *Restoration of motion blurred document images*, Proceedings of the 27th Annual ACM Symposium on Applied Computing, Trento, pp. 767–770.
- [274] SUN Y, NI Z, CHNG C.-K, LIU Y, LUO C, NG CC, HAN J, DING E, LIU J & KARATZAS D, 2019, *ICDAR 2019 competition on large-scale street view text with partial labeling-RRC-LSVT*, Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Sydney, pp. 1557–1562.
- [275] SURYANARAYANA S, DEEKSHATULU B, KISHORE KL & KUMAR YR, 2012, *Estimation and removal of Gaussian noise in digital images*, International Journal of Electronics and Communication Engineering, **5(1)**, pp. 23–33.
- [276] TAKAYAMA J, NOMOTO E & ARASE Y, 2019, *Dialogue breakdown detection robust to variations in annotators and dialogue systems*, Computer Speech & Language, **54**, pp. 31–43.

- [277] TAKYAR A, 2018, *AI applications across major industries*, [Online], [Cited March 2021], Available from <https://www.leewayhertz.com/ai-applications-across-major-industries/>.
- [278] TAN M & LE Q, 2019, *Efficientnet: Rethinking model scaling for convolutional neural networks*, Proceedings of the International Conference on Machine Learning, Long Beach (CA), pp. 6105–6114.
- [279] TANG Y, 2013, *Deep learning using support vector machines*, Clinical Orthopaedics and Related Research, **2**, pp. 1–2.
- [280] TANNER S, MUÑOZ T & ROS PH, 2009, *Measuring mass text digitization quality and usefulness*, D-lib Magazine, **15(7/8)**, pp. 1082–9873.
- [281] TENSMEYER C & MARTINEZ T, 2020, *Historical document image binarization: A review*, SN Computer Science, **1(3)**, pp. 1–26.
- [282] THANGA S, 2021, *Tesseract vs Keras-OCR vs EasyOCR*, [Online], [Cited June 2022], Available from <https://medium.com/mlearning-ai/tesseract-vs-keras-ocr-vs-easyocr-ec8500b9455b>.
- [283] THE UNIVERSITY OF AUCKLAND, 2022, *Image filtering*, [Online], [Cited July 2022], Available from <https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering.pdf>.
- [284] THEVENOT J, LÓPEZ MB & HADID A, 2017, *A survey on computer vision for assistive medical diagnosis from faces*, IEEE Journal of Biomedical and Health Informatics, **22(5)**, pp. 1497–1511.
- [285] TOMASI C & MANDUCHI R, 1998, *Bilateral filtering for gray and color images*, Proceedings of the 6th International Conference on Computer Vision, Bombay, pp. 839–846.
- [286] TURING AM, “Computing machinery and intelligence”, in: *Parsing the Turing Test*, Springer, 2009, pp. 23–65.
- [287] UKKONEN E, 1995, *On-line construction of suffix trees*, Algorithmica, **14(3)**, pp. 249–260.
- [288] UNTITLED PATTERN, 2022, *Digitize your receipts using computer vision*, [Online], [Cited October 2022], Available from <https://regexr.com/>.
- [289] VAMVAKAS G, GATOS B, PRATIKAKIS I, STAMATOPOULOS N, RONIOTIS A & PERANTONIS SJ, 2007, *Hybrid off-line OCR for isolated handwritten Greek characters*, Proceedings of the 4th International Conference on Signal Processing, Pattern Recognition, and Applications, Innsbruck, pp. 197–202.
- [290] VAN DEN BOOMGAARD R & VAN BALEN R, 1992, *Methods for fast morphological image transforms using bitmapped binary images*, Graphical Models and Image Processing, **54(3)**, pp. 252–258.
- [291] VAN DER AALST WM, BICHLER M & HEINZL A, 2018, *Robotic process automation*, Business & Information Systems Engineering, **60(4)**, pp. 269–272.
- [292] VAN ENGELEN JE & HOOS HH, 2020, *A survey on semi-supervised learning*, Machine Learning, **109(2)**, pp. 373–440.
- [293] VAN HOUDT G, MOSQUERA C & NÁPOLES G, 2020, *A review on the long short-term memory model*, Artificial Intelligence Review, **53(8)**, pp. 5929–5955.
- [294] VIJAYARANI S & SAKILA A, 2015, *Performance comparison of OCR tools*, International Journal of UbiComp (IJU), **6(3)**, pp. 19–30.

- [295] VORSTER C, 2021, Data Scientist at Capitec Bank, [Personal Communication], Contactable at chrisvorster@capitecbank.co.za.
- [296] VOULODIMOS A, DOULAMIS N, DOULAMIS A & PROTOPAPADAKIS E, 2018, *Deep learning for computer vision: A brief review*, Computational Intelligence and Neuroscience, **2018**, pp. 7068349–7068349.
- [297] WALCZAK S, “Artificial neural networks”, in: *Encyclopedia of Information Science and Technology, Fourth Edition*, IGI Global, 2018, pp. 120–131.
- [298] WANG H, MA C & ZHOU L, 2009, *A brief review of machine learning and its application*, Proceedings of the International Conference on Information Engineering and Computer Science, Wuhan, pp. 1–4.
- [299] WANG J, LEUNG MK & HUI SC, 1997, *Cursive word reference line detection*, Pattern Recognition, **30(3)**, pp. 503–511.
- [300] WEISS K, KHOSHGOFTAAR TM & WANG D, 2016, *A survey of transfer learning*, Journal of Big Data, **3(1)**, pp. 1–40.
- [301] WEIZENBAUM J, 1966, *ELIZA — A computer program for the study of natural language communication between man and machine*, Communications of the ACM, **9(1)**, pp. 36–45.
- [302] WERBOS PJ, 1990, *Backpropagation through time: What it does and how to do it*, Proceedings of the IEEE, **78(10)**, pp. 1550–1560.
- [303] WHAT-WHEN-HOW, 2020, *Geometric transformations (introduction to video and image processing) part 1*, [Online], [Cited June 2022], Available from <http://what-when-how.com/introduction-to-video-and-image-processing/geometric-transformations-introduction-to-video-and-image-processing-part-1/>.
- [304] WILLIAMS RJ & ZIPSER D, 1989, *A learning algorithm for continually running fully recurrent neural networks*, Neural Computation, **1(2)**, pp. 270–280.
- [305] WU H & GU X, 2015, *Towards dropout training for convolutional neural networks*, Neural Networks, **71**, pp. 1–10.
- [306] WU J, 2017, *Introduction to convolutional neural networks*, National Key Lab for Novel Software Technology, **5(23)**, pp. 495.
- [307] XU B, WANG N, CHEN T & LI M, 2015, *Empirical evaluation of rectified activations in convolutional network*, arXiv preprint arXiv:1505.00853.
- [308] YAMASHITA R, NISHIO M, DO RKG & TOGASHI K, 2018, *Convolutional neural networks: An overview and application in radiology*, Insights Into Imaging, **9(4)**, pp. 611–629.
- [309] YANHUI C, 2018, *A battle against amnesia: A brief history and introduction of recurrent neural networks*, [Online], [Cited March 2022], Available from <https://towardsdatascience.com/a-battle-against-amnesia-a-brief-history-and-introduction-of-recurrent-neural-networks-50496aae6740#:~:text=1.1%20The%20History,architecture%20was%20invented%20in%201997.>
- [310] YAP BW, RANI KA, RAHMAN HAA, FONG S, KHAIRUDIN Z & ABDULLAH NN, 2014, *An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets*, Proceedings of the International Conference on Advanced Data and Information Engineering, Kuala Lumpur, pp. 13–22.
- [311] YE Q & DOERMANN D, 2014, *Text detection and recognition in imagery: A survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **37(7)**, pp. 1480–1500.

- [312] YEE OS, SAGADEVAN S & MALIM NHAH, 2018, *Credit card fraud detection using machine learning as data mining technique*, Journal of Telecommunication, Electronic and Computer Engineering (JTEC), **10(1-4)**, pp. 23–27.
- [313] YOSINSKI J, CLUNE J, BENGIO Y & LIPSON H, 2014, *How transferable are features in deep neural networks?*, Advances in Neural Information Processing Systems, **27**, pp. 3320–3328.
- [314] YU W, YANG K, BAI Y, XIAO T, YAO H & RUI Y, 2016, *Visualizing and comparing AlexNet and VGG using deconvolutional layers*, Proceedings of the 33th International Conference on Machine Learning, New York City (NY).
- [315] ZHONG D, YANG Y & DU X, 2018, *Palmprint recognition using siamese network*, Proceedings of the Chinese Conference on Biometric Recognition, Beijing, pp. 48–55.
- [316] ZOU J, HAN Y & SO S.-S, 2008, *Overview of artificial neural networks*, Artificial Neural Networks, pp. 14–22.
- [317] ZOU Z, SHI Z, GUO Y & YE J, 2019, *Object detection in 20 years: A survey*, arXiv preprint arXiv:1905.05055.
- [318] ZYON, 2016, *Remove background noise from image to make text more clear for OCR*, [Online], [Cited June 2022], Available from <https://stackoverflow.com/questions/33881175/remove-background-noise-from-image-to-make-text-more-clear-for-ocr>.