Froth Texture Extraction with Deep Learning

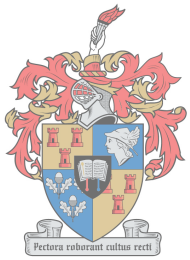
by

Zander Christo Horn

Thesis presented in partial fulfilment
of the requirements for the Degree

of

MASTER OF ENGINEERING
(EXTRACTIVE METALLURGICAL ENGINEERING)



in the Faculty of Engineering
at Stellenbosch University

Supervisor

Dr Lidia Auret

Co-Supervisors

Prof Chris Aldrich

Prof Ben Herbst

March 2018

Declaration of Originality

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 2018

Abstract

Soft-sensors are of interest in mineral processing and can replace slower or more expensive sensors by using existing process sensors. Sensing process information from images has been demonstrated successfully, but performance is dependent on feature extractors used. Textural features which utilise spatial relationships within images are preferred due to greater resilience to changing imaging and process conditions.

Traditional texture feature extractors require iterative design and are sensitive to changes in imaging conditions. They may have many hyperparameters, leading to slow optimisation. Robust and accurate sensing is a key requirement for mineral processing, making current methods of limited potential under realistic industrial conditions.

A platinum froth flotation case study was used to compare traditional texture feature extractors with a proposed deep learning feature extractor: convolutional neural networks (CNNs). Deep learning applies artificial neural networks with many hidden layers and specialised architectures for powerful correlative performance through automated training. All information of the input data structure is determined inherently in training with only a limited number of hyperparameters. However, deep learning methods risk overfitting with small datasets, which must be mitigated.

A CNN classifier and a framework for unbiased comparison between feature extractors were developed for predicting high to low grade classes of platinum in flotation froth images. CNNs can perform all the functions of a soft-sensor, but this may bias performance comparison. Instead, features were extracted from hidden layers in CNNs and fed into a traditional soft-sensor. This ensured performance measurements were unbiased across all feature extractors. With a full factorial experiment, the following CNN hyperparameters were evaluated: batch size, number of convolutional filters, and convolutional filter size.

Accuracy of grade classification was used to score feature extractors. These reference texture feature extractors were compared to CNNs: Local Binary Patterns, Grey-Level Co-occurrence Matrices, and Wavelets. The impact of spectral features (bulk image features such as average colour) was also evaluated, as CNNs can also use spectral image properties to create features, unlike traditional texture extractors. Extractors were tested with input resolutions from 16x16 to 128x128 with two soft-sensor models: Linear Discriminant Analysis, and k-Nearest Neighbour classifiers. Optimal grade classification accuracies were: CNN – 96.5%, LBP – 100%, GLCM – 73.7%, Wavelets – 98.3%, and Spectral – 98.4%

Training CNNs to extract features was successful with robust results regardless of hyperparameters selected. The only statistically significant differences obtained during training were that smaller batch size and smaller input resolution gave superior training performance. Results were found to be reproducible for all models.

Analysing learned CNN features indicated both textural and spectral features were utilised. Overall results showed spectral features gave good classification performance, potentially adding to CNN performance. CNNs showed comparable performance to other texture feature extractors at all resolutions.

This proof of concept implementation shows promise for deep learning methods in mineral processing applications. The resilience of CNNs to changes in imaging and process conditions could not be evaluated due to limited data in the case study. Future work with deep learning methods, while promising, will require larger datasets which are more representative of a variety of process conditions.

Abstrak

Inferensiële waarneming is van belang in die mineraalverwerkingsveld. Dit kan stadiger of duurder aanlynsensors vervang met bestaande prosesveranderlike sensors. Waarneming van proses inligting uit beelde is suksesvol gedemonstreer, maar werkverrigting is afhanklik van die kenmerk-ekstraksie metode. Teksturele eienskappe wat ruimtelike verhoudinge binne beelde gebruik geniet voorkeur as gevolg van hul veerkragtigheid teenoor veranderings in beeldopname en prosesomstandighede.

Tradisionele tekstuurkenmerk-ekstraksiemetodes benodig iteratiewe ontwerp en is sensitief vir veranderinge in beeldopnameomstandighede. Tekstuurkenmerk-ekstraksiemetodes kan baie hiperparameters hê wat stadige optimalisering veroorsaak. Robuuste, akkurate waarneming is 'n belangrike vereiste vir minerale verwerking. Huidige tegnieke van tekstuur-ekstraksie het beperkte potensiaal onder realistiese industriële toestande.

’n Platinumflottasieskuim gevallestudie is gebruik om tradisionele tekstuurkenmerk-ekstraksie te vergelyk met ’n voorgestelde diep-leer kenmerk-ekstraksiemetode: konvolusionele neurale netwerke (KNNs). Diep-leer pas kunsmatige neurale netwerke, met versteekte lae en gespesialiseerde argitektuur, toe. Dit maak voorsiening vir ‘n kragtige korrelatiewe prestasie en geoutomatiseerde opleiding. Alle inligting rakende die insetdata struktuur word inherent bepaal tydens opleiding met slegs ’n beperkte hoeveelheid hiperparameters. Klein datastelle kan egter lei na oormatige passing in diep-leermetodes. Dié risiko moet versag word.

’n KNN-klassifiseerder en ‘n raamwerk vir onbevooroordeelde vergelyking tussen kenmerk-ekstraksiemetodes is hier ontwikkel om lae tot hoë platinumgraadklasse van flotasieskuimbeelde te voorspel. KNNs kan dieselfde funksies as inferensiële sensors verrig, maar dit kan sydig wees in prestasiemetings. Om dit te voorkom is kenmerke van versteekte lae in die KNNs onttrek en as insette in die tradisionele inferensiële sensor gebruik. Dit het verseker dat kenmerk-ekstraksie nie sydig was as gevolg van die korrelerende vermoëns van KNNs nie. Die volgende KNN-hiperparameters is evalueer deur ’n vol-faktor eksperiment: bondelgrootte, toenemende of konstante aantal konvolusiefilters, en konvolusiefiltergrootte.

Tekstuurkenmerk-ekstraksiemetodes is gegradeer volgens die klassifikasie-akkuraatheid van platinumgraad. Hierdie tradisionele tekstuurkenmerk-ekstraksiemetodes is vergelyk met KNNs: Lokale Binêre Patrone (LBP), Grysskaalmede-aanwesigheidsmatrikse (GSMMs), en Golfie-transformasies. Die impak van spektrale eienskappe (byvoorbeeld massa eienskappe soos gemiddelde kleur) is geëvalueer, aangesien KNNs spektrale beeldkenmerke ook kan toepas om eienskappe te skep, anders as tradisionele tekstuur-ekstraksiemetodes. Die kenmerk-ekstraksiemetodes is getoets in 'n reeks insetresolusies van 16x16 tot 128x128 met een van twee sagte-sensor modelle: Lineêre Diskriminant Analise (LDA) en k-Naaste

Buurman (k-NB) klassifiseerders. Die beste klassifikasie-akkuraatheid vir elke metode was as volg: KNN – 96.5%, LBP – 100%, GSMM – 73.7%, Golfie-transformasies – 98.3%, en Spektraal– 98.4%

Die opleiding van KNNs vir kenmerk-ekstraksie was suksesvol, met robuuste resultate ongeag die gekose hiperparameters. Die enigste statisties beduidende resultate wat behaal is, is dat kleiner bondelgrootte en kleiner insetresolusie 'n beter opleidingsprestasie het. Herhalingstoetsing het bevind dat opleidingsresultate reproduseerbaar was.

Ontleding van geleerde KNN-kenmerke het aangedui dat beide teksturele en spektrale kenmerke gebruik is. Oor die algemeen het inferensiële toetse getoon dat spektrale kenmerke tot uitstekende klassifikasie-prestasie gelei het, wat moontlik die KNN se prestasie verbeter. KNNs het vergelykbare prestasie getoon met ander tekstuurfunksie-ekstrakte by alle beeldresolusies, en het klassifiekasie-uitslae geproduseer wat geskik is vir beheer- en moniteringdoeleindes.

Met hierdie bewys-van-konsep implementering, toon diep-leermetodes belofte vir gebruik in minerale verwerkingsprobleme. Die veerkragtigheid van KNNs teen verandering in beeld- en prosesomstandighede kon egter nie geëvalueer word nie, as gevolg van beperkte data in die gevallestudie. Verdere werk met diep-leermetodes, terwyl belowend, sal groter datastelle benodig wat meer verteenwoordigend is van ‘n verskeidenheid prosesomstandighede.

Acknowledgements

First and foremost, I would like to thank Lidia for providing me the opportunity to pursue further studies and produce this work. Her support and patience were invaluable and helped me through some tough times. I apologise to subjecting her to my sometimes-vexing understanding of deadlines.

To my wife, Judy-Ann, I thank her endless support during this work and for being my proof-reader extraordinaire. Without her my semantically reckless use of commas and semicolons would still be terrorising the readers.

To my family, for tolerating my desire to continue studying and not enter the job market. Their support and love always encourages me.

Lastly, I would like to thank all my friends and colleagues for listening to me ramble about neural networks and flotation froths and for keeping me sane with dinner, drinks, and board games.

Table of Contents

Nomenclature

ANN	Artificial Neural Networks
ANOVA	Analysis of variance
b_0	Bias term of a neuron
β_1	1st moment estimate exponential decay rate
β_2	2nd moment estimate exponential decay rate
$C_{i,j,c}$	Convolutional filter value applicable to $I_{i,j,c}$
CNN	Convolutional Neural Networks
δ	Gradient stabilisation term
∂_{ij}	Kronecker delta of predicted class j and actual class i
DNN	Deep Neural Networks
F_{cfmax}	Maximum convolutional filter size
F_p	Max-pooling size
$f(x)$	Some activation function for neuron
GLCM	Grey-level Co-occurrence Matrices
GPGPUE	General Purpose Graphics Processing Unit
GPU	Graphical Processing Unit
H	Activation outputs of a layer of neurons
H'	Normalised activation outputs of a layer of neurons
H_i	i-th neuron in hidden layer
H_0	Null-hypothesis
H_1	Alternative hypothesis
HSL	Hue, Saturation, Lightness
HSV	Hue, Saturation, Value
I_i	i-th neuron in input layer
I_i	i-th input to neuron
$I_{i,j,c}$	Pixel at location i,j and channel c of image window
k-NN	k-Nearest Neighbours
L	Classification loss
LAB	Luminosity, colour channels A and B
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
m	Total number of activations in a layer
MIA	Multivariate Image Analysis
μ	Mean

N	Number of images in a batch
n_{conv}	Number of convolutional layers
O_0	Output of a neuron
O_i	i-th neuron in output layer
OSA	On-stream Analyser
PCA	Principal Component Analysis
PGM	Platinum Group Metal
PLS	Partial Least Squares
RBM	Restricted Boltzmann Machines
ReLU	Regularised Linear Units
RGB	Red, Green, Blue
r_{input}	Input image resolution
SGD	Stochastic Gradient Descent
σ^2	Variance
SVM	Support Vector Machines
UG2	Upper Group 2
W_i	Weight for i-th input to a neuron
w_i	Weighting term for classification loss calculation
y	Actual class label
$\hat{y}$	Predicted class probability
x	Summed weighted input of neuron
x_i	Summed weighted input of i-th neuron in layer

Glossary

Architecture (Neural Networks)	The specific arrangement of neurons in and between layers
Assay	Determination of fraction of particular mineral or metal in ore
Backpropagation	Updating of neural network parameters through error derivative of parameters in neural network
Classification	Categorisation of data points into specific discrete representative classes
Classifier	Mathematical model which places data within discrete categories
Comminution	Grinding of mineral ore into smaller particle sizes
Convolution (Neural Networks)	Neural architecture where the input is divided into smaller overlapping input areas which all share the same parameters
Convolve	The process of convolution, a piecewise matrix multiplication with convolutional parameters, centred around each input element
Deep Learning	The application of deep or many layered neural networks
Dimensionality Reduction	Decreasing the number of values in a descriptive dataset while ideally maintaining descriptive representation
Feature	Any variable or variables which describe or represent data being analysed
Feature Extraction	The process of obtaining features from larger datasets through manipulation, selection, or other transforms
Forward Propagation	Feeding data into a neural network and producing a final output. Used in conjunction with backpropagation to calculate training error.
Hyperparameter	User settable parameters which affect or determine a large number of hidden parameters
Inferential-Sensor	Algorithm which estimates system properties while not directly measuring those properties
Labelled Data	Data which is associated with a known classification or regression output which can be trained against
Layer	A collection of neurons performing the same action at the same depth in the neural network. Usually no intra-layer connections between neurons
Loss Function	Mathematical model which describes predictive error for a model across the entire dataset being tested
Machine Learning	The automated parameter search for a model as controlled by a suitable loss function and related objective i.e. minimise loss

Machine Vision	Application of computerised imaging for data extraction purposes
Overfitting	The process of encoding a limited training dataset into model parameters and failing to discover underlying true correlations
Parameter	Any property within a model whose value is determined through machine learning and not manual human setting
Pooling	Reduction of dimension in images through grouping and merging of pixels in a certain area, and repeating across the entire image
Regularisation	Any method which introduces additional information to prevent overfitting
Slurry	A liquid like suspension of small particles in (most commonly) water to facilitate processing and transportation of solids
Soft-Sensors	A detection device which uses computer processing to correlate input information to a desired output
Sparge	To homogeneously distribute air through a liquid
Spectral	Information related to the relative intensity of different frequencies in data such as colour in images
Stride	The distance by which an operation on sub-data shifts within a dataset before operating on the next set of sub-data
Supervised Learning	Updating the parameters of a model through a targeted goal as determined by information correlated to the input data
Testing (Data)	Previously unseen inputs for evaluation of a trained machine learning model
Texture	Spatial relationships and patterns in image data
Training (Data)	Inputs used during updating of model parameters, not suitable for evaluating final performance due to risk of being encoded
Unsupervised Learning	Updating the parameters of a model through reconstruction of input data or other objective function not related to correlated information of the input
Validation (Data)	A subset of training data used during parameter updating to provide a crude indication of training performing while training
Variant	A set of data generated from original data with some transform applied to improve training effectiveness by increased representation

Chapter 1: Introduction

Process monitoring forms a fundamental basis of modern process plant operation and optimisation in almost all industries. Measuring process variables and determining key performance indicators is an integral part of this. Image-based soft-sensing is playing a growing part in process monitoring as computational power and imaging algorithms advance.

Soft-sensing through image analysis forms a core component of this study in which the application and performance of deep learning as a method of gathering inferential process data will be analysed. The intent is to extract useful textural feature information from images via deep learning and compare how effectively these may be coupled to process variables. The deep learning method selected will be benchmarked against traditional feature extraction methods in a soft-sensor framework such as that in Figure 1-1.

Of interest is implementing image-based soft-sensing for froth flotation systems, specifically in the platinum industry. Froth flotation is a complex process and many process variables influence froth appearance. In brief, froth flotation exploits hydrophobic and hydrophilic properties of minerals within an ore to induce separation. Within a flotation cell, air flows through a slurry, forming a froth at the surface to which (in most cases) the desired mineral has preferentially attached to. By examining flotation froths with suitable feature extraction methods, one or more process variables of interest may be correlated with the extracted features; this would allow for more effective process control and decision making at this step of platinum concentration. Key performance indicators for flotation such as grade and recovery are typically expensive or slow to measure with direct sensing methods.

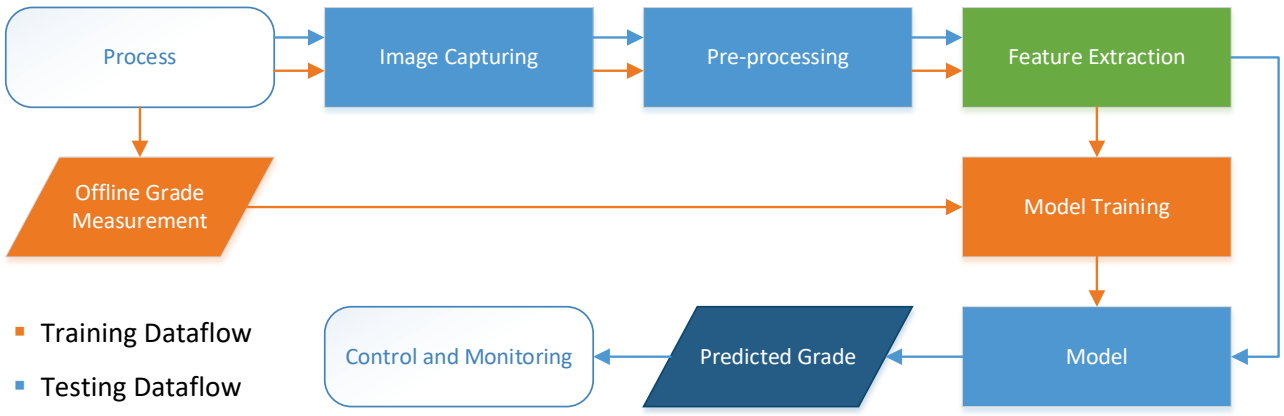


Figure 1-1: Overview of a Soft-Sensor Training and Usage Framework

Traditional methods of extracting data from images have several drawbacks hindering widespread implementation. Parametric texture-based methods make specific assumptions of features present and have several hyperparameters (parameters which are fixed prior to training and affect hidden parameters) to tune. Spectral methods (simplistic bulk measurements of image properties, i.e. the average red, green, and blue value of the entire image) have been examined as well but are sensitive to changes in image capture conditions. These methods need iterative training and require specific prior knowledge of image texture

types to determine which method and hyperparameter combination is suitable for the type of process images being analysed.

Deep learning in comparison has properties which make it an attractive alternative to current multivariate image analysis methods implemented by industry. Deep learning is an evolution of shallow neural networks by applying many neurons in many layers. The large number of layers in deep learning networks allows specific layer architectures to be implemented, mitigating many shortcomings of shallow neural networks. It does not require prior knowledge to extract structured information from images due to its training process. Deep learning methods inherently discover any hidden features in images without making any assumptions of feature types and are less sensitive to hyperparameter selection.

Deep learning methods can form soft-sensors completely within themselves, both extracting features and producing predictions. However, to provide a baseline for comparison between methods, deep learning methods will only be used as feature extractors in this study. Feature sets from each feature extraction method will be compared in a shared soft-sensor architecture with common classifiers (mathematical models which predict class) to correlate features to process data.

1.1 Motivation

Except for specific applications and sensor systems, accurate and robust image-based soft-sensors remain uncommon in the process industry. Current image sensors are dependent on feature extractors which require slow iterative design or prior knowledge during selection to achieve good performance as shown by the variety and complexity of methods available (Tian, 2013). These systems do not generalise well to large changes in process conditions and are susceptible to imaging variances. If image-based sensors are to be more prevalent in process industries, alternative methods of feature extraction that are both more robust and generalizable must be examined.

Deep learning is a promising candidate for machine vision systems in process industries. In other image classification tasks, it outperforms most other methods previously considered (Xie, 2008), and is more robust and accurate under varying imaging conditions and use scenarios.

A search for literature with the following keywords was performed: “deep learning process engineering”, “deep learning chemical engineering”, “deep learning soft-sensing”, and “convolutional neural networks chemical engineering”. It was discovered that recent research is limited, with most results from the early 90s. Some of the later research by (Bharati & MacGregor, 1998) still faced problems with data and computational limitations. The focus on newer research has remained in shallow neural networks as classification algorithms. Despite the proliferation of high performance Graphical Processing Units (GPUs) for machine learning and new process data sets, deep learning methods have not been further evaluated in recent work for process engineering application (Duchesne, et al., 2012).

Widespread application of froth flotation in industry makes it an attractive candidate for developing and testing a deep learning based soft-sensor (Fuerstenau, et al., 2007). Case studies are possible in a variety of different conditions and mineralogical systems; in addition, suitable data capturing systems are already in place at many froth flotation separators at South African mining operations. These systems measure parameters such as froth height and bubble velocity, but often discard the original images due to storage limitations. Additional sensors such as on-stream analysers (OSA) may also be available to draw information from.

If a soft-sensor is successfully developed for froth flotation, it may easily be adapted to similar image analysis and sensing tasks due to the flexible nature of deep neural networks. Deep networks can transfer what they have learned in their training to new systems (Oquab, et al., 2014), or can be further trained with new data to adapt to other processes.

1.2 Challenges

Sufficient data for training a deep learning soft-sensor is a primary concern in this work. Deep learning systems require large amounts of training data to enable accurate prediction under actual operation. Insufficient data coupled with naïve training will result in such a system not recognising new information accurately (Srivastava, et al., 2014). This phenomenon is known as overfitting and deep learning systems are especially susceptible to it, resulting in a model of little value for sensing applications.

Gathering sufficient data itself is problematic; deep learning methods require significantly more labelled data than required to train soft-sensors with traditional methods of feature extraction. While image data from froth flotation cells may be captured easily, it is more difficult to obtain process data which can be correlated with those images. It may be possible to use standard offline analyses (with appropriate assumptions and estimates) performed at the plant in question; alternatively, a sampling campaign must be performed in concert with imaging for later assay. Some compromise must be made between new samples, existing data, and training optimisation as the number of samples required for naïve deep learning would be financially prohibitive to analyse.

Deep learning itself also provides some challenges. Aside from overfitting mentioned above, deep learning also requires significant computing power to perform training, especially in the case of high dimensional applications such as image analysis. The problem may be mitigated somewhat with proper selection of deep neural network architecture and using efficient deep learning software libraries. However, this may limit the types of neural nets and training methods which can be utilised (Krizhevsky, et al., 2012).

Lastly, to be considered a viable online sensor, a deep learning soft-sensor must be both fast and accurate. This project will investigate the requirements of a soft-sensor in monitoring froth flotation and determine

which key performance indicators must be considered to evaluate performance of deep learning in contrast to other methods.

1.3 Objectives

Three primary objectives have been developed from the challenges facing this project:

1. Develop and evaluate a deep learning system for froth flotation sensing.
2. Compare deep learning feature extractors with traditional ones and evaluate the performance requirements for a soft-sensor in froth flotation
3. Contextualise results of deep learning system development with industry requirements and challenges for vision-based soft-sensing systems

1.4 Scope

This is a preliminary research project into the application of deep learning for soft-sensor systems. It would be premature and unrealistic to produce a complete and final product for industrial application. This study will focus on creating a proof of concept for soft-sensing in froth flotation. Any additional work beyond the initial scope of the project will only be considered once the primary objectives have been realised.

Chapter 2: Froth Flotation

Froth flotation is an effective and well-established process used for concentration in the mining industry, with applications to processes ranging from copper to platinum concentration. Its ability to rapidly separate complex or low-quality ores with small particle sizes makes it an attractive process compared to gravity or density separation methods. While sensors exist to measure parameters such as air flow and froth height in a flotation cell, systems which measure process conditions within flotation cells such as platinum grade or fault conditions are rare (Liu & MacGregor, 2008; Liu, et al., 2005). To meet the performance requirements of industrial applications, many parallel and series flotation cells circuits are used in a flotation plant with recycle streams to achieve separation and throughput targets.

As with all mineral concentrators, the operation of a froth flotation cell is characterised by a grade/recovery curve. Grade (the fraction of the output consisting of desired minerals) and recovery (the fraction of total input desired minerals in the output) are both important performance indicators of the process unit which would provide control benefits if measured online. Currently on-stream analysers (OSAs) are used to measure these parameters. However, they are expensive to purchase and require continuous calibration and maintenance. Additionally, certain OSAs utilise X-ray or other ionising radiation sources for sensing, presenting a risk to workers (Owen, 1988). One effect of this is that OSAs are generally purchased to measure the output of banks of flotation cells, sensing from each cell would be prohibitively expensive due to the large number present in each bank (Holtham & Nguyen, 2002). This increases dead-time significantly and eliminates the possibility of detecting changes and problems in individual flotations cells.

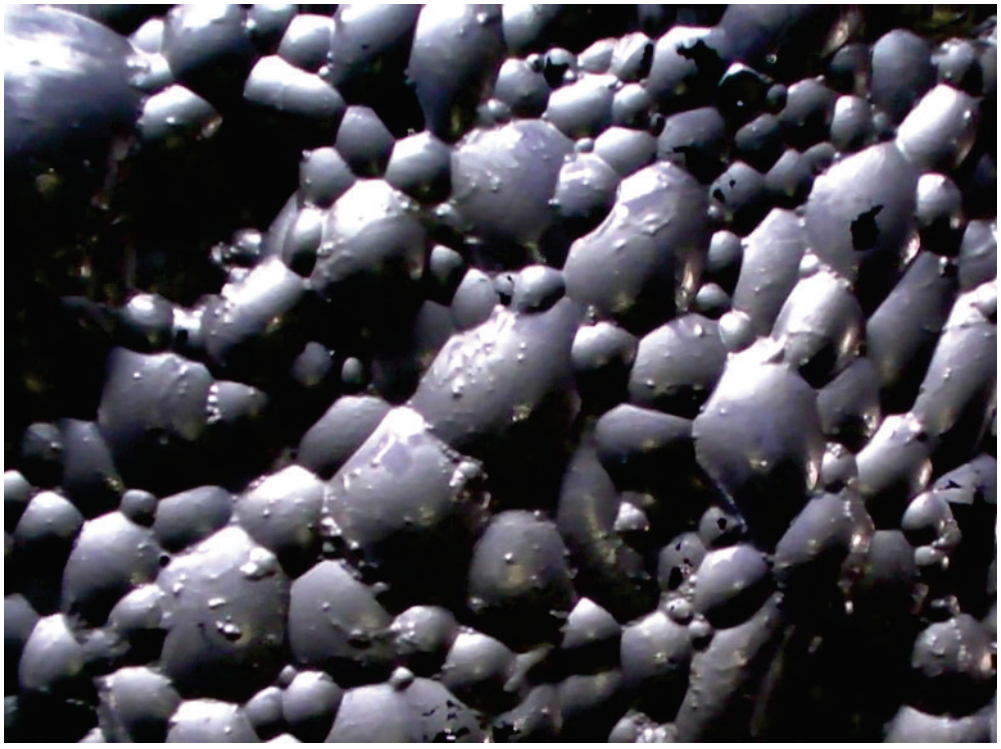


Figure 2-1: Image of Platinum Flotation Froth

In the platinum mining industry, from which the case studies for this work are taken, froth flotation is the primary concentration method for ores. An example image of the flotation froth can be seen in Figure 2-1. The case study will be limited to grade prediction due to limitations in the available process data.

2.1 Flotation Operation Theory

Flotation cells operate on mineral slurries resulting from prior stages of comminution (particle size reduction). As summarised in Figure 2-2, mineral ore is first crushed into progressively finer particles through primary, secondary, and tertiary comminution. The resultant slurry contains water with fine mineral particles suspended throughout it.

Flotation chemicals, described in further detail later, are added to the slurry to enhance properties beneficial for the flotation process. In the cell, the mixture is agitated, and air is sparged (uniformly pumped) through the cell to form a froth. As the bubbles rise to the surface of the cell, two distinct phases are formed: the liquid phase, known as the pulp, and the froth phase.

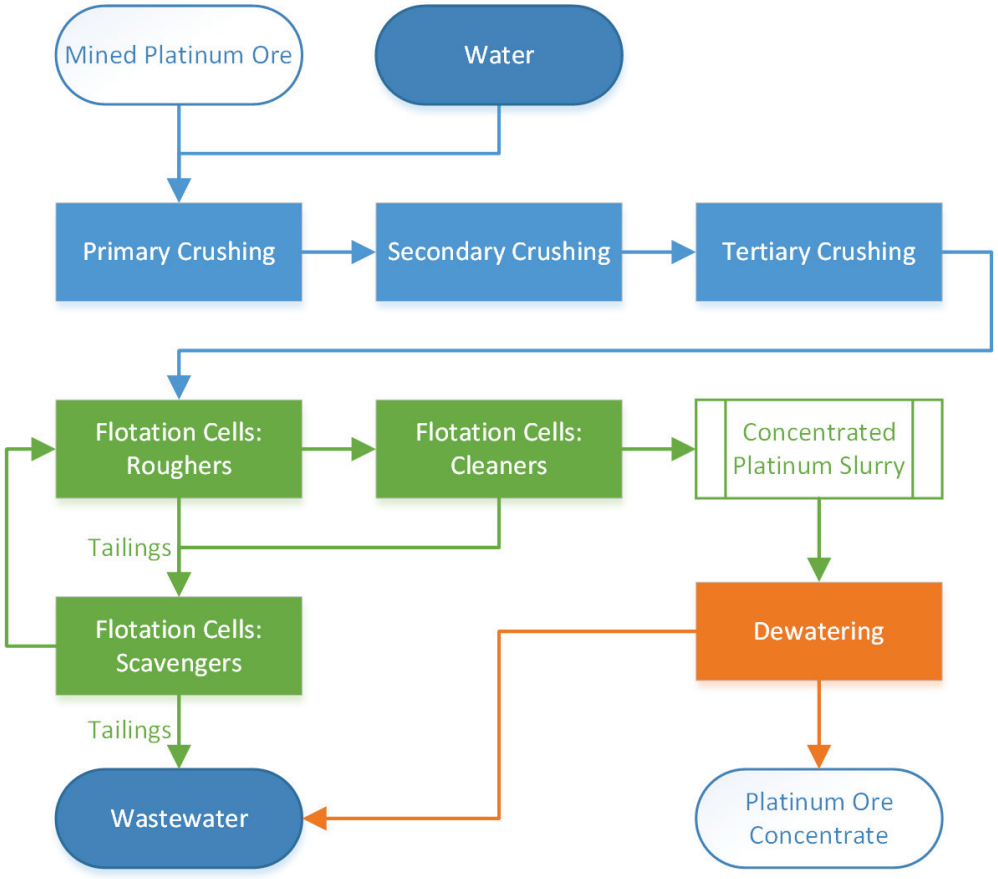


Figure 2-2: Simplified Overview of Froth Flotation Process for Platinum Beneficiation

Desired minerals, through action of the flotation reagents, are hydrophobically attracted to the bubble surfaces as they rise, while waste minerals remain within the pulp phase. This results in separation and concentration of desired minerals into the froth phase (Leja, 1982).

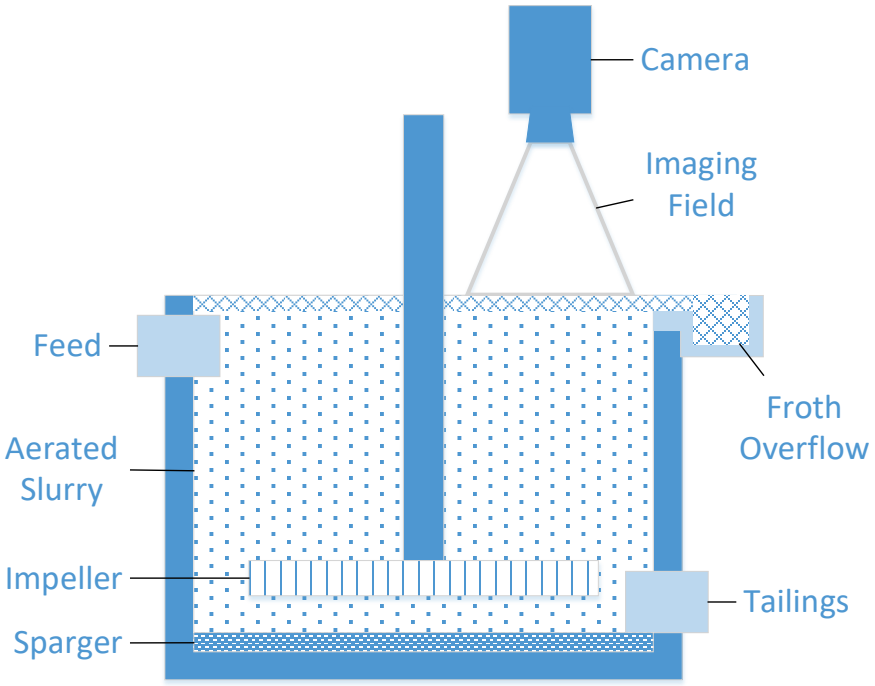


Figure 2-3: Flotation Cell Cross-Section Showing Relevant Components and Setup

Eventually the froth phase rises, up and over the weir level of the flotation cell as shown in Figure 2-3, where it is either sent to the next flotation cell for further concentration, or to final processing. A good overview of flotation systems may be found in (Gupta & Yan, 2006) and will be briefly summarised with supporting works as needed in the following sections.

2.1.1 Froth System Inputs

Process conditions, ore properties, and reagent addition all have a significant impact on both pulp and froth phases within a flotation cell. The impact of these on the flotation process will be evaluated and discussed.

2.1.1.1 Ore Type

A platinum process case study is used for this work and focus will be placed on the properties of platinum bearing ores. The effect of ores originating from specific reefs on outputs from the flotation process are likely to be significantly different from other platinum ores or processes not beneficiating platinum group metals (PGMs).

2.1.1.2 Collector Reagent

Certain minerals present in ores are generally hydrophilic and must be bound to a chemical agent to form the necessary hydrophobic properties for flotation. Collectors selectively adsorb onto minerals as a monomolecular layer through electrochemical processes, giving hydrophobic properties.

Flotation then occurs with hydrophobic particles clinging to bubble surfaces as they travel through the pulp phase, inducing separation. Excessive collector dosage may decrease hydrophobicity and adversely affect

separation efficiency. In the case of platinum processing, Xanthan is generally used as a collector (Wills & Napier-Munn, 2005).

2.1.1.3 Frothing Reagent

Frothers act on air bubbles in the flotation cell using a similar electrochemical mechanism to that of collector reagents. They reduce liquid surface tension in the slurry to promote bubble growth and stability, ensuring both sufficient surface area for minerals to interact with the bubbles, and sufficient bubble lifespan to allow successful removal from the top of the flotation cell (Wills & Napier-Munn, 2005).

Excess reagent will promote greater bubble growth, but may result in excessive froth overflowing the flotation cell weir or other operational issues in downstream processes.

2.1.1.4 Modifier Reagents

As the name suggests, modifiers change conditions within the pulp phase to ensure ideal conditions for separation of valuable minerals from waste. There are several different types of modifiers which may be categorised as follows:

- Activators – modify mineral surfaces to make them more accepting of collector reagents
- Depressants – modify waste surfaces to reject collector reagents more strongly
- pH regulators – control pH levels to balance flotation performance and equipment corrosion
- Dispersants – remove clay from mineral surfaces which may form slimes, hindering flotation
- Precipitants – induce dissolution of undesired ions in flotation pulp

Additionally, careful control of these reagents can allow for differential flotation, a process in which multiple desired minerals may be separated in sequential flotation cells without having to use other separation processes.

2.1.1.5 Agitation Rate

Bubble dispersion and liquid/gas interaction within a flotation cell are controlled mainly through mechanical agitation via an impeller. As impeller speed changes, so does the kinetic energy imparted to the pulp, changing bubble-particle interaction.

2.1.1.6 Air and Bubble Properties

Size, stability, and rate of bubble flow through the pulp phase are controlled by a combination of air flow rate and mechanical properties of the air distributor (sparger) within a flotation cell. Decreased air flow rate results in more stable froths containing a higher grade of desired minerals, at the expense of lowered mineral recovery and potentially lower mass throughput (Barbian, et al., 2005).

2.1.1.7 Pulp Particle Size Distribution

For a given ore, the particle size distribution must balance competing requirements of comminution costs, mineral liberation, and maximum size for flotation. The lower bound of particle size is determined by the size of mineral inclusions in the ore and cost of comminution. The ore should be crushed finely enough to allow for liberation of most valuable minerals within the ore. While there is no real limitation on further reducing particle size, capital and operating expenses increase strongly with smaller particle sizes.

The upper bound for particle size is that which can no longer participate in flotation due to gravitational forces acting on the particles.

2.1.1.8 Pulp Level

Froth height can be controlled directly by modifying pulp height within the flotation cell. Altering airflow and pulp height can ensure that requisite residence times are achieved in the pulp phase (Wills & Napier-Munn, 2005).

2.1.1.9 Temperature

Temperature has an impact on froth phase viscosity, in turn affecting bubble stability, and in some cases, can act as a depressant for minerals (Wills & Napier-Munn, 2005). However, slurry temperatures are generally not controlled due to energy requirements. Slurries may be warmer directly from comminution and cooler at later flotation stages, though overall process impact is small.

2.1.2 Fundamental Flotation Processes

There are several fundamental processes which occur in both the pulp and froth phases within a flotation cell. Inertial, gravitational, hydrodynamic, capillary, and surface forces all play a role which can be quantified through several effects.

2.1.2.1 Pulp Phase

Within the pulp phase, collection of desired minerals is determined by collisions, attachments and detachments of particles and bubbles. The more bubbles, particles, and turbulence within the cell the higher the probability of collision occurring between bubbles and particles. Each collision gives rise to the possibility of particles attaching or detaching from bubbles as they rise through the pulp.

The attached bubbles are affected by many intermolecular, interphase, and other physical forces. The chemistry within the pulp determines which particles preferentially attach and detach from the bubbles which allows separation to occur within a flotation cell (Nguyen & Nguyen, 2009).

2.1.2.2 Froth Phase

The froth phase is intimately linked to the pulp phase, though while effects within the pulp can lead to good separation of minerals to the froth phase, there are several paths of material transfer within the froth phase which can affect final recovery and grade of minerals in the froth overflow of a specific cell. The pulp phase chemistry has the largest impact on true flotation, as the particles attach selectively to bubbles due to hydrophobic effects.

Two linked processes in the froth phase are drainage, and entrainment. As the froth builds on the surface of the pulp, some pulp is captured (entrained) in the froth phase as bulk fluid which hasn't undergone frothing, ultimately reducing selectivity of the flotation cell. Some of this fluid, along with unattached particles on the bubble surface will drain back to the pulp phase. During drainage, some of the unattached particles may attach to bubbles as with true flotation, making this an important process in maintaining good selectivity in a flotation cell (Mathe, et al., 1998).

Finally, the froth moves out of the flotation cell and to the concentrate stream. This is controlled by a combination of air-flow rate and pulp level in a flotation cell to maintain a requisite residence time for the pulp phase components. Additionally, mechanical paddles may also be used to accelerate froth transfer, though their use is uncommon.

2.1.3 Froth Appearance

Bubble size, shape, velocity, stability, and mineral loading all affect appearance of the bulk froth. As the froth is the final output of the flotation cell, its appearance can be used to infer various process conditions in the flotation process (Moolman, et al., 1995).

Each of these visual characteristics are affected as follows:

- Bubble size – drainage from the froth phase and residence time
- Bubble shape – in addition to the bubble size effect, froth phase depth has a large impact on shape
- Velocity – directly a function of the rate of froth removal, and thus residence time
- Stability – strongly affected by pulp flow rate into the flotation cell

Lastly, there is also a colour factor apparent within the froth phase which is directly affected by minerals bound to the bubble surfaces. Combining bubble shape, size, and colour gives overall textural and spectral features which may be used for control and monitoring of to varying degrees of success (Aldrich, et al., 2010). Inferring grade or recovery from these characteristics is made difficult as they are influenced by other process variables.

2.2 Industrial Application

In concentrators, several flotation cells are used both in series and in parallel, grouped in banks, to achieve requisite grade/recovery profiles and mass pull rates. Mass pull is the fraction of concentrate output mass from flotation compared to total input mass. This is distinct from recovery and measures how much of the original ore passes through flotation, a lower fraction indicates higher selectivity. There is a practical limitation on flotation cell size which, coupled with the specific flotation process requirements, limits the throughput of a single flotation cell.

Each bank of cells consists of sequential flotation cells operating under similar conditions. Concentrate is passed to each cell in sequence, further increasing the grade of the concentrate. The first bank is known as the primary rougher. Here focus is placed on recovery, with most of the valuable minerals being separated from the feed slurry. The concentrate from the primary rougher is sent to the rougher cleaner bank, where requisite grade is achieved for the final concentrate. Ultimately the final grade is a balance between flotation expenses, total recovery, and smelting costs of the concentrate.

The tailings from the rougher sections may be sent to a further set of flotation cells, the scavengers. As with the roughers, there may be both primary and cleaner scavenger, with the same goals of first getting maximum recovery, and then requisite grade. Operating conditions though are optimised for lower yield slurries as most of the desired minerals have been removed in the roughers.

While it is possible to optimise each flotation bank to maximise grade, recovery, and mass pull, this is dependent on having a good understanding of changing operating conditions within each bank. In practise each bank is operated conservatively to ensure recovery, and mass pull targets are met under varying feed conditions.

2.3 Performance Monitoring

Research indicates that the appearance of froth from flotation cells may be strongly correlated with flotation cell conditions as shown by (Li, et al., 2016), (Marais & Aldrich, 2011), and (Moolman, et al., 1995). This makes a vision-based soft-sensor an attractive candidate for use in froth flotation. Texture analysis especially is attractive as it has been found to better correlate with process conditions than spectral methods which ignore spatial relationships and are less susceptible to changing imaging conditions (Liu, et al., 2005).

Process operators use appearance of flotation froths to determine operating parameters with some success (Shean & Cilliers, 2011). This provides further confirmation that relevant process information can be extracted from flotation froth images.

Individual properties such as bubble size, and velocity have been used in soft sensing systems such as Visiofroth™ (Metso, 2007)and FloCam (Mintek, 2015) for controlling flotation cell operating conditions such air flow and froth level (Sadr-Kazemi & Cilliers, 1997). These systems focus on monitoring and controlling

conditions within the flotation cell such as pulp level and air feed rates. Other process conditions such as grade and recovery have not been monitored or controlled with froth appearance in commercial applications even though research indicates this to be possible (Aldrich, et al., 2010).

2.3.1 Industry Requirements and Resources

Controlling flow rates and pulp levels are issues which have been investigated with varying levels of success in developing systems for froth flotation control. Industrial users of flotation cells require real time information of grade and mass-pull in concentration processes. Currently this may be done through on-stream analysers, but the expense and maintenance requirements often limit these to the output of a complete flotation bank, not individual cells. Image-based soft-sensors may be combined with existing online grade analysis to provide more information on individual flotation cells and perform additional tasks such as fault detection of in flotation cells that is not possible with other process sensors.

Availability of existing process data, and widespread use of froth flotation in local industry makes flotation an ideal candidate for soft-sensing. As shown in (Moolman, et al., 1995) and (Marais & Aldrich, 2011), useful process data may be extracted from textural features in flotation froths with varying degrees of success. Ideally, image-based soft-sensor systems would accept inputs from various upstream processes and other flotation cell operating set-points such as reagent dosage to maximise sensor capabilities.

The existence of extensive local industry allows professional relationships to be leveraged to gather further data for research into froth flotation. Drawbacks identified in current datasets through the course of this work can be corrected by communicating with industrial partners and obtaining additional new data through them.

Chapter 3: Soft-Sensing and Computer Vision

Soft-sensors allow process variables to be measured indirectly via existing sensors or with non-traditional sensing systems such as vision and audio systems. Such sensors can be very flexible under suitable conditions and allow for online measurement of process variables which would otherwise require offline measurement or more expensive direct sensors.

Soft-sensors do not require defined fundamental models linking measured and inferred variables, measured variables can be any process condition which is affected by the property to be inferred. A property such as process appearance which has a complex and undefined relationship with desired information may be used to develop a correlative model for monitoring and control (Kadlec, et al., 2009).

Intimately related to soft-sensing is the use of computer vision, a wide-ranging term encompassing computer processing of digital images, from acquisition to processing. It may also be referred to as machine vision, however the terminology is used interchangeably in literature. The heavy use of computer power to extract images under analysis in this work results in a preference for the term computer vision.

To provide relevant context for the selection of deep learning for sensing, this chapter will provide a broad overview of technologies and practices used in image-based soft-sensors for mineral processing.

3.1 Soft-Sensing

Online and non-intrusive measurement of key process variables is one of the hallmarks of soft-sensors. With the advent of cheaper electronics and increased computation power soft-sensor implementations and studies have increased considerably in the last 30 years. These systems are also known as soft-sensors in the process industry (Kadlec, et al., 2009).

Using measurements from standard process sensors such as pH, flow rate, temperature, or pressure additional process information may often be correlated when using a suitable model. In-depth information on soft-sensor implementation and use can be found in (Kadlec, et al., 2009), (Lin, et al., 2007), and (Kim, et al., 2013).

This contrasts with direct sensing methods which can only monitor a limited number of physical properties as determined by sensor technology. While of interest in many processes and the first step in control and monitoring, these values often do not indicate process performance unless interpreted with expert knowledge.

3.1.1 Soft-Sensing in Process Control

Online prediction of process variables, generation of additional process data, sensor validation, and process dynamics analysis are just a few of the applications of soft-sensing. The most common is online prediction of process variables.

Many process variables of interest are difficult to measure directly, and where direct sensors exist they are often expensive and too slow relative to the rate of change in the process under measurement. In online prediction, key process variables are accurately estimated in real-time or near real-time (Kadlec, et al., 2009).

These variables are more commonly known as key performance indicators and are often not apparent to standard direct sensors. Increasing measurement accuracy and speed leads to improved process control, which has a cascade effect on production efficiency, product quality, and profit (Kim, et al., 2013).

Process operator knowledge and judgement plays a key role in the smooth running of process systems and form a key part of the control process (Kluge, 2014). The knowledge and experience operators require are not trivial; estimating process performance directly from standard sensors is not possible for most processes. As processes become more complex and economic pressure grows larger, the human performance of operators is no longer sufficient, which is where soft-sensing provides an advantage.

Using data provided by soft-sensing, new forms of automation may be implemented to reduce control load on operators, reducing risk of erroneous control and improving process efficiency and safety. Inferred data may also be presented to the operator in lieu of standard variables. This abstraction of information can improve operator judgement and reduce cognitive load, albeit at the risk of hiding abnormal events which a skilled operator could otherwise identify. Any soft-sensor is only as good as the framework and data it was trained with, when presented with previously unseen or unknown process operating conditions, erroneous results may be produced by the soft-sensor.

Sensor validation is another use case for soft-sensing. Accurate process control is dependent on reliable sensor data, so for critical measurements some form of validation is commonly applied. Using a soft-sensor in this role allows for independent measurement and verification of a process variable allowing fault detection and diagnosis of both the process and the primary sensor. Often the soft-sensor may replace the primary direct sensor temporarily in the case of a fault (Fortuna, et al., 2007).

Lastly, the models developed for soft-sensors may be used in analysing process dynamics of the system by simulating scenarios of various process conditions and operation states. This may be artificially generated data or real process data gathered during unusual operating conditions or during fault conditions. This may lead to improved control methodologies as process responses are better understood from the results of scenario testing (Fortuna, et al., 2007).

3.1.2 Soft-Sensor Models

Inferential-sensors use one of two model types to correlate measured variables with desired variables: model-based, and data-based (Kadlec, et al., 2009). Vision-based sensors are inherently data-based as will be seen in the following definitions.

Model-based soft-sensors attempt to model the process being monitored in fundamental or phenomenological terms. These analytical solutions are often sufficient for steady-state models of low complexity. However, the assumptions and errors in defining such a model for a soft-sensor often limit the ability of the system to make accurate predictions under more realistic or complex process conditions, such as encountered in mineral processing systems (González, 2010). Often the analytical model may simply be too difficult to implement or computationally expensive to allow for economical real-time measurement.

In contrast to model-based sensors, data-based sensors seek empirical correlations between input and output variables. Using various statistical methods and historical data, an empirical model is derived for the sensor system. This bypasses the risks of errors and assumptions in model-based systems and is more effective at establishing correlations which may be unknown in current fundamental models. Data-based sensors are also capable of predicting process conditions where no fundamental models have been developed or are feasibly possible, such as vision-based sensors. However, data-based sensors can be sensitive to errors and bias in input data, which may require significant pre-processing to produce data that reflects the full range of process conditions accurately. Additionally, this also makes extrapolating for unknown process conditions riskier when compared to model-based sensors.

For these reasons, data-based or hybrid (data and model-based) approaches are preferred in industrial applications (González, 2010). Additionally, data-based models can be continuously improved (though also requiring continuous calibration) as more data is gathered and analysed (Fortuna, et al., 2007), mitigating some of the initial disadvantages of deploying a data-based sensor into a process. However, there is a trade-off between simpler soft-sensor models which require more frequent updating or more complex models which can be updated less frequently, but require more complex methods to develop and implement (González, 2010).

3.1.3 Developing Soft-Sensors

This review of soft-sensor development and implementation will be limited to data-based models, which are the focus of this study. Further information on model-based sensors may be found in (Fortuna, et al., 2007).

The first step in soft-sensor development, regardless of model type, is to identify the desired output, or inferred variables and potential input, measured variables. A generic framework for developing data-based soft-sensors is summarised in Figure 3-1. The identification and measurement of process variables pertinent to developing an accurate soft-sensor is often made easier by the significant volume of historical data collected in modern process plants. For older plants or in cases where data has not been gathered, the task of collecting relevant data is often more time consuming and expensive than developing the soft-sensor model itself (Kadlec, et al., 2009).

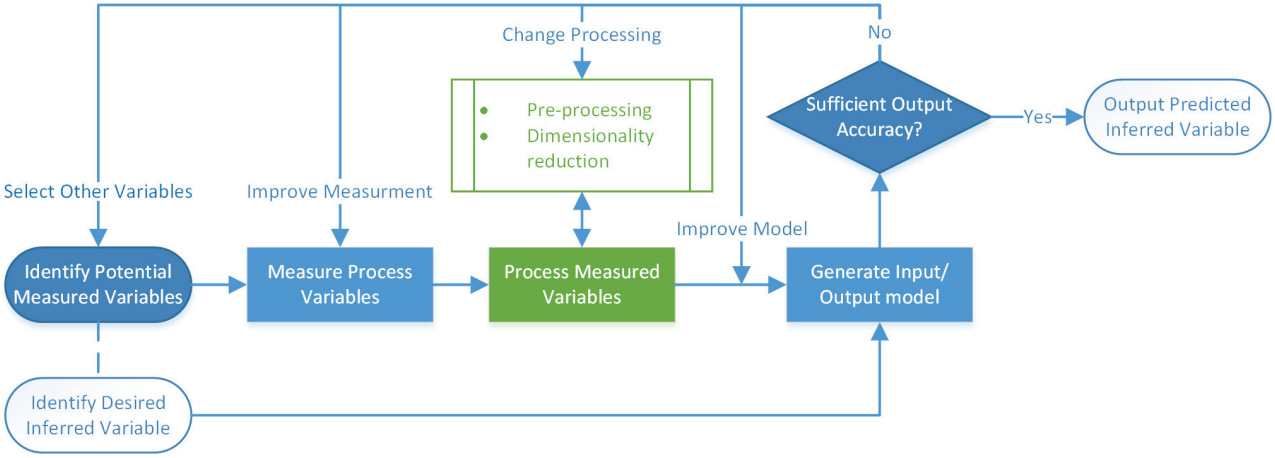


Figure 3-1: Development Framework for a Generic Data-Based Inferential Senor

Data fed into a soft-sensor during training is often pre-processed to minimise bias and correct for missing data which may affect later models, though this is not compulsory for all models. In some cases, later dimensionality reduction methods or specific soft-sensor models may be resilient to data deficiencies, thus pre-processing is skipped.

Final processing of the data is generally some form of dimensionality reduction when the number of measured variables is large, as is the case with vision-based systems. In this step, original data is transformed or mapped to new data that is smaller in size, but still representative of differences within the original dataset. This reduced representation (known as the feature set) is used to generate the final input/output model of the inferential-sensor with greater accuracy and speed than possible with the original measured variables. In this study, the capability of dimensionality reduction (specifically feature extraction from images) of various methods will be compared in the context of process engineering requirements.

One of the more common methods of analysis applied in process engineering is principal component analysis (PCA), which projects data onto axes of maximum variation (Elshenawy, et al., 2010). The most important of these principal components may be selected and the rest discarded as a form of dimensionality reduction. As these components are linear transforms of the original measurements, they do not necessarily have any effect on the separability of non-linear data (Jolliffe, 2014). For vision-based soft-sensors, other strategies are more commonly applied.

As vision-based soft-sensing is the primary component of this work, only data-based input/output models will be evaluated and discussed. There exists no fundamental model between appearance and conditions for most processes, or it is so complex as to be infeasible to implement. The input/output model may either perform regression or classification, depending on the requirements and properties of the desired inferred variable. The most common methods used in soft-sensors include: partial least squares (PLS), shallow artificial neural networks (ANN), and support vector machines (SVM) (Kadlec, et al., 2009). To train the input/output model, a linked dataset of the inferred variable and measured variables is required to learn the required classification or regression models.

The measured variables, pre-processing and processing methods, and input/output model are changed and improved until sufficient accuracy in the inferred variable is achieved for monitoring and control tasks.

3.2 Computer Vision

Digital images are stored as a grid of points representing luminous intensity at each point, more commonly known as pixels. In colour images, each pixel consists of three channels with luminous intensities for the primary colours red, green, and blue, as shown in Figure 3-2. The three channels in the image form the colour space of the image, in this case RGB colour space. Other mappings exist such as HSL (Hue, Saturation, Lightness), and LAB (Luminance, A and B colour components). However, these other colour spaces are intended to improve image editing by presenting image channels as they are interpreted by human vision and have little to no effect on computer vision algorithms (Sonka, et al., 2015).

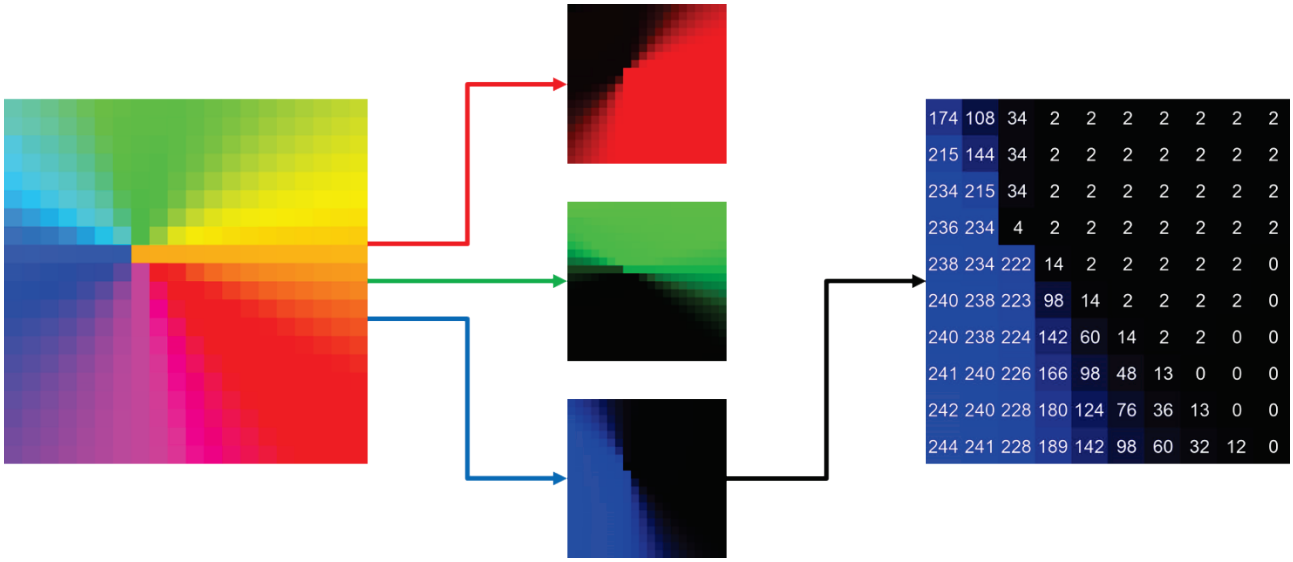


Figure 3-2: Breakdown of Information Storage in an 8-bit RGB Image

Spectral bands other than red, green, and blue may be detected by vision systems, such as ultraviolet and infra-red, which may provide additional information not apparent to the human eye. Additional optical phenomena such as polarisation may also be recorded, but is outside the scope of most computer vision systems and this work.

In standard consumer imaging systems, the luminosity values are represented as 8-bit values, leading to 256 distinct intensity levels from 0 (darkest) to 255 (lightest). For the purposes of computer vision, each pixel is considered a variable which itself may be monochrome or consist of multiple spectral values for colour images. Unlike other multivariate data sources in soft-sensing, images are highly correlated across a 2-dimensional (3-dimensional if including colour channels) field, requiring specialised methods of feature extraction. These are known as textural features and are a function of spatial relationships within images (Haralick, et al., 1973).

If the spatial information is not considered important by later analysis or bulk spectral information is required, the images may be unrolled into vectors. For, example a 16x16 pixel colour image would unroll to 768 variables. A 32x32 pixel image would unroll to 3072 variables. This demonstrates the rapid growth in variables with image resolution which requires the use of feature extractors in vision-based soft-sensors.

The ultimate goal of computer vision is to detect and extract information which could have been obtained by a human physically observing the process and additional information which would not be apparent to a human observer. As a result, many computer vision implementations, including deep learning, attempt to recreate biological models of vision. Computer vision systems have improved with increasing computational power and can qualitatively and quantitatively outperform humans in several tasks (He, et al., 2015), (Ciregan, et al., 2012). However, computer vision systems remain limited to specific applications and cannot match the flexibility of humans (Yu & MacGregor, 2004).

3.2.1 Computer Vision in Soft-Sensing

Most computer vision systems are a specialised form of soft-sensors where images form multivariate inputs to a soft-sensor. They are inherently data-based due to the extreme difficulty of developing accurate fundamental models linking image appearance with process conditions. As with any other soft-sensor input, pre-processing and processing steps must occur before the input/output model is developed.

Starting with the imaging device, images must be fed into the soft-sensor under potentially demanding industrial conditions. Traditionally it has been important to maintain uniform and consistent conditions under which to capture images. However, in more advanced processing methods such as deep learning, this may no longer be the case with sufficient data to train such systems to compensate for changes during image acquisition (LeCun, et al., 2004).

Common pre-processing tasks for images include: normalisation, histogram equalisation, and channel extraction. Only certain spectral channels in conjunction with contrast enhancement may be selected for further processing. This is especially important for most traditional feature extraction methods but is not as important for methods such as deep learning which can incorporate such actions within the model because of its structure and training process.

As previously mentioned, the high dimensionality of images imposes the requirement of feature extraction when used in soft-sensing. Most input/output models for soft-sensors perform poorly with many input variables and often take substantially longer to train the larger the input dataset is. Dimensionality reduction must be able to extract useful information from the image data while discarding as much extraneous information as possible. Generally, two types of features are extracted from the original data: spectral and textural features.

Spectral features only consider bulk parameters of pixels in an image, such as mean values for each colour channel. They do not consider spatial relationships, allowing more rapid processing with methods such as PCA. Their performance is often satisfactory under controlled conditions, but due to the spectral nature, the features are very dependent on imaging conditions (Vathavooran, et al., 2006).

Conversely, textural features are strongly dependent on spatial distributions within input images. In order to fully exploit these and intelligently reduce dimensionality, it must be ensured that pre-processing preserves spatial properties in all images. Traditional textural feature methods such as local binary patterns (LBPs), grey-level co-occurrence matrices (GLCMs), and wavelets are suited to specific image types, with no method universally applicable to all images under all conditions (Yu & MacGregor, 2004). Their parameters directly relate to spatial properties within the input images and good performance is dependent on expert selection of them.

Textural and spectral methods are generally limited to only analysing one of the feature types, discarding the other potentially valuable process information. However, methods such as deep learning networks may, through training, inherently consider both textural and spectral information to produce reduced feature sets. More detailed information on textural features will be available in the next chapter.

Regardless of the dimensionality reduction method used, the reduced feature sets are fed into the input/output models the same way low dimensional data would be fed directly into a soft-sensor without feature extraction. No unique input/output models are required outside of the training required for all soft-sensors.

3.2.2 Computer Vision in Industry

There are several industrial applications of computer vision systems which can be divided into four broad categories: defect detection, like finding abnormalities in products such as textiles (Özdemir, et al., 1998); object detection, determining if an object or shape is of interest in autonomous systems (Oquab, et al., 2014); classification, discriminating between several possible classes for an image such as detecting if there are or aren't faces visible in them (Chan, et al., 2015); and regression, continuous response to input images such as classifying mineral properties (Marais & Aldrich, 2011).

The last two methods show good applicability to mineral processing and make computer vision an attractive tool for process engineers (Stanford, et al., 1992). The appearance of solids and mineral rich slurries has been shown to provide information on current process conditions in mineral processing (for example, (Kistner, et al., 2013)). One of the primary objectives of this work is to implement an image-based soft-sensor using deep learning in a mineral processing context.

Several commercial systems have been developed for the minerals processing industry in specialised applications such as mineral detection in slurries and froth flotation bubble feature detection (Barbian, et al., 2005).

Froth flotation was selected as the application for a deep learning system in this study. As with other metallurgical concentration methods, there are two primary measures of economic performance: grade and recovery (of valuable minerals in the concentrate output). Grade and recovery are linked via a curve specific to the mineral composition and process under evaluation. The operating position of the process on this grade/recovery curve is difficult to ascertain quickly with traditional sensors. In addition, the shape of the cure may vary with changes in process conditions. Online measurements are preferred for these properties.

Recovery cannot easily be measured online as the quantity of the specified mineral must be known both in the input and final output streams of the entire process. This can typically be achieved only as part of a steady state mass balance after assay of all the process inputs and outputs. In contrast, grade is more easily measured online as it is defined as the fraction of valuable minerals in the total product stream under measurement (Duchesne, 2010). Grade will be the key performance indicator measured by the deep learning soft-sensor developed in this work.

Chapter 4: Image Feature Extraction

Feature extraction, as introduced in the previous chapter, is a method of dimensionality reduction which maintains meaningful information of the original input data for use in various applications. While it can refer to any method of dimensionality reduction, within the context of this work, feature extraction refers to (primarily textural) methods to reduce dimensionality of images for use in process sensing and monitoring.

This chapter will focus primarily on textural methods of feature extraction as those are likely to be more robust for process engineering applications. Textural features are complex and contextual image properties, but at their simplest, may be considered representations of repeating spatial organisation within an image (Xie, 2008). Textures vary from image source to image sources and few methods are suitable for all image sources when using the same extraction parameters.

Older methods of texture feature extraction require assumptions about structural content of images and have a number of parameters which must be carefully selected. Conversely, deep learning methods as discussed in the next chapter, are more flexible and do not make assumptions of image properties, rather they are learned inherently during training (Baldassi, et al., 2016).

4.1 Approaches to Texture Feature Extraction

The approaches selected for texture extraction vary based on the types of images to be analysed and the task which is to be achieved. Most tasks in texture analysis can be classified as one of the following:

- Segmentation – detecting and dividing images into areas of similar textures
- Synthesis – identifying and modelling the fundamental structure of textures to recreate them
- Classification – detecting and categorising images into pre-defined classes

Segmentation commonly uses structural approaches to model texture properties in terms of fundamental features. These texture primitives (fundamental texture components) may be areas of uniform brightness, line segments, or other simple geometric shapes. Textures are defined by the spatial relationships and statistics of these primitive features which can then be detected and output as feature sets. The feature sets may be used directly as outputs in terms of the fraction of the image covered by a certain texture or they may be used as an input in another texture processing step, such as identifying areas of interest for other texture extraction methods.

Synthesis is more complex and has not been used in vision-based soft-sensors. Feature or texture synthesis involves the extraction of a reduced feature set, and the reconstruction of a texture from the reduced features. While a reconstructed texture is not much use in a soft-sensor, it is a powerful tool for validating reduced feature sets. By sufficiently reconstructing an arbitrary texture from a reduced feature set it can be shown that the reduced feature set contains critical information on the texture itself. Little to no information would be lost by using the reduced feature set instead of the complete image in a soft-sensor framework.

Several approaches may be followed in texture synthesis, though the method of feature reduction is not as important as the texture reconstruction method. Autoencoders are a special class of neural networks which are particularly suited to synthesis, operating on the principle that the same parameters used to reduce the data can be used to reconstruct it, simplifying the task of reconstruction (Baldi, 2012).

Classification is commonly coupled with statistical and transform based approaches and is one of the most useful tasks in image processing. Classes are defined, and images are sorted into these classes on a statistical basis using methods such as LDA and SVM.

These statistical methods may either rely on first order or higher order features. First order methods ignore spatial relationships and thus are not considered a form of texture analysis within the definition of this work. Higher order analysis considered and preserves spatial information in images by considering multiple pixels at any single time. These methods will be discussed in the rest of this chapter. Deep learning methods share some characteristics with both first and higher order methods, however, such distinctions are not easily applied to their functionality as it varies with implementation and trained network structures.

Transform based methods rely on filtering of images to reduce dimensionality and highlight certain features in the image. Through application of suitable filter matrices, transform methods can perform many of the goals of segmentation very efficiently; an example is the identification of edges using Gabor filters (Mehrota, et al., 1992). Similarly, the input stages of some deep learning hierarchies such as convolutional neural nets act as filters and detect high level features within images (Krizhevsky, et al., 2012).

Sensitivity to rotation or scaling of features is an important quality of feature extractors. Extractors have varying sensitivity to this change in input images and whether they should be insensitive to them is dependent on the application. When a feature extractor is insensitive to changes in feature scale or orientation it is known as invariant to that property. Bubble size in flotation froths has been shown to only weakly correlate with grade, and orientation of bubbles is likely to be random within the imaging field (Aldrich, et al., 2010). A feature extractor invariant to scaling and rotation would likely be ideal for froth flotation as it would allow consistent performance regardless of camera mounting on the flotation cell or directional effects within the flotation froth.

4.2 Methods of Texture Feature Extraction

The methods of feature extraction focussed on in this chapter can best be grouped as deterministic methods. For a given set of parameters and input images, the same set of features can be expected with every implementation. Deep learning methods discussed in the next chapter can be categorised as stochastic. A trained neural network will have reproducible outputs, but when retrained with the same hyperparameters, slightly different feature sets are likely to be output. This is due to random initialisation of network weights and numerical stability within training algorithms.

Some commonly used texture feature extraction methods include:

- GLCM – grey-level co-occurrence matrices
- Wavelets
- LBP – local binary patterns
- Textons
- Steerable pyramids
- Ranklets
- Granulometry
- ANN – artificial neural networks (including deep nets)

This project will focus on comparing three commonly used methods of texture feature extraction - GLCM, LBP, and wavelets - with newer deep learning based methods, specifically convolutional neural networks. A brief summary is provided for the selected traditional methods, though more in-depth descriptions can be found in the works of (Prasetyo-Wibowo, et al., 2010), (Xie, 2008), and (Kistner, 2013). All the methods summarised below require the input images to be converted to greyscale during pre-processing, this was done by taking the average of the three channel intensities as a single output for each pixel.

4.2.1 GLCM Extractors

GLCM is one of the more widely used methods for texture extraction and measures the spatial dependency of two grey-levels at a specified distance between them. These are gathered into 2D matrices which represent spatial dependencies across the entire input image. From these, one or more of the following texture features can be determined: energy, entropy, contrast, homogeneity, and correlation. GLCMs are easy to implement though they have some drawbacks in terms of computational efficiency and there are difficulties in determining optimum values for the method's hyperparameters. They are often combined with other filtering methods due to their poor performance on most image processing tasks (Xie, 2008). However, they have shown promise in froth flotation applications (Kistner, 2013).

The number of grey-level pairs considered may be up to the maximum available in an image (256 for the most commonly used 8-bit RGB format). However, the grey-levels are usually scaled to increase the probability of grey-level combinations existing within a local area in an image. Scaling essentially bins the pixel values into a narrower range, such as the four bins shown in Table 4-1.

Once this is done a co-occurrence matrix is generated by counting the number of neighbouring pixels that are the same within a specified area and for a specified direction, an example of which is shown in Table 4-1. The matrix is made symmetrical by adding it to its transpose, which solves the problem of pixels on the edges of area under consideration not having valid neighbours for some specified directions.

Table 4-1: Empty Grey-Level Co-Occurrence Matrix for a 5 Grey-Level Image for a Single Direction

		Neighbour Pixel Value (binned)				
		0	0.25	0.5	0.75	1
Reference Pixel Value (binned)	0					
	0.25					
	0.5					
	0.75					
	1					

This matrix is normalised to express a probability for pairings, which can be called the GLCM. These GLCMs are fed into the previously mentioned texture feature methods. Their functionality is summarised as follows:

- Contrast – emphasises contrast features within the image, may also be called sum of squares variance or inertia within the measured area
- Homogeneity – or the inverse difference moment, finds areas of low contrast within the image, essentially the opposite of the contrast method.
- Energy – calculates the angular secondary moment of the GLCM and is a measure of how similar or dissimilar images are within the area measured and represents increased order in an image
- Entropy – is within the context of GLCMs the opposite of energy and is a measure of noise or rare outlier pixel values within a measured area
- Correlation – measures linear dependencies between pixels in a measured area, the higher the value, the higher the probability of predictable relationships between neighbouring pixels

A tutorial on GLCM implementation may be found in (Hall-Beyer, 2017).

4.2.2 LBP Extractors

Local binary patterns are like GLCMs in that greyscale levels are compared. A central pixel of a sliding window is compared with neighbouring pixels from which a weighted sum is produced for the region. The distribution of these weighted sums provides the texture features in this method. It is relatively rotation invariant and is computationally simple to implement. It is effective when utilised in classification tasks, however, it may have inferior performance in defect detection tasks compared to GLCMs (Monadjemi, 2004)

A specified pixel is compared to all pixels in a predefined neighbourhood, in the original implementation of GLCMs these were the 8 pixels immediately around the specified pixel. Neighbours that have an intensity higher than the specified pixel are assigned values of one, those lower, zero. Weights are mapped to these neighbours which sum to a specified LBP level if all pixels passed the threshold of the specified pixel. The actual summed output is the LBP value. The process is performed stepwise across the entire image to produce the LBP image.

A histogram is produced of the LBP image and the number of occurrences of each LBP value forms the final feature set. When an arbitrary number of neighbours and neighbourhood radiuses are defined, artificial neighbours are created by weighted interpolation of the nearest real pixels.

While conceptually simpler and easier to implement programmatically than GLCMs, LBPs can be much more computationally expensive when there is a large number of LBP levels specified.

4.2.3 Wavelet Extractors

Wavelet transforms are a specialised application of Fourier transforms which preserve spatial data in a similar manner to applying a windowed Fourier transform. Fourier transforms and their related functions usually act on the frequency-amplitude/time domain, for 2D image analysis spatial positions are correlated to time and intensity is coupled with frequency-amplitude. This leads to images being treated in a spatial-spatial-intensity domain with a small computational penalty.

This method is particularly suited to defect detection as it is sensitive to spatially repetitive textures. The relative computational ease of calculating wavelets and their wide variety of prototype functions allows them to be optimised for a wide variety of processes relatively easily and as such are used in a wide variety of image processing tasks (Xie, 2008). However, in prior work by (Kistner, 2013) the optimal wavelet types for froth flotation images were found to not be invariant to rotation and translation of input images. In contrast, the expectation is these variances may be problematic in more random images such as flotation froths.

More advanced wavelet types are capable of forming invariant representations of images (Kingsbury, 2006), but those are beyond the scope of this work as the traditional feature extractors will be based on reference implementations from prior work by (Kistner, 2013) on a similar froth flotation dataset.

4.3 Texture Feature Extractors in Industry

Wherever vision-based soft-sensors are used it is highly likely that some form of texture feature extraction has been applied. Texture feature extractors are more robust than simple spectral dimensionality reduction methods and provide more information relevant to visual changes in most mineral processes (Vathavooran, et al., 2006).

GLCM methods have been successfully applied to many mineralogical processes, including flotation systems in zinc flotation (Marais & Aldrich, 2011). However, wavelet systems have shown superior performance in froth monitoring applications than GLCMs (Liu & MacGregor, 2008). Both methods have been applied to defect detection with success and are often combined as complimentary methods to increase classification and detection accuracy in a variety of methods as shown in (Latif-Amet, et al., 2000), (Fathi, et al., 2012), and (Bharati, et al., 2004).

Application of LBPs in industry have mostly been limited to surface inspection and discriminating between different surface types, with the earliest industrial applications being surface defect detection (Pietikaeinen, et al., 1994).

Chapter 5: Artificial Neural Networks

Artificial neural networks are a blanket term for a number of methods which attempt to mimic biological decision-making systems. The cornerstone and simplest building block of all neural networks is the artificial neuron, which is inspired by the biological counterpart. They have multiple inputs, a single output, and an activation function determining what output to produce based on the inputs.

They may operate alone or in conjunction with millions of other neurons in deep neural networks to perform increasingly complex tasks. Their use has increased dramatically in the past decade with emergence of powerful computer systems and new methods of training these networks to intuitively perform tasks.

It is the capability to be trained with sufficient datasets which makes artificial neural networks most promising. Other statistical methods may be suited only to linearly separable data or require additional hyperparameters to be specified to separate non-linear data. However, ANNs will automatically, through their training and standard hyperparameters, discover relationships between data and their labels without human intervention or knowledge of the data structure.

This powerful correlative ability comes at the risk of overfitting, where the large number of internal parameters may encode the data being trained with instead of discovering underlying relationships in the data (Goodfellow, et al., 2016).

5.1 Artificial Neurons and Multilayer Networks

Artificial neural networks have their basis in research from the 1940s which attempted to emulate the functionality of biological nervous systems (McCullough & Pitts, 1943). The first simplified neuron as defined by them is summarised in Figure 5-1, which outputs either true (1) or false (0) (in initial implementations known as the perceptron) based on a weighted sum of all inputs to the neuron.

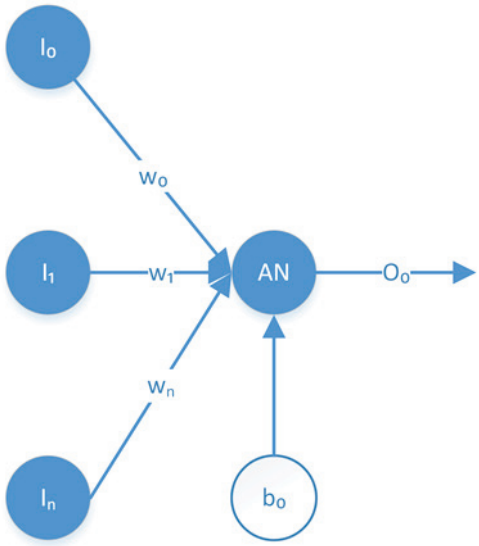


Figure 5-1: Artificial Neuron

Each neuron consists of the following basic components:

- One or more inputs, $I_i, i = 1, \dots, n$
- Weights associated with each input, W_n
- A bias term, b_0
- Some activation function which acts on the summed inputs, $f(x)$
- Output of the activation function, O_0

The factors and the representation in Figure 5-1 may be expressed as the equation below:

$$O_0 = f\left(\left[\sum_{i=0}^{n} W_i \cdot I_i\right] + b_0\right) \qquad (5-1)$$

Modern neurons have more advanced activation functions which are generally continuous and may have limit values other than simple true (1) or false (0). The number of weights, W_n, grow exponentially as the number of neurons in the network increases, leading to the previously mentioned risk of overfitting.

Though artificial neurons are not particularly useful on their own, when paired with multiple neurons in multiple layers, neural networks can be created. Neural networks may consist of multiple layers in which all neurons in a layer are connected to all neurons in the previous layer, the earliest implementation of this being the multilayer perceptron (Rosenblatt, 1962). Within layers, neurons are independent of other neurons in the same layer and are not interconnected.

The deep interconnection between layers allows very complex actions to be performed by the neural network. The computational power is limited only by the number of neurons in each layer and the total number of layers. A simplified representation of a neural network with one hidden layer is shown in Figure 5-2Figure 5-1.

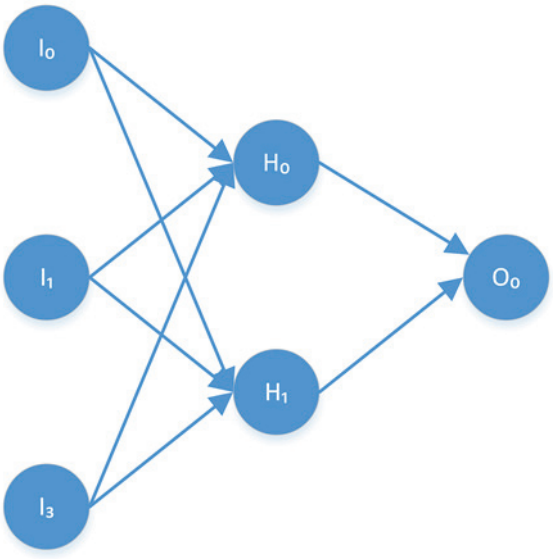


Figure 5-2: Simplified Neural network

For the simplified neural network depicted in Figure 5-2, inputs are modelled as neurons, represented as I_0 through I_3. These connect to two hidden neurons, H_n, which finally connect to an output neuron, O_0. This is an example of the simplest form of artificial neural network, a single hidden layer fully-connected with the input and output. The hidden and output neurons each perform the action defined in Equation 5-1, leading to an overall equation as shown below in Equation 5-2.

$$O_0 = f\left(f\left(\left[\sum_{i=0}^{n} W_i^{H_0} \cdot I_i\right] + b_{H_0}\right) \cdot W_{H_0} + f\left(\left[\sum_{i=0}^{n} W_i^{H_1} \cdot I_i\right] + b_{H_1}\right) \cdot W_{H_1} + b_{O_0}\right) \qquad (5-2)$$

As can be seen by Equation (5-2), the complexity of the network and the number of weights has increased significantly due to the fully connected nature. While this is advantageous for complex modelling purposes, when dealing with high dimensional input data the number of weights and biases may become excessive, making training unacceptably slow and overfitting a high risk. This rapid growth of network complexity is known as the curse of dimensionality.

Deep neural networks, which consist of many layers with many neurons in each layer, may employ different strategies for interlayer connections, which mitigates the curse of dimensionality.

5.1.1 Activation Functions

A defining characteristic of the artificial neuron is that is has a non-linear response to its input. The simplest response is a step change in response to a threshold being passed. However, several possible activation functions are possible as summarised in Table 5-2. The importance of an activation function being analytically differentiable is of considerable importance when it comes to training neural networks. The act of feeding information into a neuron and reading its output is known as forward propagation (Goodfellow, et al., 2016).

The most commonly used activation functions are sigmoid and hyperbolic tangent, while rectified linear unit (ReLU) activations are commonly used in more complex, convolutional neural networks. Selection of an activation function is determined by several factors, most being strongly linked to training performance and will be detailed in the next section. At the very least, most practical implementations of neural networks require that the activation function be continuously differentiable.

Table 5-1 : Softmax Activation Function Summary

Equation	Derivative	Output Range	First-Order Derivative Continuous?
$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$	$\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})\left(\partial_{ij} - f_j(\vec{x})\right)$	$(0,1)$	Yes

An activation function distinct from those listed in Table 5-2 is the softmax or normalised exponential function as summarised in Table 5-1. It acts across an entire layer of neurons, calculating a probability for each neuron in the layer. The summed probabilities of all neurons in the layer will equal one.

Table 5-2: Activation Functions for Artificial Neurons

Name	Equation	Derivative	Output Range	First-Order Derivative Continuous?
Identity (linear)	$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	Yes
Binary step	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \; for \; x \neq 0 \\ ? \; for \; x = 0 \end{cases}$	$\{0, 1\}$	No
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)\big(1 - f(x)\big)$	$(0,1)$	Yes
Hyperbolic Tan	$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	Yes
Arctangent	$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	Yes
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$	$[0, \infty)$	Yes
Sinusoid	$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	Yes

When combining a suitable activation function with multiple neurons (as in Equation 5-2) the neural network can approximate arbitrary functions. This is what leads to the strong correlative capability of neural networks.

5.1.2 Training and Backpropagation

In normal operation, a neural network operates by forward propagation. Data is input and cascades through the layers of the neural networks until the output layer produces some result, akin to a classifier or regressor. To train the network, backpropagation and stochastic gradient descent are utilised.

Training the network requires updating the weights in response to error in the output. The normal output of the network is taken, and a loss (error) function is applied to it. Once the function is applied, the network weights are differentiated with respect to it. This process, called backpropagation, results in an objective function with respect to each weight and bias in the network (Gallant, 1993).

The network with loss function applied forms a complex high dimensional surface (one dimension per weight or bias). The surface represents loss values at all possible combination of weights and biases, to which gradient descent is applied to find the minimal loss. This gradient, as calculated by backpropagation, shows the direction in which the weights must change to reach the closest minima of loss. However, depending on the shape of the loss surface, this may not be the true minima. Several strategies are employed to ensure that gradient descent methods do not get stuck in local minima, hindering training performance. These strategies are discussed in further detail in the next section on deep learning.

For backpropagation to complete successfully, the activation functions applied to the neurons should adhere to the following requirements in addition to those mentioned previously:

- Finite output range
- Monotonic input response
- $f(x) \approx x \; when \; x \approx 0$

These properties allow training to successfully occur with random initialisation of all weights and biases. If the system is not differentiable at all points or if a saturated output (zero gradient at neuron) is presented at a neuron, it may not be possible to calculate the gradient for the weights of that neuron relative to the error in the output. As network size increases, speed of calculating the derivatives also becomes a concern.

The network is now ready for training in a supervised manner with labelled data. Data is input to the network and gradient descent through backpropagation occurs to calculate updated weights. Originally, each weight would be updated independently, and the backpropagation results would be analysed again, but most modern implementations update all weights and biases in the neural network under training simultaneously (Gallant, 1993).

The simplest and most common method applied at this stage to improve training performance is batch training. Multiple independent inputs are fed into the network and produce an averaged backpropagation result with which to perform weight updates with. By averaging the gradient direction over a larger number of inputs, a smoother loss surface can be calculated which more realistically represents the data correlations being trained for (Hecht-Nielsen, 1988). This lowers the risk of training getting stuck in local minima and greatly increases training speed of the neural network. However, batch sizes which are too large often converge very slowly to the global minimum due to contributions from many weights resulting in large single weight changes having a small overall effect.

5.1.3 Advantages and Disadvantages

Artificial neural networks are capable of modelling complex relationships without requiring significant human knowledge or intervention. The training of neural networks inherently explores and discovers relationships within datasets and give good results in many applications (Özdemir, et al., 1998), (Al-Thyabat, 2008), (Marais & Aldrich, 2011).

However, their application is limited by the curse of dimensionality inherent in their design. Artificial neural networks in the fully-connected form scale poorly to high dimensional data such as images. The large number of weights and biases cause two primary issues: they make the network computationally expensive to train and increase the risk of overfitting.

Overfitting occurs when the number of weights and biases in a network is relatively large compared to the number of data samples used to train the network. Instead of learning the underlying structure of the data

used for training, the neural network encodes the training data directly, leading to poor performance with unseen data and poor generalisation to related data (Srivastava, et al., 2014).

Deep neural networks mitigate many of the disadvantages of smaller, shallow neural networks.

5.2 Deep Neural Networks

A deep neural network is, by the simplest definition, any artificial neural network with more than one hidden layer of neurons. While this is true for some deep neural networks, the term is more commonly used to refer to networks with specialised interlayer connections.

Deep neural networks can abandon the fully-connected strategy of shallow neural networks and apply hierarchies specific to the general data properties being analysed (Krizhevsky, et al., 2012). This allows the curse of dimensionality to be mitigated or allows time-based data to be analysed relative to other data in the sequence. While this does reduce the generality of deep neural networks compared to shallow networks, the dramatic increase in performance for specific applications is considered an acceptable trade-off (LeCun, et al., 2015).

It is only with the advent of modern computational resources and advances in network training techniques that deep neural networks and other deep learning methods have become viable tools.

5.2.1 Deep Network Architectures

The increased computational difficulty of training deep neural networks typically means that a limited set of activation functions are selected for the neurons in them. Sigmoid and hyperbolic tangent functions are popular as their derivatives can be easily calculated from the previously calculated forward propagation values as shown in Table 5-2. Certain implementations such as convolutional neural networks include use of specific activation functions such as ReLUs, these implementations will be detailed further in the following sections.

Some common deep network architectures will be discussed in this section. Convolutional neural networks which form a core part of this work will be discussed in a later section.

5.2.1.1 Training Deep Neural Networks

As with shallow neural networks, one of the largest challenges in deep learning is efficient training and avoiding overfitting. Deep neural networks tend to get stuck in local minima while training, but there are several methods to avoid this (Bengio, 2013).

Fundamentally, backpropagation becomes more difficult as neural network size increases. Gradients become small, resulting in prohibitively long training times. Derivative calculations are limited by computational accuracy of the internal representation of weight values within the neural network, once pass a certain

threshold, errors in the weight values may lead to sudden and erroneous large gradients and associated weight updates which saturate neurons and prevent further updating of weights. This is known as gradient collapse (Schmidhuber, 2015).

Improved gradient descent methods and optimisation strategies can help mitigate these effects (Goodfellow, et al., 2016). Batch normalisation is an especially effective optimisation method for deep neural networks. During weight training procedures all weights are updated simultaneously, despite individual weight updates being calculated under the assumption that only that weight will change. With small networks this does not have a significant impact on training, but with deep networks, the large number of weight changes has a cascading effect of overcorrection within the network. Batch normalisation mitigates this effect by limiting the rate of change in neuron activations between subsequent layers. This is achieved by normalising the activation outputs in an entire layer through Equation 5-3. The activations, H, are normalised (H') through a mean and variance term.

$$H' = \frac{H - \mu}{\sigma} \tag{5-3}$$

As Equation 5-3 forms part of the neural network, it also takes part in backpropagation. Thus, the mean and variance are calculated at each training step from the activation outputs for the specified layer as indicated by Equations 5-4 and 5-5. The δ term in Equation 5-5 is a small constant added to ensure that the case of an undefined gradient doesn't occur when the calculated variance is zero, m is the number of activation outputs in layer being normalised.

$$\mu = \frac{1}{m}\sum_i H_i \tag{5-4}$$

$$\sigma = \sqrt{\delta + \frac{1}{m}\sum_i (H - \mu)_i^2} \tag{5-5}$$

By calculating the mean and variance at each calculation step, training can adaptively correct for overly large changes in neuron activation. This moderates the effects of weight updates on each layer, allowing them to have a meaningful effect on network performance and prevent neuron saturation.

Dropout is also commonly applied to fully-connected networks to improve training and final performance. A fraction of neurons in the targeted layer are randomly deactivated during each training step, decreasing the number of weights available for training and thus reducing the risk of overfitting (Srivastava, et al., 2014).

These optimisation methods also result in some form of regularisation in the network - they improve generality without only improving training performance as well (Goodfellow, et al., 2016). During training the network objective function has a direct goal of reducing classification or regression loss in the training dataset. Regularisation methods modify the error function (or network weights themselves) to penalise

overly good fits of the training data, it is expected that reducing the training loss in this manner will increase network performance when encountering new data. Regardless of operation, these methods force network training to learn underlying correlations, and not merely encode training information.

Statistical methods are generally trained via either supervised or unsupervised methods. Supervised methods have an objective function defined by the error between actual data labels and predicted data labels, directly learning for the problem but requiring labelled data. Unsupervised methods utilise unlabelled data and attempt to generate meaningful distinctions without external guidance, this may be achieved through clustering methods or generative models which use reconstruction error as an objective function. In deep neural networks, supervised methods have been extensively used due to success with large datasets and overfitting mitigation strategies with smaller datasets. The labelled data provides a clear objective for gradient descent methods to update network weights.

Datasets are labelled with classes or continuous labels which can be directly used as objective functions when training networks. Small datasets may be enhanced by creating variants from the original data with the same labels. These variants include rotation and scaling of the original images, reducing the risk of overfitting while also making trained networks more resilient to scaling and rotation of input images in trained implementations (Dosovitskiy, et al., 2014). Variants attempt to increase the representative range of input datasets by emulating a variety of conditions. If data mostly represents a single condition, the risk of overfitting increases, and deep neural networks are already more susceptible to overfitting than shallow neural networks or other statistical methods. However, most well-known implementations of deep learning in other fields were only made possible by the availability of large datasets for domain specific problems (Srivastava, et al., 2014).

Unsupervised methods are less common in neural network applications. Specific hierarchies known as auto-encoders are commonly used when insufficient labelled data is available. These networks attempt to recreate the input data, using a reconstruction error as a target for the training objective function. These hierarchies consist of symmetrical layers around a central bottleneck layer. This layer forces a reduced representation of the input data from which the input is reconstructed. Once trained the reconstruction layer is discarded and the bottleneck layer is used as an output to further classification methods (neural network based or otherwise). However, the symmetry requirement of these networks causes them to suffer strongly from the curse of dimensionality (Van Der Maaten, et al., 2009).

A third approach is semi-supervised learning. These methods include using generative models such as Restricted Boltzmann Machines (RBM) to train on unlabelled data then transfer the weights to a deep network for final training with a limited, labelled dataset. Other statistical methods such as principal component analysis or t-Stochastic Neighbour Embedding may be used to generate artificial, pseudo-labels for unlabelled data which can then be fed into a deep neural network. Once trained on the artificial labels,

the network may be briefly retrained with limited labelled data to achieve classification or regression with real data (Dosovitskiy, et al., 2014), (Lee, 2013).

5.2.1.2 Training with Gradient Descent Methods

The optimisation functions used for training deep neural networks are classified as gradient descent methods. As previously described, these algorithms follow the gradient established by backpropagation to adjust network weights and minimise training loss. The rate at which the weights are updated is modified by a scaling term applied to all weight gradients known as the learning rate. Depending on the gradient descent method used, depends how the learning rate is applied and changed as training progresses. The simplest gradient descent method is stochastic gradient descent.

In stochastic gradient descent (SGD), a fixed learning rate is applied during training to all weights Weight changes are calculated through backpropagation as the average of a small random (stochastic) selection of examples from the training set, allowing better scaling with large datasets. A fixed scaling (the learning rate) is applied during training to the calculated weight changes until sufficient performance is achieved or training is stopped. While effective when correctly implemented, SGD is sensitive to the selection of learning rate. During initial training, a large learning rate is desired to rapidly change network weights, as the network may be far from the minimum loss state and training may be very slow with a small training rate. Additionally, small training rates run the risk of getting stuck in local minima and not achieving optimum training results (Goodfellow, et al., 2016). However, large training rates often fail to converge to the global minimum, instead moving either side of it as the aggressive learning rate over-adjusts network weights.

One solution to this problem is to apply a momentum term stochastic gradient descent. This calculates an exponentially decaying moving average of previously calculated gradients. On average, the gradients should be moving towards the global minimum. By averaging past gradients, learning size or step size depend on the size and alignment of sequential gradients. As gradients align, the step size increases, allowing training to more rapidly approach the global minimum. If training oscillates around the global minimum, the gradients will fall out of alignment and step size will decrease, allowing training to approach the global minimum more closely. However, momentum functions require an additional hyperparameter, the exponential decay rate, to be added and potentially optimised in training algorithms (Goodfellow, et al., 2016).

An alternative to stochastic gradient descent is adaptive learning rate models. These do not have a fixed learning rate or step size, but rather automatically determine future learning rates based on training performance in previous training steps. The three most common methods are:

- AdaGrad – all model parameters have individual learning rates adjusted to them as an inverse function of the square root sum of historical contribution to past weight gradients; however, this

method often reduces learning rate too rapidly in deep learning scenarios due to the small contribution of individual parameters to the overall gradient (Duchi, et al., 2011)

- RMSProp – improved implementation of AdaGrad with gradient contribution measured as an exponentially weighted moving average, by reducing the impact of earlier training performance; RMSProp avoids rapidly decreasing learning rates which reduce the functionality of AdaGrad in deep learning networks (Hinton, et al., 2012)
- Adam – an adaption of RMSProp, Adam adds first and second-order moment terms to estimate gradients and step sizes to apply; the second-order term accounts for initial weights which may lead to bias during initial training; however, this method is reasonably robust to any selected training hyperparameters and the defaults can often be used (Kingma & Ba, 2014)

Further information on the algorithmic implementation of these methods may be found in the exhaustive work of (Goodfellow, et al., 2016).

5.2.1.3 Recurrent Neural Networks

Recurrent neural networks are strongly geared to sequential analysis tasks such as text and speech recognition (Schmidhuber, 2015). They contain a form of internal memory which is used to compare data across a temporal range and may compute any task as they fulfil the definition of Turing complete machines (Siegelmann & Sontag, 1991). Turing complete machines are mathematically proven to be capable of emulating any other computational system and by extension, perform any computer defined calculation. While this suggests that recurrent neural nets have wider real-world applications, in reality, they have been less flexible.

The memory within recurrent neural networks is formed from forward and reverse movements of information between and within hidden layers as indicated in Figure 5-3. This causes information to remain within the neural network and interact with new information or subsequent data in a time series. Although this structure is elegant, it is an unstable configuration which can easily lead to chaotic behaviour of the output neurons with small changes in weights and biases. This problem is exacerbated when input dimensionality increases. As such, training for recurrent neural networks is generally done with genetic algorithms and specialised backpropagation algorithms (Jaeger, 2013).

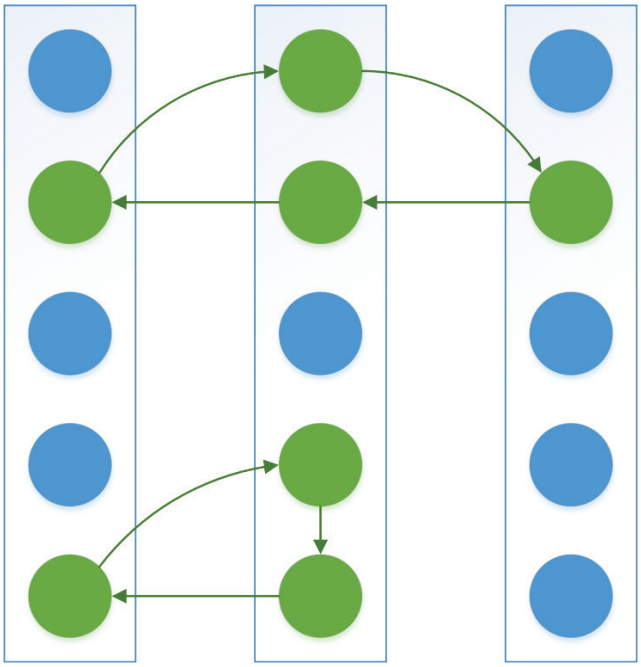


Figure 5-3: Three Hidden Layers of an RNN Showing Two Memory Cells (in Green), Full Connections Between Layers are Ommited for Clarity

Limited literature is available on the use of recurrent neural networks for image processing, though this is focussed on coupling object recognition with object relationships and not the more abstract feature extraction explored in this work. Coupled with its high computational cost it thus will not be used for any further deep learning analyses in this work.

5.2.1.4 Deep Belief Networks

Deep belief networks are a subset of deep learning networks. They are generative models which use many layers of Restricted Boltzmann Machines (RBMs) and are trained in an inherently unsupervised mode (Hinton, et al., 2006).

A standard RBM consists of a number of fully connected hidden and visible units (excepting between units of the same type) which allow calculation to occur in either direction. Feeding data into the visible units generates a representation output at the hidden units. Unlike normal feedforward networks which only calculate in the forward direction, representation data can be fed back into the hidden units and inputs reconstructed at the visible units. RMBs are described as generative models due to this reconstructive ability. Training is achieved by adjusting the weights and biases until the reconstructed image at the visible units has a suitably low error when compared to the original image.

Some of the same concerns regarding overfitting apply to deep belief networks as with other deep architectures. However, training is greatly simplified as the network can be trained one layer at a time. Once a layer has been trained, its hidden output is taken as the visible output of a new layer of RBMs and training

is repeated. Even with highly dimensional data the computational complexity of this method is manageable (Hinton, et al., 2006).

As mentioned in the training section, the unsupervised nature of RBMs allow them to initialise standard deep learning networks with their weights in a form of transfer learning. This leads to greater success in deep neural network training when labelled data is limited (Bengio, 2009).

While deep belief networks are less computationally intensive to train compared to standard feedforward networks, their image processing performance is worse than that of convolutional networks (Schmidhuber, 2015). As such they will not be considered further in this study in the context of vision-based soft-sensing.

5.2.2 Comparing General ANNs and Specialised Architectures

Even greater than shallow neural networks, one of the strengths of deep networks is the ability to model complex and subtle relationships between data without requiring prior knowledge of these relationships and how they are structured. As the number of neurons and hidden layers increase, so does the ability of the deep network to model larger datasets and more complex statistical relationships.

However, this brings an even larger risk of overfitting than with shallow neural networks. As a result, very large datasets and advanced methods of overfitting prevention must be used to allow successful training of deep neural networks (Srivastava, et al., 2014).

The architectures of deep neural networks, especially convolutional neural networks, are highly parallel, performing the same operations on multiple pieces of data. The availability of computational resources such as general-purpose graphics processing units (GPGPUs) make training these and other deep neural networks orders of magnitude faster than possible a decade ago. While shallow neural networks also benefit from these computational advances, their training speeds do not scale as well as those of deep neural networks.

5.2.3 Feature Extraction with Deep Networks

Deep artificial networks are unique as a method of texture extraction as they do not necessarily make structural assumptions of the data as with traditional texture extraction methods. During their training the network may intuitively detect features which are consistent with those found by traditional methods, but they are not constrained by assuming that those are the relevant features beforehand. This has the potential to greatly simplify implementing vision-based soft-sensors in a number of process applications.

As previously mentioned, the inherent characteristics of deep learning allows its use in all three stages of processing and modelling in a soft-sensor design. For this study deep learning will only be applied as a feature extraction (dimensionality reduction) method in the soft-sensor system by extracting hidden, intermediate layer outputs as a feature set. This exploits the data compression and processing characteristics which are inherent in deep learning systems.

While deep learning systems may excel as dimensionality reducers, this method of usage is not standard. The most reasonable application of deep learning systems for vision-based soft-sensors is for the deep neural network to form all parts of the soft-sensor, from input pre-processing to the final input/output model. The network can inherently be trained that way and often optimal performance will be achieved following this procedure.

However, to eliminate bias when comparing texture feature extraction methods, the soft-sensor framework must be as similar as possible between feature extractors, otherwise superior classification performance of the final deep neural network layer may falsely indicate that deep learning features are superior to traditional feature extraction methods. Thus, features are extracted from deeper, hidden layers.

5.3 Convolutional Neural Networks

Convolutional neural networks are deep learning networks with a specialised input structure. Unlike standard feedforward networks (in which neurons are fully connected between layers), the input of a convolutional network is segmented into multiple input areas which share common parameters and are not fully connected with intermediate layers as shown in Figure 5-4 (Fukushima, 1980). This structure is intended to specialise in image analysis as it emulates the neural structure of biological eyes and visual processing centres (Hubel & Wiesel, 1968).

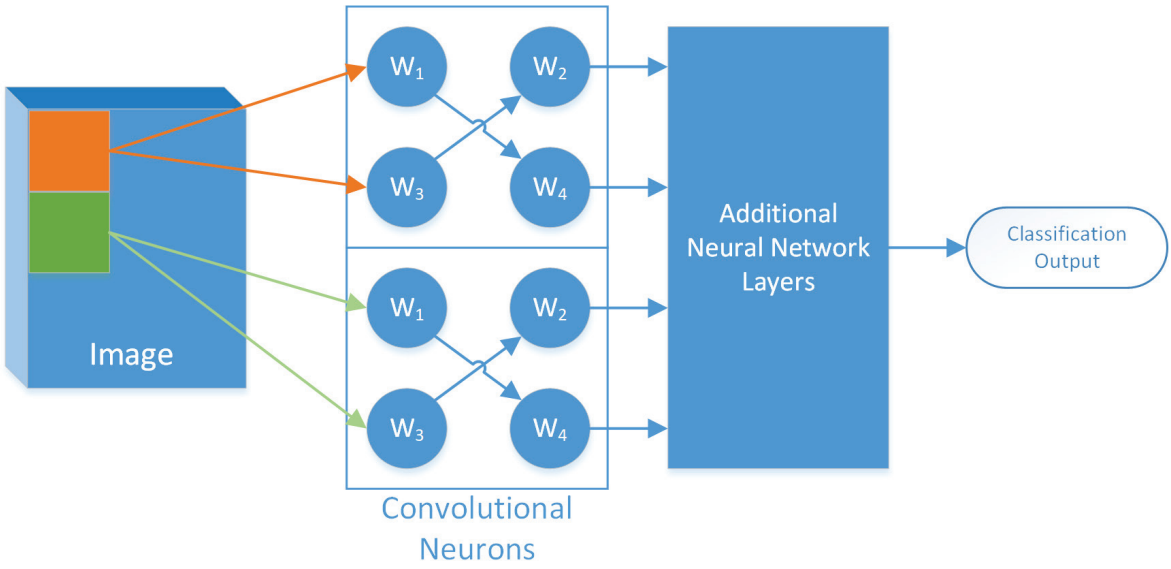


Figure 5-4: Simplified CNN Network Diagram Showing Shared Convolutional Weights (W_1 through W_4) Across Image Area Without Fully Connected Layers

It is believed that biological vision systems act by detecting locally significant features in images and applying the same detection criteria across the entire image. Features which are locally significant should form globally significant features and the spatial relations between local features are preserved deeper in the convolutional network. The exact position of the features is not as important as knowledge of their presence and information about spatial relationships with other detected features.

In typical convolutional neural networks, the first stage of a convolutional layer is the convolution function. Filters are applied stepwise across the entire image and are convolved with local pixels, which has several computational advantages. The input weights (filters in this case) are identical across the entire field of the input image, strongly mitigating the curse of dimensionality. The convolutional function is summarised in Equation 5-6 for a multichannel input field being convolved by a single filter.

$$O_{i,j} = \sum_{i=0}^{X} \sum_{j=0}^{Y} \sum_{c=0}^{C} C_{i,j,c} \times I_{i,j,c} \qquad (5-6)$$

In this equation, the convolutional implementation proceeds stepwise with each pixel (and its colour channels) in the image. A neighbourhood is selected around the pixel equal to the size of the convolutional filter and dot product is performed with the filter to produce a single output per pixel. This process is repeated for each pixel until a complete output is calculated. Additional convolutions are performed for however many convolutional filters are in use, resulting in one channel per filter in the convolutional output.

However, special actions must be taken at the edges of the image to allow convolution to occur. Two strategies most commonly used are: padding, in which case zeros are padded around the image to allow valid convolution of all pixels in the image; and valid convolution, in which convolution starts deeper in the image with the image edges overlapping with the filter edges. In the first case, the output of the convolutional calculation is the same size as the input image. In the second case, the output is slightly smaller by approximately the size of the convolutional filter.

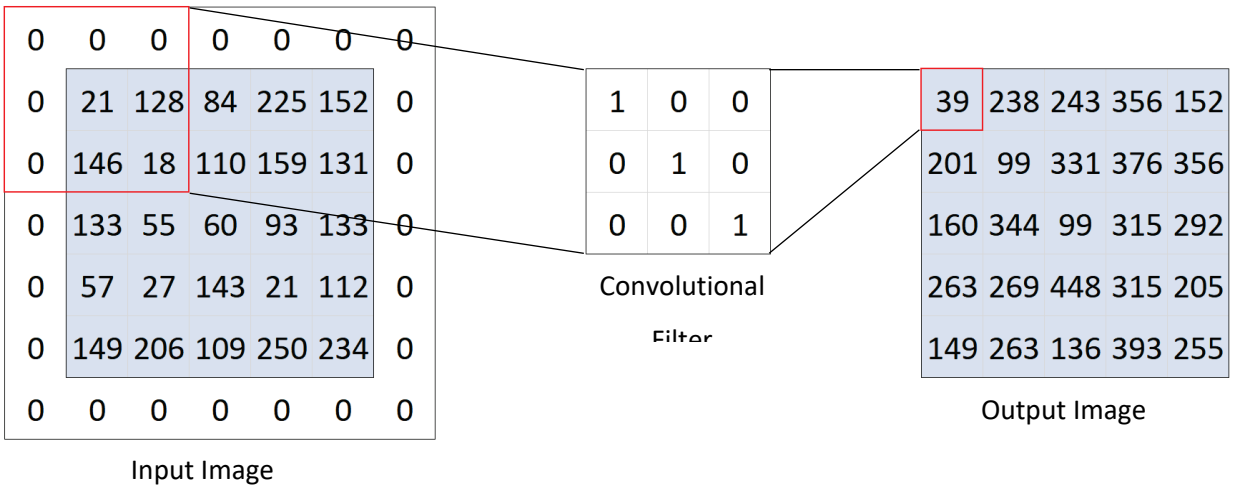


Figure 5-5: Demonstration of Convolution with Padded Inputs

The first method, padding, is used in most applications, and is demonstrated with a 5x5 input image and a 3x3 convolutional filter in Figure 5-5. Following this convolutional step, an activation function is applied to each pixel. Hyperbolic tangent activations can be used, however, ReLUs are more popular for use in convolutional networks and result in improved performance with image processing (He, et al., 2015). As all the output values are positive in the example in Figure 5-5, ReLU activation has no effect on the outputs.

The ReLU function as detailed in Table 5-2 is used for both its computational simplicity and strong non-linear response. There is lower risk of neuron saturation and resulting gradient collapse during training (Krizhevsky, et al., 2012).

The final stage of the convolutional layer is a pooling method. Pooling methods combine a number of pixels in a field and output a single value, strongly reducing the dimensionality of the input data. Max pooling is most commonly used in convolutional neural networks and improves non-linear response by discarding all input values but the largest one in the field under evaluation. This is demonstrated in Figure 5-6 with 2x2 pooling. As with convolution, padding may be applied to ensure that pooling can occur successfully. Empirical evidence indicates that the pooling method is an important step in ensuring satisfactory performance in convolutional neural networks (Boureau, et al., 2010), though debate continues (Goodfellow, et al., 2016).

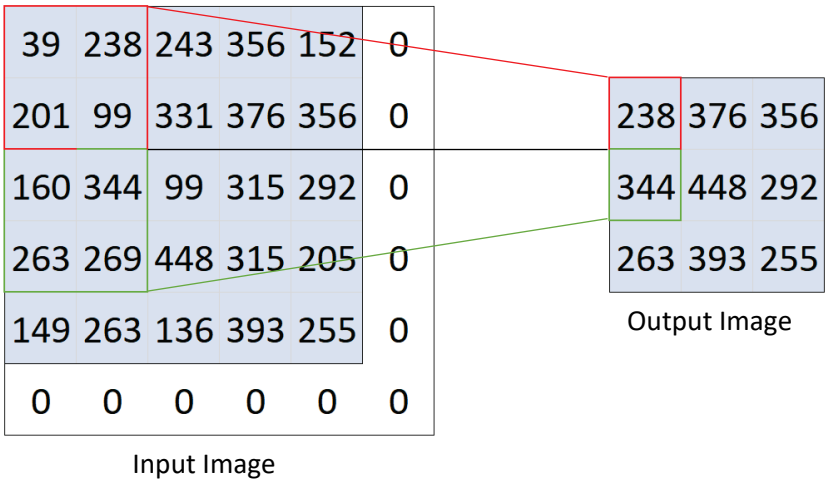


Figure 5-6: Demonstration of 2x2 Max Pooling with Padded Inputs

These three methods together form a complete convolutional layer. These layers may be repeated until sufficient dimensionality reduction or feature detection has been achieved. Final classification or regression is achieved by feeding the 2D output of the final convolutional layer into a fully-connected network. As there is no spatial awareness in fully connected networks, the 2D output is unrolled row by row into a single long vector. This may be considered a large feature set which is fed into the input layer of the fully-connected network.

5.4 Artificial Neural Networks in Process Engineering

Vision-based soft-sensing has begun to be more commonly accepted in the toolbox of process engineers, but implementations typically make use of traditional methods of texture extraction. The new and emerging field of deep learning has yet to make a large impact on this industry, with very little literature available after the year 2000 (Bharati & MacGregor, 1998), (Bharati, et al., 2004). One of the most recent publications found citing the use of neural networks was for controlling a steel pickling process as detailed by (Kittisupakorn, et al., 2009).

Some research has been done on applying multilayer neural networks by (Marais & Aldrich, 2011) and (Al-Thyabat, 2008) in regression and classification sensors. The majority of work with image analysis and neural networks has focussed on small networks and fault detection (Özdemir, et al., 1998). Some of these applications are:

- A shallow neural network classifier of froth image features (Moolman, et al., 1995)
- A model predictive neural network controller for steel pickling (Kittisupakorn, et al., 2009)
- A shallow neural network classifier of grade from froth image features (Wang, et al., 2014)
- A shallow neural network classifier inspecting fruit and vegetable quality (Cubero, et al., 2011)
- Shallow neural networks modelling copper adsorption in industrial leachates (Turan, et al., 2011)
- Shallow neural network models for solubility of supercritical CO_2 in ionic liquids (Eslamimanesh, et al., 2011)
- Using shallow neural networks for general process control and fault detection (Himmelblau, 2000)

Again, there is a distinct lack of recent literature available on attempted implementations in the process industry.

Deep learning methods may not have been applied to industry because process data has not been available in sufficient quantity until recently. Advances in computing power and storage mean that process industries are now generating large quantities of data and storing it for future analysis. However, significant work is still required to process this data and produce meaningful labels for deep learning systems. Process data must be aligned with potential labels and time-based process dependencies must be taken into consideration when analysing large datasets with time series.

Another reason for the limited impact of deep learning methods is a difference in tolerance for risk between research groups. The field of deep learning is new and rapidly evolving, often eschewing the traditional peer review system in favour of self-publishing. Proposed methods are proven by successful applications and implementations. Consequences of failure are typically limited to losses of time or prestige. In contrast, process engineering (and process industries in general) is more conservative, and novel methods will not be implemented without significant testing, as the consequences of failure are more significant.

The potential of vision-based soft-sensing with deep learning systems shows potential when compared to traditional methods, but a lack of published results is holding back implementation of such systems.

5.5 Continuing Advances

The field of deep learning is currently experiencing rapid growth and development. Not only is computational capability ever increasing, but so is understanding of neural network functionality. Due to the rapid pace of development, this work should be seen as a product of research which was widely recognised and implemented in 2016. At the conclusion of this research the accepted cutting-edge methods have already

been supplanted and considerably different neural network architectures have been proposed for image processing tasks such as fully-convolutional networks, residual networks and long short-term memory networks. In addition, the software tools have matured considerably in the last two years, making these new architectures more accessible to researchers in all fields.

Chapter 6: Key Literature

A summary of literature key to understanding the bulk of the work done is provided in this chapter. While the preceding chapters provide the full background and motivation for the work performed, this summary indicates key concepts which were applied and enables a basic understanding of the methodology.

6.1 Froth Flotation Sensing

The enabling factor in this work is that the appearance of certain mineral processes may be linked to process conditions. In the case of the platinum case study, froth appearance has been shown to correlate well with process conditions such as grade (Moolman, et al., 1995). In order to exploit the appearance of processes for control and monitoring, an inferential soft-sensor must be developed, and this has been done previously with froth flotation systems (Marais & Aldrich, 2011).

One key performance indicator important to industrial operators is grade within the flotation cells. As this has been measured before with some success, it presents an ideal baseline for comparison between different types of soft-sensors (Kistner, et al., 2013).

6.2 Image-based Soft-Sensors with Convolutional Neural Networks

Any soft-sensor system developed for sensing process data from images requires some form of feature extraction to mitigate difficulties associated with high dimensional data such as images. While previous work has achieved some success with traditional feature extraction methods, they are difficult to implement without carefully controlled imaging conditions with extensive hyperparameter selection and final tuning. Traditional feature extraction methods are generally limited to specific applications, with no single method being best for all types of process images (Bharati, et al., 2004).

Deep learning, and specifically convolutional neural networks, may be an alternative to traditional methods of texture feature extraction. They can be resilient to changes in imaging conditions and respond well to unseen data or data outside normal operating conditions (LeCun, et al., 2015). Additionally, they do not make assumptions of image feature types and deep networks themselves can replace the traditional soft-sensor framework, forming all the necessary parts for classification or regression from input images.

The deep network architecture selected for this work is a convolutional neural network. This network type has been specifically developed for image processing purposes (Fukushima, 1980). It sacrifices some generality compared to fully-connected neural networks but is much less susceptible to the curse of dimensionality.

However, in order to accurately evaluate the ability of deep neural networks, they must be evaluated only on their ability to extract features, not correlate features with classes or other output data. Thus, the neural network must be trained and then restricted to output information from hidden layers as features for a

traditional soft-sensor framework. This is similar to methods that use unsupervised training such as autoencoders which then change the output layers of trained networks for data compression purposes (Baldi, 2012).

The lack of successful unsupervised learning systems with deep learning leads to the selection of supervised learning for the convolutional feature extractor. To combat the risk of overfitting and poor training performance of limited datasets, several techniques are employed. Dropout improves sparsity within neural networks and inhibits overfitting (Srivastava, et al., 2014). Variant generation increases network invariance to rotation and scaling of inputs and increases representation in datasets of changes in imaging or process conditions, further reducing the risk of overfitting (Dosovitskiy, et al., 2014). Finally, batch normalisation reduces neuron saturation in complex networks and improves gradient descent performance (Goodfellow, et al., 2016).

Chapter 7: Methodology

There are three primary areas of work in this project which required specific methods and software and are intimately related to the objectives of the work. Firstly, features must be extracted, the bulk of the work in this part will be concerned with developing the convolutional neural network feature extractor and training it (fulfilling objective 1 – developing and evaluating a deep learning method). This is presented as a separate section from the other feature extractors. Secondly, the performance of extracted features from each method are compared in a soft-sensor for grade classification (fulfilling objective 2 – comparing performance of feature extractors). Lastly, the third objective is achieved by analysing the results of the final soft-sensor within an industrial context

7.1 Dataset Information

The primary dataset in this work came from a case study conducted at a flotation process using platinum group metals. The data was originally gathered and classified in the work of Marais as detailed in their thesis (Marais, 2010).

The original data consisted of several hours of video footage from a primary recleaner cell, during which time platinum grade in the flotation cell was altered by manipulating air flow rate to the cell with mass-pull controllers disabled. Once the froth had stabilised to the new process conditions (as determined by visual inspection), samples of the froth were taken for platinum grade assays. Correlation studies confirmed platinum grade did not correlate with changing airflow rate when using linear methods, though the study was not expanded to non-linear correlations.

The footage was correlated with four distinct platinum grade classes with each subsequent class representing a lower relative grade as summarised in Table 7-1. Some efforts were made to keep lighting conditions constant during imaging, however, some changes in lighting did occur. As such caution is recommended when using spectral methods during image analysis as they inherently use colour properties which are changed by changing lighting conditions

Table 7-1: Platinum Grade Classes

Class	Relative Platinum Grade
1	1.00
2	0.59
3	0.38 to 0.40
4	0.11 to 0.16

In the work of Kistner the dataset was simplified by subsampling at a specified time interval in four time-regions for each class to produce a dataset as summarised in Table 7-2, with a total of 2720 images and their associated classes (Kistner, 2013). The subsampled dataset was chosen to minimise correlation between

subsequent images, allowing a more diverse representation of each class as further described in Section 7.2.1.

Table 7-2: Kistner Dataset Class Image Numbers

Class	Number of Images
1	780
2	480
3	678
4	782
Total	2720

7.2 Pre-Processing of Datasets

Once a set of images with associated process data (classes or values) has been identified it is imported into MATLAB R2016b for processing and storage as a mat file. In the pre-processing step, several tasks are performed:

- Sub-sample data to increase representation range (if required)
- Enhance dataset size through variant generation
- Identify and separate testing data from training data
- Generate complete datasets at different resolution scales
- Maintain coherency between scales, i.e. image 247 is the same image in each resolution scale dataset
- Save images and associated data in a single file for each resolution

All the above steps are performed in MATLAB as several feature extractors have already been developed in MATLAB for prior work. The flowsheet in Figure 7-1 summarises this process. For each resolution scale two datasets will be produced; a large training dataset which consist of both base images (original and unaltered), and variant images (generated from base images). A small testing dataset is also produced which is a random 10% sampling of the original base images. Variant images are not generated for the test dataset as they serve no purpose in evaluating final performance. There are no common images between the two datasets. This independence allows the testing dataset to be used for performance evaluation without biasing performance due to previously seen images.

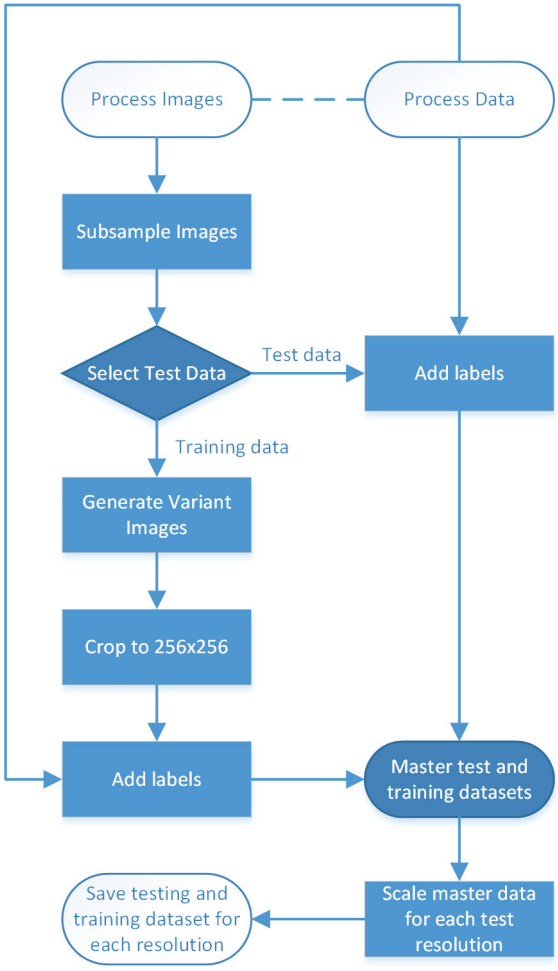


Figure 7-1: Imageset Generation Flowchart

In the training stage of the various soft-sensor components, a random sampling of the training set will be used to validate the training (this will be referred to as the validation set). This random sample will be constant within each set of tested parameters to ensure no validation is done on previously seen data. The settings used to generate the variants are summarised in Table 7-3. Where several variants and variant range increments are supplied, it may be assumed that the variants are split evenly between negative and positive ranges. In the noise variant, the mean and variance of noise is for an image with pixel values in the range [0, 1], for other pixel value ranges, the ranges are converted before applying noise and then returned to their original pixel value ranges.

Table 7-3: Properties of Variants Generated for Testing Data

Variant Type	Number	Range Increment/Parameter
Rotation	8	-15°
Scaling	4	±0.2
Colour Space	1	HSV
Noise	0 (added to 50% of total images)	Gaussian; $\mu = 0; \sigma^2 = 0.01$
Total	13	

7.2.1 Sub-sampling and Variant Generation

Sub-sampling in this case refers to temporal sub-sampling. In a typical set of images there may be thousands of images taken of a process over a long period. However, these images may be temporally close and do not indicate significant changes visually, or are too closely spaced in time for typical process condition change rates. Ordinarily these redundant images would not constitute any problems, though in the context of deep learning representative datasets are desired or overfitting may occur as described in Section 5.2.1.1. Images which present little or no change within them may exacerbate overfitting in deep neural networks and do not contribute to improving neural connections. Sub-sampling of the flotation dataset improves the quality of the testing dataset by increasing the distance between images, reducing the risk of subsequent images being too similar.

The sub-sampling is performed by selecting every nth image chronologically in the dataset, corresponding to an image every nth second. It is difficult to correlate the sub-sampling regime with froth dynamics as they are dependent on many process factors which results in a wide range of time constants for typical froth properties such as bubble lifespan, size, and velocity (Chidzanira, 2016). The value of n is chosen by visual inspection as demonstrated in Figure 7-2. Here the fourth image from the base (Image K) has sufficient variance, however this is not the case across the entire dataset.

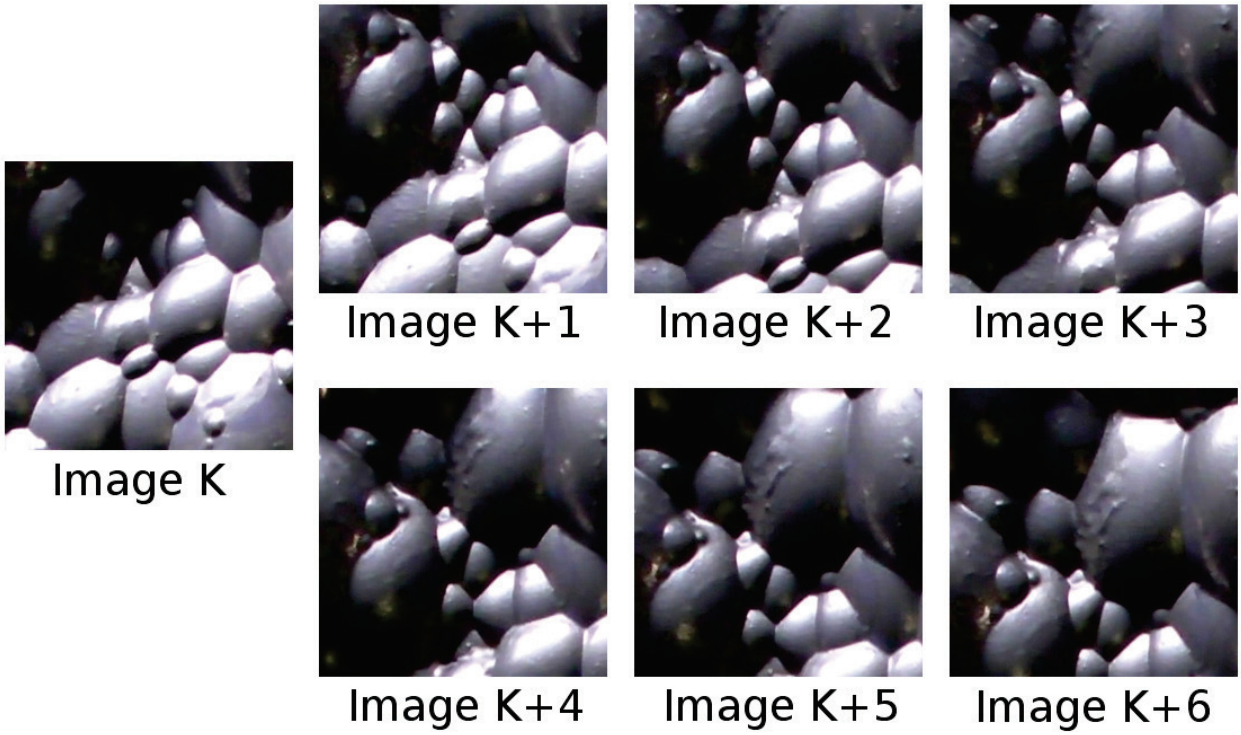


Figure 7-2: Subsequent Images in Dataset

A too large value of n will result in too many viable images being discarded, increasing training difficulty. Too small a value will reduce the representative range of the data and lead to poor validation and testing datasets. For the primary dataset in this work every fifth image was taken from the original dataset. The resultant number of images in each class are summarised in Table 7-4 for the primary dataset used in this work.

Table 7-4: Number of Images per Class after Sub-sampling

Class	Number of Images
1	156
2	96
3	136
4	156
Total	544

Within the context of minerals processing, the sampling interval has an upper limit constrained by process dynamics. In froth flotation, significant process changes generally occur within the timescale of minutes, requiring a sampling interval which should be shorter than that. The original dataset consisted of one image captured every second, by sampling every fifth image, a five second internal of information is created. This is well within the expected dynamics of the froth flotation process and should capture any process variances.

Once the sub-sampled images have been selected their order is randomised and the testing set is selected at random from them. No variants will be generated for the testing dataset as the goal of the variants is to enhance training; they serve no purpose in testing.

The remaining images are the base images of the training dataset. Each base image will have several variants generated from it with the same classification label as the base image. Once a transformation has been applied to the base image, the central 256x256 pixel area of the image is cropped for saving. The size of this initial crop is constrained by the following parameters:

- Must be equal or larger than the largest final resolution set
- Must be equal or smaller than the smallest scaled variant resolution
- Must only contain valid image data for each rotation variant (In the worst case this is less than the minimum dimension of the image divided by $\sqrt{2}$ as shown in Figure 7-3)

Each transform may alter the resolution and orientation of the base image. Selecting the entire image for saving would result in differing resolutions for each variant. Additionally, certain variants may include null data to ensure that the final image remains rectangular. Cropping the central region of the variants ensures consistent sizing and precludes null data from entering the images.

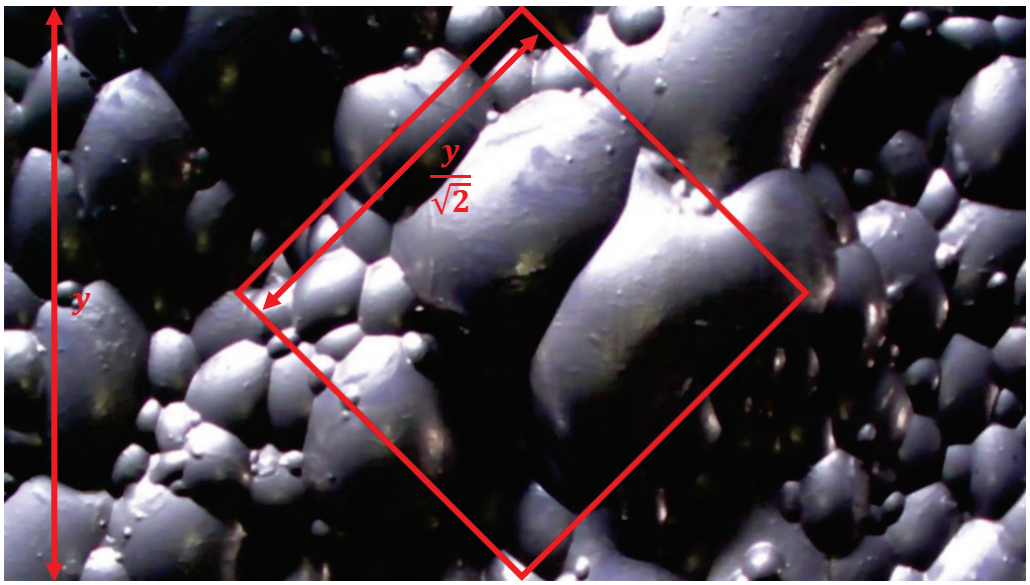
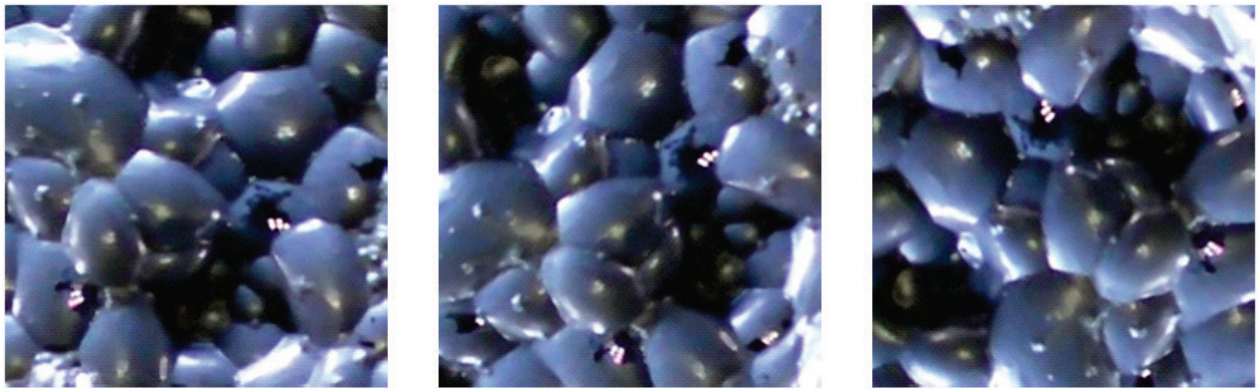


Figure 7-3: Central Crop Size Limitation with Maximum Rotation

7.2.1.1 Rotation Variants

From the base image, images are generated with a linear increase in anticlockwise rotation between each image. An example of rotated images from a base image is shown in Figure 7-4.



Base Image -45° Rotation -90° Rotation

Figure 7-4: Rotation Variants

The number of images and rotation angle is predetermined for a dataset, and should encompass a wide range of rotations to make the system invariant to rotational changes in the imagery. Eight rotational variants were created with a rotation step of 15 degrees, giving a rotation range from -0 to 120 degrees.

7.2.1.2 Scale Variants

As with rotation variants, scale variants are generated from the base images with a linear increase and decrease in scale in successive images as shown in Figure 7-5.

Scale size and number of scaled images is predetermined for a dataset. The scaling ensures that spatial relationships are generalised into ratios rather than absolute measurements. Four scaled variants were created with a scaling step of 0.2, giving a range of scales from 0.6 to 1.4.

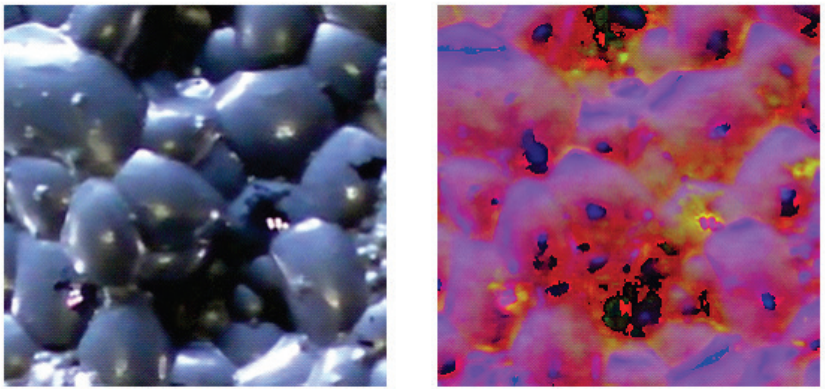
0.6 Scale Base Image 1.4 Scale

Figure 7-5: Scale Variants

7.2.1.3 Colour Space Variants

There are a number of different possible colour spaces in which an image can be transformed. The overall appearance of the image is not altered but rather the colour channels are changed as shown in Figure 7-6.

The actual colour representations are still preserved, but when the new channels are mapped to the traditional RGB colour channels the image appears drastically altered and artificial. The new channels when mapped correctly appear identical to normal RGB images.



Base Image HSV Transform (RGB Channel Map)

Figure 7-6: Colour Space Variant

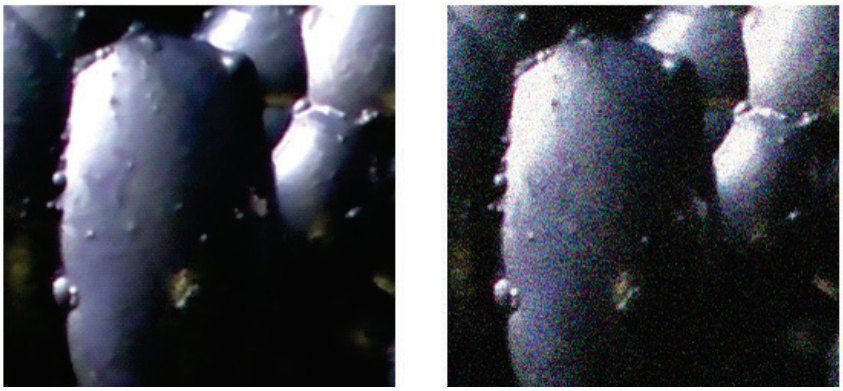
Instead of each channel representing a distinct colour intensity, the colour space is changed to make colour more perceptually relevant such as in biological vision. The redefined colour channels may represent hue, saturation, value, luminosity, and so forth. In this work HSV colour space is used due to the speed of conversion from standard RGB colour space. In the HSV space, the three channels are:

- Hue – perceived base colour
- Saturation – intensity of the perceived colour
- Value – luminous intensity of the colour

Using an alternative colour space allows more intuitive spectral differentiation between images during training and provides a source of textural information independent of spectral information in the "Value" channel. This textural information may assist the neural network in learning textural features which would otherwise appear only weakly in tradition RGB colour channels. Only one colour space variant is generated from the base image.

7.2.1.4 Gaussian Noise Variants

Gaussian noise, as demonstrated in Figure 7-7, is added to a random subset of images from one base image. The noise intensity and fraction of variant images to add noise to is predetermined. Addition of noisy images makes the feature extractor more robust to common imaging artefacts resulting from environmental conditions or imaging equipment malfunction.



Variant Image Gaussian Noise Addition

Figure 7-7: Gaussian Noise Variants

Gaussian noise with a mean of 0 and variance of 0.01 is applied uniformly to the selected images for the primary dataset of this work. As previously mentioned, this mean and variance are assuming the pixel value ranges are [0, 1]. The images in the dataset have an integer value range from 0 to 255, this is uniformly scaled to [0, 1], noise is applied, and the noisified image is scaled and binned back into the original 0 to 255 value pixel range.

7.2.2 Resolution Scale Set Generation and Saving

Once all the variant images are created they are saved along with their base images and a unique identifier for each base/variant set in the complete dataset. This is done by cropping to 256x256 pixels. From this set all other resolution datasets are generated, ensuring complete parity in image identity between all datasets. The images are scaled to the lower resolutions using bicubic interpolation (Keys, 1981). This maintains local relationships within the images which are important for textural features by smoothly combining pixel neighbours.

The original 256x256 images from the primary dataset were too large to train effectively with the computational resources available. 16x16, 32x32, 64x64, and 128x128 scale datasets were created and used to train feature extractors. If the final scaled sets are larger than the maximum size for a MATLAB mat file, 2GB, the datasets are automatically segmented and saved in separate numerically identified files. The split datasets are later recombined when features are extracted.

7.3 Convolutional Neural Network Definition and Training

Two software tools were used to develop the neural network used in this work. TensorFlow by Google's machine intelligence research organisation (Abadi, et al., 2015) and the TFLearn high level library in Python (Damien, 2016). While it is possible to code neural networks and other machine learning methods directly in TensorFlow, this offers a level of control not required for a proof of concept and requires extensive extra coding. TFLearn simplifies neural network creation within the requirements of this work.

The convolutional neural network consists of two distinct network hierarchies, reflecting architectures commonly used at the time of writing: the convolutional layers which perform feature extraction, and the fully connected layers which takes output from the convolutional layers to perform classification and regression. Details on the operation of these layers may be found in Chapter 5. The number of convolutional layers was chosen to produce sufficient dimensionality reduction for the largest input resolution and the fully connected layer size was selected to reflect feature sizes from traditional methods and the average number of variables output from the final convolutional layer.

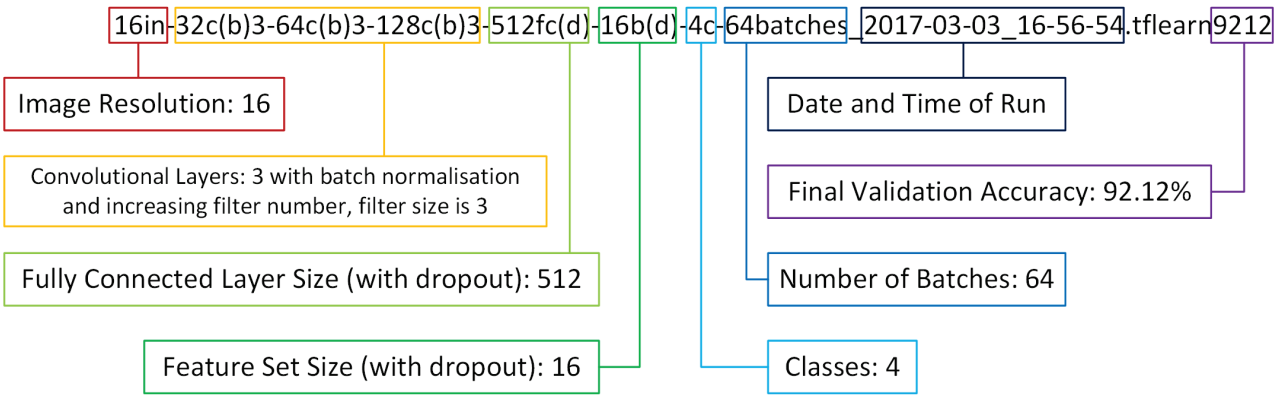


Figure 7-8: Neural Network File Naming Convention

Prior to each training run, the weights, biases, and other variables in the network are randomly initialised to a truncated normal distribution with mean 0.0 and standard deviation 0.02. Any values more than two standard deviations from the mean are discarded and re-picked to mitigate zero gradients from saturating activation functions in the first step of training.

Training extended to 50 epochs on each network to ensure that all networks converged fully and that overfitting prevention measures could be evaluated. Upon completion of an epoch, the neural network will have examined all training images in a training set. Each subsequent epoch re-examines the images to continue convergence to error minima in the network.

A naming scheme was devised to uniquely identify all network variants generated in this work as summarised in Figure 7-8 . The name is constructed from identifying strings indicating the input resolution, number of convolutional and fully connected layers, number of neurons in fully connected layers, filter size in convolutional layers, number of filters in each convolutional layer, whether batch normalisation or dropout is applied to layers, and the number of batches. Additionally, networks are saved with a timestamp identifying when training was started and the validation accuracy after the last step of training.

7.3.1 Convolutional Layers

Each convolutional layer in the neural network performs three distinct operations:

- An image is convolved with a convolutional kernel or filter
- An activation function is applied to each pixel after convolution
- Dimensionality is reduced non-linearly through max-pooling

These operations are defined separately in the neural network, but unless otherwise stated the group will be referred to as the convolutional layer. The number of convolutional layers in the network vary according to the input image resolution; more convolutional layers result in greater dimensionality reduction before the fully connected layers. All networks examined in this work had three convolutional layers. For 256x256 pixel images it was expected that 5 convolutional layers would be required for sufficient dimensionality reduction, though hardware limitations precluded testing of 256x256 images.

Most operations of image data in convolutional neural networks operate over a specified stride. Stride is the distance in pixels which a window of operation moves to cover the entire image. A stride of one is most commonly used, i.e. windows of operation only shift by one pixel at a time to cover the entire image field.

There are no specific guidelines on selecting the convolutional layer hyperparameters. Values similar to published implementations are suggested as a starting point for initial network design and hyperparameters to ensure convergence. Optimisation of network setup and hyperparameters through searching the local hyperparameter value space is then done to achieve optimal classification performance.

7.3.1.1 Convolution Parameters

The variables of each convolutional filter are learned during training. Stride was kept at one for each tested network. A padding value of "same" was always used to ensure the output image is the same resolution as the input image to simplify filter size selection. The output image is padded with zero value data. The filter size is kept constant for all convolutional layers, requiring that the maximum filter size is less or equal to the output image size of the second-to-last convolutional layer. This can be determined programmatically as per Equation 7-1:

$$F_{cfmax} = \frac{r_{input}}{F_p^{n_{conv}-1}} \qquad (7-1)$$

The input resolution (r_{input}) is divided by the max-pooling size (F_p) to the power of the number of convolutional layers minus one ($n_{conv} - 1$), giving the maximum permissible convolutional filter size (F_{cfmax}).

The filter size was one of the hyperparameters which were varied for neural network optimisation, from one to the maximum permissible size for the network under evaluation. Additionally, the number of convolutional filters was either kept constant across all convolutional layers or increased geometrically by a factor of two in subsequent convolutional layers. The number of convolutional filters in the first convolutional stage was kept constant at 32 for each test run, resulting in either 32 or 128 filters in the final, third convolutional stage.

Once each convolution function is performed, a ReLU activation function is applied to output values to increase non-linear response.

7.3.1.2 Pooling Parameters

Max-pooling is applied to the output of the activation functions from each convolutional layer. A pool size of 2x2 was used for all networks under test, resulting in a fourfold decrease in the number of pixels after each convolutional stage. As three convolutional stages were used in each network, this resulted in a final convolutional output resolution eight times lower than the initial input image resolution for a 64-fold reduction in the total number of pixels.

Pool size is selected to achieve dimensionality reduction goals in the neural network. If very high-resolution input images are used or few convolutional stages exists, a larger pool size would be specified.

7.3.2 Fully Connected Layers

The fully connected layer hyperparameters were kept uniform across all neural networks tested. Prior work by Kistner has shown that classification can be achieved on the dataset with traditional non-linear modelling tools, making it unlikely that a complex neural network is required for classifying extracted features (Kistner, 2013).

As such, the first fully connected layer was specified to have as many neurons as the worst-case number of outputs from the final convolutional layer: 4096 as found with 128x128 input pixels and incremental scaling of filter numbers.

This first fully connected layer is followed by a second smaller layer. 512 neurons were selected for the second layer as an arbitrary intermediate between the maximum 4096 inputs and the four class outputs. Both the first and second layers use hyperbolic tangent activation functions on their outputs.

The final fully connected layer acts as an output for the network during training and has one neuron per class, for a total of four neurons.

Between the final and second layers in the fully connected stage is a third fully connected layer, distinctly identified as the feature layer in this work.

7.3.2.1 Feature Layer

The feature layer during training is a normal fully connected layer with hyperbolic tangent activation functions and 16 neurons. However, to produce an unbiased comparison between the convolutional neural network and other feature extractors, the feature layer is used as an output layer once the network has been trained.

The raw outputs are used as feature sets similar in utility to those produced by the other feature extractors under evaluation. These outputs are only recorded once the network has been trained.

The number of neurons used in the feature layer is a balance between providing a concise feature set while potentially providing more information than may be assumed from the class labels alone. Prior work on other feature extractors indicated that a number of features between 8 and 18 provided sufficient classification performance, 16 was thus selected within this range.

7.3.2.2 Output Layer

The output layer of the network consists of four neurons, one for each class, and uses the softmax activation function across the total output. Softmax was chosen as it provides a mutually exclusive probability for each class in the training data as the classes themselves are mutually exclusive. As such the sum of all outputs from the output layer will always be one.

7.3.3 Normalisation and Dropout

In order to reduce the risk of overfitting in the trained network and improve network convergence, two strategies are employed: batch-normalisation in the convolutional layers, and dropout in the fully connected layers.

Batch normalisation is used in the convolutional layers as it normalises across the entire field of variables, reducing the impact of neuron saturation in the initial training steps. During initial testing, it was found that omitting batch normalisation resulted in arbitrary training results and a failure to consistently converge networks regardless of which hyperparameters were selected.

Dropout in the fully connected layers for all networks was fixed at 0.5, meaning that 50% of neurons in each fully connected layer are randomly left out of backpropagation updates.

7.3.4 Batch Sizes

Two batch sizes were tested for all networks, 64 images per batch, and 128 images per batch. The upper limit for batch size is constrained by computational resources and targeted network accuracy. The larger a batch size the quicker training occurs due to parallelism within modern GPUs (though the batch must remain within GPU memory limits), at the expense of producing mediocre training results. 128 images per batch was chosen as it is the maximum batch size achievable with 128x128 pixel images on most networks specified during training with the computational resources available. This batch size allowed an epoch of training to be completed in 97 computation steps.

The lower limit for batch size is 1, though adjusting gradients on one image at a time often results in slow convergence or convergence to poor local minima during training. Additionally, small batches result in excessive training times. The smaller batch size of 64 was chosen as half of the larger batch size to investigate if a lower batch size has significant impact on training.

Table 7-5: Convolutional Neural Network Hyperparameters

Input resolution (pixels)		16x16		32x32		64x64		128x128	
Convolutional layers		3		3		3		3	
Number of convolutional filters in layer	**1**	32		32		32		32	
	2	32	64	32	64	32	64	32	64
	3	32	128	32	128	32	128	32	128
Convolutional filter size range (pixels)		1x1 to 4x4		1x1 to 8x8		1x1 to 16x16		1x1 to 32x32	
Batch sizes (images)		64 and 128		64 and 128		64 and 128		64 and 128	
First fully connected layer size (neurons)		4096		4096		4096		4096	
Second fully connected layer size (neurons)		512		512		512		512	
Feature layer size (neurons)		16		16		16		16	
Output layer size (neurons)		4		4		4		4	
Fully connected dropout factor (fraction)		0.5		0.5		0.5		0.5	

7.3.5 Training Search Space

To summarise, a full factorial design is used to characterise interaction between all the hyperparameters. Table 7-5 shows the hyperparameters specified during training, including the range of values tested. The final range of hyperparameters result in 240 unique hyperparameter combinations being tested. A single repeat for each combination of hyperparameters was performed.

7.3.6 Loss and Gradient Descent Methods

The loss function used was categorical cross entropy (Goodfellow, et al., 2016), as defined by Equation 7-2:

$$Loss = \frac{1}{N}\sum\left(-\sum_{i} y_i \log(\hat{y}_i)\right) \qquad (7-2)$$

Cross entropy calculates the number of bits of information required to correct for an error between a predicted distribution ($\hat{y}$), and an actual distribution (y). The risk of calculating $\log(\hat{y})$ where $\hat{y} = 0$ is very low as the softmax activation function rarely returns 0 for any class and most algorithmic implementations of cross entropy skip calculating $\log(\hat{y}_i)$ when $y_i = 0$ (as the non-correct classes in the actual distribution do not contribute to the cross-entropy calculation).

The loss at each training step is the average cross entropy for each image in the batch (N would be batch size as specified in Table 7-5). The gradient descent function used in training was Adam as described in Chapter 5 with a learning rate of 0.001, and default decay rates of $\beta_1 = 0.9$, and $\beta_2 = 0.999$ for the first and second-order moments respectively. Adam was selected as training with initial networks indicated that the large number of weights and small gradients in the network presented too great a challenge for naïve stochastic gradient descent. The resilience of Adam to training hyperparameters enables them to be set and left as-is without further experimental optimisation.

7.3.7 Training and Convolutional Feature Extraction

Networks are individually trained for each of the identified hyperparameters and their final weights and biases stored for future usage. After each epoch, classification accuracy of the network is evaluated using a random sample of validation images taken from the training data at the start of training. This and loss are performance indicators for the networks during training. This process is detailed in Figure 7-9 with the general training procedure shown in Figure 7-10.

The best networks, in terms of final validation accuracy, are chosen from each input resolution size to act as feature extractor for the soft-sensor. Additionally, five repeats of training are performed with the identified networks to evaluate variability in network training.

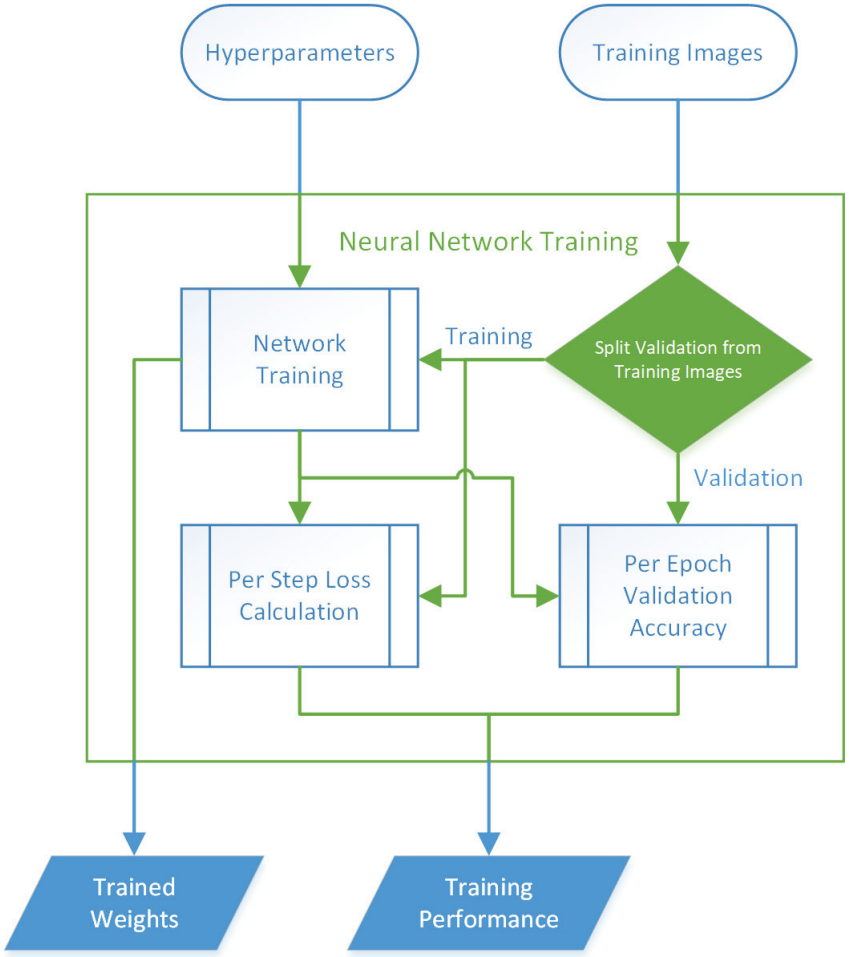


Figure 7-9: Detailed Flowsheet of Neural Network Training Results Calculation in TFLearn

The weights from the selected networks are reinitialised to networks with the feature layer as the output layer. All training and testing images are input, and distinct features sets are generated for each using the newly defined networks. The training features are used to train the final soft-sensor and testing features evaluate the final sensor performance. During feature extraction, time to extract features from a set number of images is recorded for comparison with other feature extractors.

The extracted features are initially stored in separate mat files, which are later combined into final mat files representing all features for each input resolution along with the associated true class labels. The overall training and feature extraction process is summarised in Figure 7-10.

The general flowsheet in this figure represents only one input resolution. The entire process is repeated for each resolution under test with the appropriate associated set of hyperparameters.

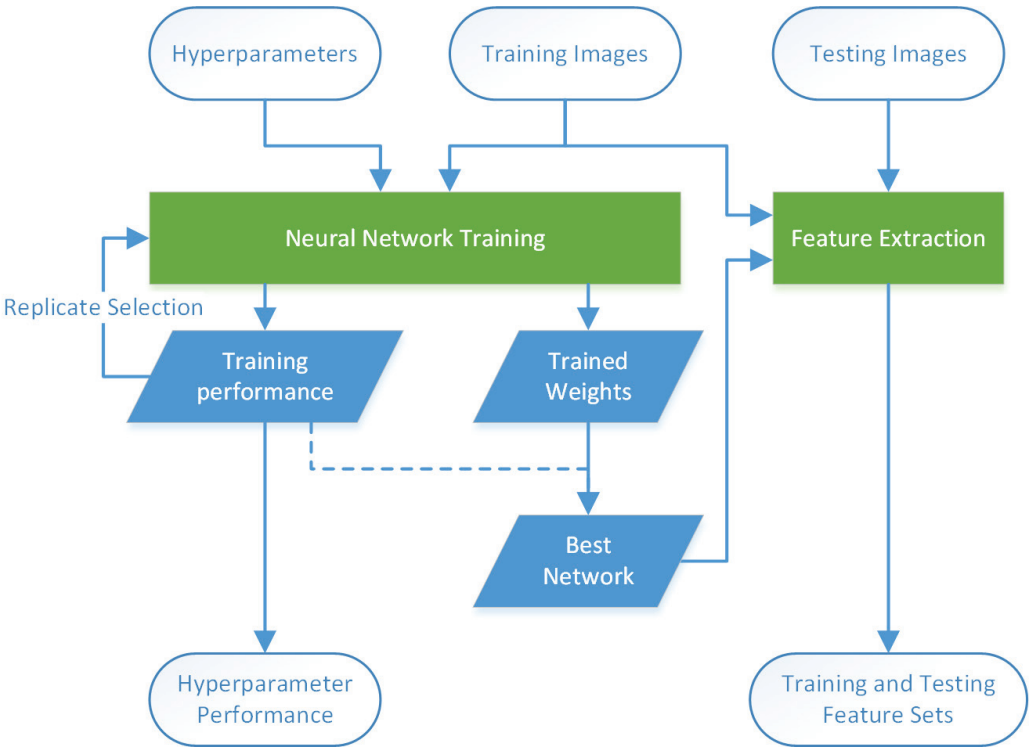


Figure 7-10: General Flowsheet for Training and Feature Extraction in Neural Networks

7.3.7.1 Evaluating Network Training Results

Variations between the different hyperparameters are represented by plots of both the loss and accuracy against convolutional filter size for a specific input resolution. Separate plots are made for each hyperparameter factor combination, resulting in four plots per graph from batch size and convolutional filter increases.

Due to interaction of the gradient descent algorithm, Adam, and training step size adjustment, the loss and accuracy move around final minima in gradient space without ever reaching it. To determine a true loss and accuracy value for each network under testing, multiple values in the final training epochs are taken as a representative population for loss and accuracy. As loss is calculated and stored during each training step, the last 50 values calculated in the final epoch of training is taken as the loss population for a specified network. Validation accuracy is only calculated once per epoch. As such, the accuracy from the final five epochs of training are taken under the assumption that the final minimum has been reached before then.

The first round of training has single replicates, from which loss and accuracy populations are merged. For replicates of the final selected network for each input resolution, the populations are compared to determine repeatability of training with random weight initialisation.

The loss and accuracy are plotted to represent an overall view of hyperparameter effects. However, the large error in these values makes a direct visual comparison inaccurate. Adding error bars does not improve the readability of the figures due to overlap of their ranges. Instead, for each hyperparameter factor tested, a measurement is made of the fraction of tests where a change in a hyperparameter results in statistically

significant change in the loss and accuracy of the neural network. The fraction is calculated for each hyperparameter comparison where the specified hyperparameter is the only one which changes.

7.3.7.2 Statistical Tests Used in Training Evaluation

To initially determined if there are significant differences between means of the performance results, a one-way analysis of variance (ANOVA) using the F-statistic is performed. This has a null-hypothesis (H_0) which states that all means tested are equal within the significance level, the alternative hypothesis (H_1) is that not all the means are equal. While this test may reject the null-hypothesis, it does not provide any information on which of the means, and thus hyperparameter settings, results in statistically significant differences in training results.

A potential solution is to do pairwise testing between all means with a series of t-tests. However, when comparing a large number of means, the probability increases some null-hypotheses may be incorrectly rejected, skewing overall results when comparing significance of means. For example, when comparing 4 means, 6 t-tests would have to be performed. Assuming a significance level of 0.05 and that the null-hypothesis is true in each case, the probability that the tests will all correctly accept the null-hypothesis is $(0.95)^6 = 0.735$. Thus, the probability of incorrectly rejecting the null-hypothesis is 0.265, a significantly larger value than 0.05.

In order to avoid this during testing, alternative methods known as multiple comparison procedures must be followed when comparing many means. Multiple comparison procedures place an upper bound on the probability that any comparison will incorrectly reject the null-hypothesis. The procedure used in this work is Tukey's honest significance test (Tukey, 1949). The difference between the means being measured is divided by the standard error (SE_{ij}) of the combined means to produce a studentized value (q_{ij}) as shown in Equation 7-3.

$$q_{ij} = \frac{|\mu_i - \mu_j|}{SE_{ij}} \qquad (7-3)$$

The calculated studentized value is compared to a critical studentized value which is determined by the number of means, degrees of freedom, and desired group significance level. This ensures that the probability of incorrectly rejecting the null-hypothesis is at most as much as the specified significance level. As initially described in the previous section, results of this multiple comparison test are presented as the fraction of means for each hyperparameter change at specified conditions which have statistically significant differences.

7.3.8 Unsupervised Training

Initial neural network attempts in this work attempted to simulate unsupervised training by assigning unique class labels to each image in the dataset or through using class labels from clustering algorithms. This would

allow large amounts of process images without associated process data or labels to be used in training the neural network. These labels would then be used to train the convolutional neural network, from which features would be extracted as with the supervised method already discussed.

In the unique class label case, base images were assigned unique numerical classes with variant classes sharing the class of the base image (Dosovitskiy, et al., 2014). T-distributed stochastic neighbour embedding was used to form clusters of the images and then the 3D coordinates of each image used as a continuous label (Weston, et al., 2012).

The poor training performance of images labelled in this way led to these methods being abandoned for pure supervised training.

7.4 Neural Network Visualisation

To better understand the behaviour of trained neural networks, two visualisation strategies were employed: naïve feature visualisation (deepdream visualisation), and intermediate layer output visualisation (Mordvintsev, et al., 2015). These are applied primarily to convolutional layers as their weights, outputs, and features are two or three dimensional, allowing for image based visualisation. In comparison, fully connected layers have one dimensional outputs equal in length to the number of neurons in the layer, which are meaningless in image visualisation terms. Though deepdream feature visualisation may be applied to neurons within these layers. Additionally, weight visualisation may be performed on the convolutional layers, but this was not performed in this work as the visual appearance of convolutional weights are unlikely to be visually meaningful (Goodfellow, et al., 2016).

7.4.1 Naïve Feature Visualisation

This method attempts to synthesize an image which strongly activates single neurons selected within a neural network. From the input layer and selected neuron layer, a gradient may be calculated between selected neuron and input image. The method used in this work is naïve as it does not implement more complex filtering or multiscale generation for the final synthesised image.

Iteratively, features within an input image are reinforced until high activation of the target neuron is achieved. Input images may either be real images, in which case existing features are emphasised, or random noise. When random noise is used, features are generated from scratch in local noise variations. For this work, random images are generated with a uniform noise distribution between 0 and 1, then shifted into grey by adding 100 to each colour channel. Images are updated for 100 iterations with an update weight of 1 and normalised between each iteration. Additionally, a small constant is added to each image before normalisation to ensure than no divide by zero errors occur. After iterative updating is complete the image is normalised again and scaled to standard RGB pixel values for display (Mordvintsev, et al., 2015).

This method may be used on any neuron the network, but due to shallow gradients deeper in the network it may have difficulty producing visualisations from neurons in later fully-connected layers.

7.4.2 Intermediate Layer Output Visualisation

Similar to using the feature layer as an output on the trained network, any layer can be selected to produce outputs. Selected images from the test set are input to the network, and the resultant output from each layer can be visualised. This allows for the effect of each filter within a layer to be observed on the input image and how features are amplified or discarded in subsequent layers.

This method is limited to the convolutional layers where two dimensional outputs are available for display.

7.5 Feature Extraction

The traditional feature extraction methods used in this work are:

- Grey-Level Co-occurrence Matrices (GLCM)
- Local Binary Patterns (LBP)
- Wavelets

These features were extracted from the training and testing datasets using the Multivariate Image Analysis (MIA) Toolbox for MATLAB developed in prior work by (Kistner, 2013). Optimised parameters for each of these feature extractors were developed by (Kistner, 2013) using the same original dataset as used in this work, except for additional sub-sampling and resolution scaling not present in their work. The parameters were used to produce features from the sub-sampled dataset for comparison with features extracted from the convolutional neural networks.

It should be noted that only one input resolution, 224x224 pixels, was used in the work of (Kistner, 2013). Some of the parameters for the features used may have different effects depending on the input resolution, though feature performance was still satisfactory across all input resolutions.

The features were all saved to mat files as with the convolutional features described in the previous section. A brief overview of each method and altered parameters are provided in the following sections. Unless otherwise stated, parameters are the defaults as recorded in the work of (Kistner, 2013).

As a final control method, basic spectral features are also extracted and recorded for performance comparison with textural methods used in this work.

7.5.1 GLCM Parameters

A popular texture based extractor, GLCM analyses localised grey levels within a specified area and produces several statistical features describing textures in the image. The parameters used to extract GLCM features are summarised in Table 7-6. Eight features are generated for each image.

Table 7-6: GLCM Extraction Parameters

Number of Grey Levels	128
Distance (pixels)	4
Directions (degrees)	0, 45, 90, 135
Features	Contrast, Correlation, Energy, Homogeneity

7.5.2 LBP Parameters

Similar to GLCMs, LBPs analyse local textures in an image, with each pixel compared to its neighbours. The output is a histogram of LBPs in the image. The parameters specified for LBP extraction in this work are summarised in Table 7-7. 243 features are generated for each image.

Table 7-7: LBP Extraction Parameters

Neighbourhood Radius (pixels)	4
Neighbourhood Sampling Points	16
Mapping Type	Uniform

7.5.3 Wavelet Parameters

A form of image decomposition, wavelets produce directional coefficients and are generally preferred to Fourier analysis in contemporary research. Outputs are the energies of all coefficients. The parameters used to extract wavelets in this work are summarised in Table 7-8. Eighteen features are generated for each image.

Table 7-8: Wavelet Extraction Parameters

Wavelet Type	Daubechies 4
Decomposition Level	16

7.5.4 Spectral Parameters

The average 8-bit intensity of each colour channel over the entire image is calculated, resulting in three features for each image. Unlike textural features, no spatial information is recorded in these features and these features are only a naïve representation of the average red, green, and blue colour intensities in the image.

7.6 Soft-Sensor Training

The soft-sensor is trained in MATLAB using the built-in functions for Linear Discriminant Analysis (LDA), and k-Nearest Neighbour (k-NN) classification models, similar to those used in prior work by (Kistner, 2013). These methods are linear and non-linear classifiers respectively and are relatively simple to train. LDA attempts to form linear combinations of features which separate the specified classes. k-NN attempts to create clusters

which correlate with classes. The distance of a feature from one or more neighbours (as determined by the k value) determines which class it is assigned.

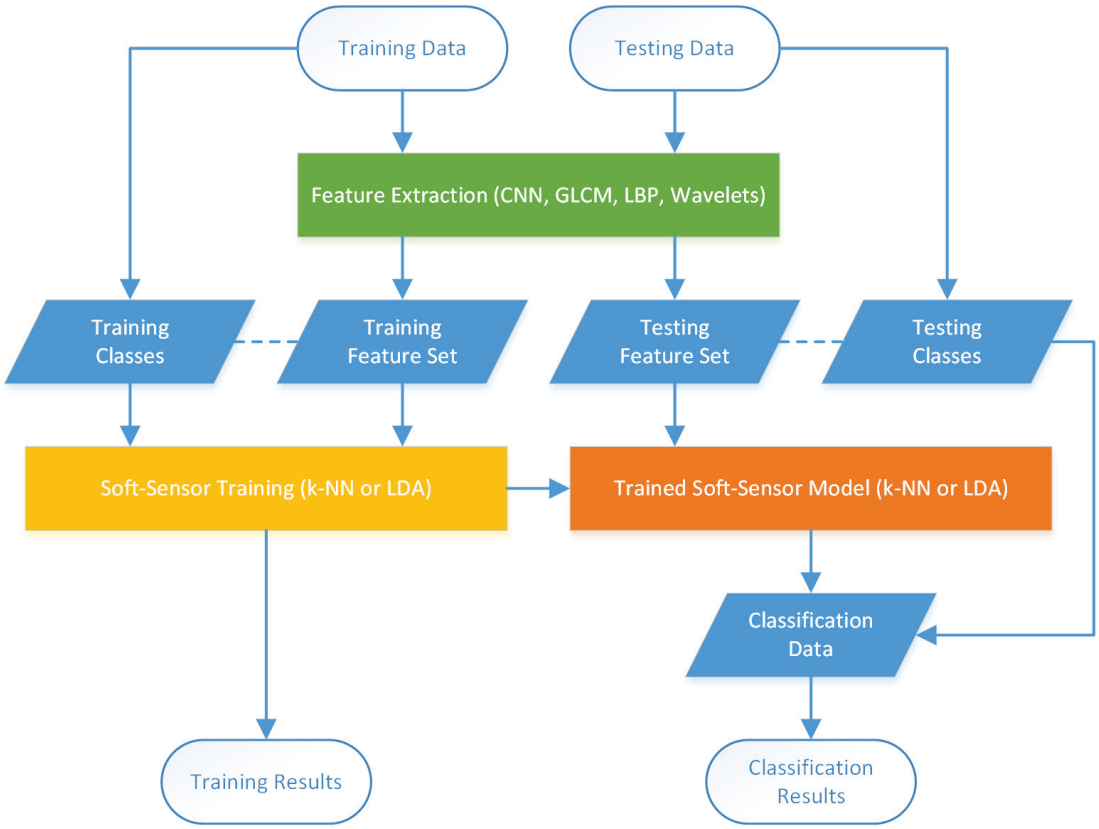


Figure 7-11: Soft-Sensor Training Process Summary

The process of soft-sensor training is summarised in Figure 7-11. For a given feature extractor, features from the training images are used to train the specified soft-sensor model. Once the LDA or k-NN model is trained, features from the test images are used to output a prediction which may be compared to the actual class labels.

Unless otherwise specified, the default parameters for each model was used in the MATLAB implementation. Relevant details are described in the following sections.

7.6.1 Linear Discriminant Analysis Modelling

The fitcdiscr function in MATLAB is used to implement the LDA model. By default, it uses a regularised linear method to fit the data, however regularisation is only used where necessary to invert the covariance matrix of the model. To account for differing numbers of each class in the training data, a weight is applied to each observation which correlates with the prevalence of each class in training data, also known as the prior probability of that class.

In simplified terms, LDA computes the axis which represent maximum separation between the various classes. Scatter matrices are calculated within and between classes which form eigenvectors from which eigenvalues are found. The largest eigenvalues are used to determine which eigenvectors form a transform

matrix. The transform matrix is then used to map the features onto a new subspace. Once this is completed, Euclidean distance from the separating axes may be used to classify the data points (Balakrishnama & Ganapathiraju, 1998).

7.6.2 k-Nearest Neighbour Modelling

Using the fitcknn function in MATLAB, one of two nearest neighbour search methods are used to correlate feature data with class labels. When the feature set consists of less than 10 features it is not sparse and a kd-tree with a maximum leaf node size of 50 is used to find nearest neighbours (MathWorks, 2017). When more than 10 features are in a feature set an exhaustive neighbour search is performed. In both cases Euclidean distance is used to calculate the distance of each point and only a single nearest neighbour is used to classify an observation during model training.Work by (Aggarwal, et al., 2001) suggests that distance measurements other than Euclidean are better suited to high dimensional data. However, prior work by (Kistner, 2013) using Euclidean distances on a similar flotation dataset and using the same feature methods yielded satisfactory performance

Unlike LDA where features are mapped to a new space to maximise separation for linear classification, the Euclidean distance in k-NN is calculated in the feature space between individual observations, not linear boundaries.

7.7 Performance Comparison Between Feature Extractors

The trained final soft-sensors are used to make predictions using the testing data. The resulting predictions are compared to actual class labels and used to calculate an overall weighted loss score. Classification losses are weighted according to the prior probability of each class in the testing data as shown in Equation 7-4. This ensures that the loss score accurately reflects a larger loss for misclassification of a small class observation.

$$L = \frac{1}{n}\sum_{i=1}^{n} w_i I\{\hat{y}_i \neq y_i\} \; where \; I\{\hat{y}_i \neq y_i\} = \begin{cases} 1 \; for \; \hat{y}_i \neq y_i \\ 0 \; for \; \hat{y}_i = y_i \end{cases} \qquad (7-4)$$

In addition to the overall score, the accuracy of each individual class prediction is summarised in confusion matrices as demonstrated in Figure 7-12. Confusion matrices summarise the fraction of images of a certain class which were correctly predicted and the fraction of those images which ended up in incorrect classes. The visualisation allows for overall classification accuracy to quickly be evaluated at a glance while also showing to which classes misclassification occurred for a given actual class.

Figure 7-12 also summarises how a confusion matrix may be read. For example, looking at predictions of class 4, all predicted class 4 images do in fact belong to class four as indicated by the green arrows. The red arrows

indicate an image in class 3 which was incorrectly predicted to be in class 1. In a perfect classifier the matrix would only have a dark red diagonal, indicating all images of each class were predicted correctly.

In this work the higher classes represent decreasing grades of platinum, thus the distance of a misclassification from its true class is also important to note. If two or more models had identical loss scores, it may be considered that the model with the closest misclassification distance is superior.

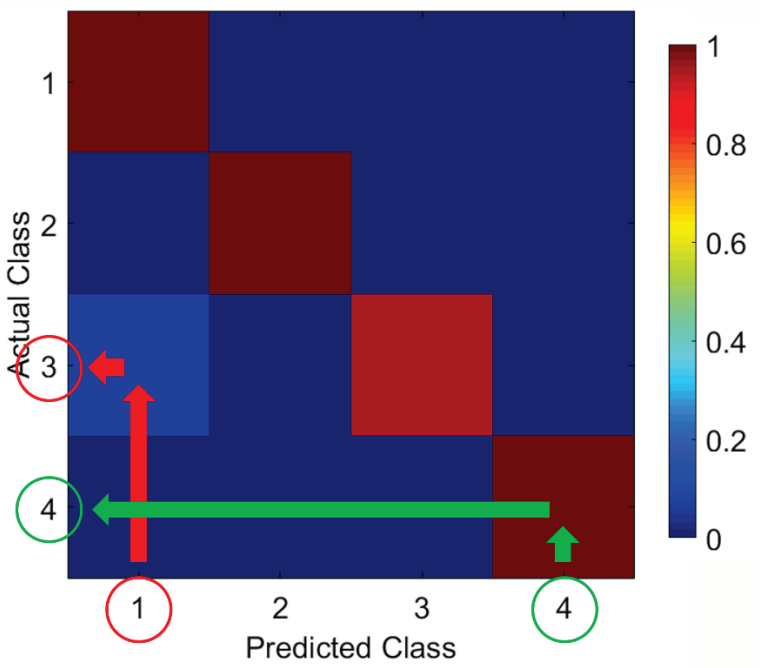


Figure 7-12: Confusion Matrix Example

Performance is considered good when less than 20% of the images are misclassified during testing.

7.8 Hardware and Software Used

The results of this work were generated using a combination of MATLAB and Python. The Deep Learning libraries used were TensorFlow and TFLearn as previously mentioned. It is possible to perform this work purely using open source libraries provided with Python. MATLAB was used due to the existence of prior code and methodologies for generating traditional textural features. All programs were used in a Linux operating system environment, but are also available under Windows and macOS.

All work was performed on an Intel i5-6500 CPU with 16GB of RAM. GPU acceleration was used in both MATLAB and Python for image processing and convolutional neural network training. The GPU used was a nVidia GeForce GTX 780 with 3GB of RAM. A consumer GPU was used for cost savings as there is no performance difference between consumer and professional GPUs for the datatypes (fp32) processed in this work. However, a more modern GPU or professional class GPU with additional RAM would allow for larger image resolutions to be processed.

Chapter 8: Results and Discussion

The results of this work are split into three areas which can be associated with the primary objectives as defined in Chapter 1: neural network training (objective 1 – deep learning development), feature extraction analysis (objective 1 – deep learning evaluation), and soft-sensing results (objectives 1 and 2 – feature extractor performance comparison). Analysing the results of various hyperparameters on neural network training is made difficult by variance in training results and lack of definitive literature on effects of different hyperparameters on feature extraction. The extensive analysis of the neural network training results and correlation with network visualisation are the best methods for evaluating performance in this unique usage scenario. There is a final results section dedicated to discussing implementing soft-sensor for flotation and the disparity between ideal soft-sensors and practically realisable ones.

The primary assumption in the methodology that network training is likely complete after 50 epochs is confirmed in Figure 8-1. Improvement of validation accuracy began to taper off after 40 epochs of training.

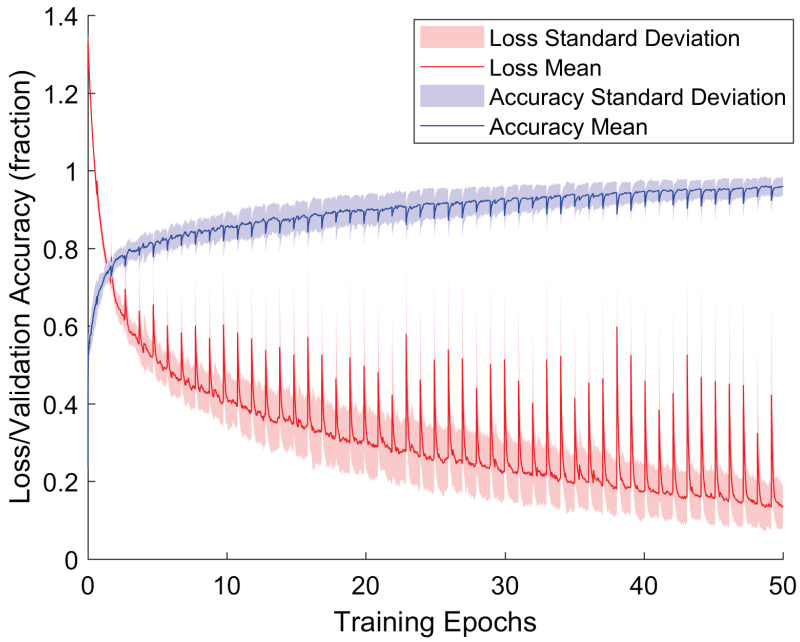


Figure 8-1: Loss and Validation Accuracy for 20 Randomly Selected Networks

8.1 Neural Network Training Results

Overall training results were good with high accuracy and loss for each hyperparameter tested, indicating a robust network architecture. However, the limited variance between different hyperparameter settings may indicate that some overfitting may have occurred, this will be examined more closely in the following discussion.

During initial testing and development of the neural network, there was a lack of consistency in training results. Repeated runs of the same hyperparameters would consistently lead to arbitrary and unrepeatable

results. Very few tested networks exhibited performance suitable for classification better than random chance.

Hyperparameters during initial testing were similar to those used in final testing as detailed in Section 7 (specifically, Table 7-5), with the exception that batch normalisation was not used. Once batch normalisation was applied, training results improved significantly as detailed in the following results. It is likely that the arbitrary initial results were due to chance in random weight initialisation which allowed successful training in some cases.

Utilising batch normalisation as detailed in Section 5.2.1.1 allowed consistently successful training by mitigating the effects of collapsing gradients across the network from neuron saturation.

8.1.1 Batch Normalised Networks Results

As detailed in Section 7, for each input resolution four results graphs are produced: one each for loss and validation accuracy, and an associated hyperparameter variance significance graph for each. Higher accuracy and lower loss is better. The results for all resolutions have been combined into two overall figures: Figure 8-2 for validation accuracy results and Figure 8-3 for loss measurement results.

It should be noted in both figures that the training results for 128x128 pixel images were prematurely curtailed by hardware limitations when testing was performed. Only the results for the least hardware intensive hyperparameter pairing (64 batches with constant number of filters) were valid. The remaining networks tested at this resolution either returned null values as results or failed outright, requiring testing to be restarted.

Overall validation accuracy results in Figure 8-2 are good but do not show significant variance between hyperparameters and have weak decreases in validation accuracy at higher resolutions with larger filter sizes. This is reflected in the low fraction of hyperparameter changes which had a significant impact on accuracy measurements. Only in 16x16 input images does a hyperparameter (the number of convolutional filters) approach significant impact on validation accuracy. This may be due to the network preferring spectral features over textural features at this resolution.

While the resolution scaling attempts to maintain textural relationships at all resolutions, there is an unavoidable loss in textural information at lower resolutions. Spectral features are much simpler (and as later results will show, effective at classifying grade) compared to textural features, making it possible that the neural network was able to more effectively discover underlying structure within the spectral feature space than the textural feature space. A more effective model of the feature space may result in more consistent validation accuracy result during training, lowering variation in training result measurements which results in statistically significant variance between hyperparameters. The direct influence of textural and spectral features is evaluated in Section 8.1.3, showing spectral features present at all resolutions.

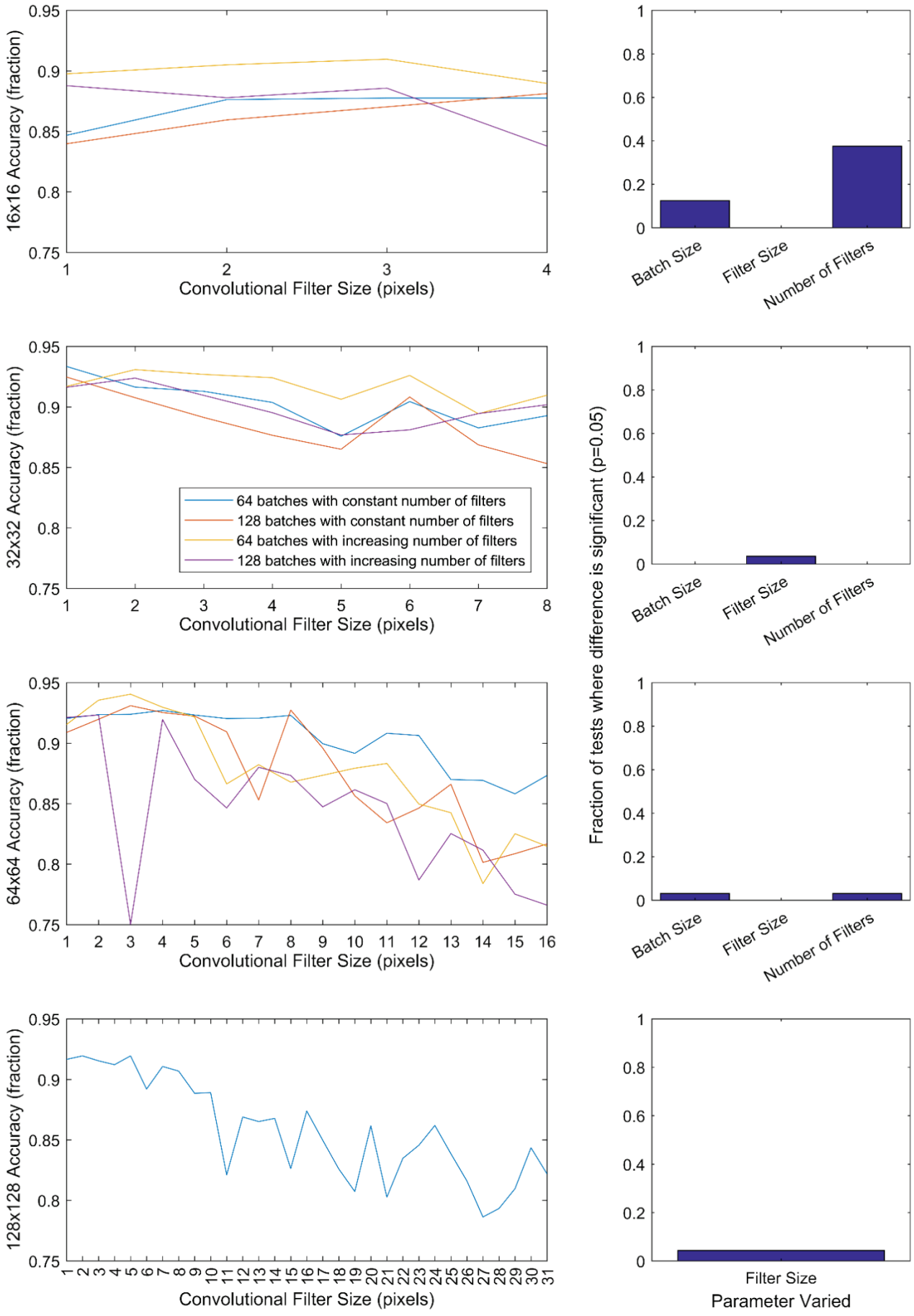


Figure 8-2: Validation Accuracy and Significance of Hyperparameter Variance for All Input Resolutions

Further examining the 16x16 results in Figure 8-2 implies that an increasing number of convolutional filters results in better validation accuracy. However, the limited significance of this hyperparameter selection should still be taken into consideration.

While not supported by statistical significance, it is interesting to note that at 64x64 pixel inputs, the smallest network (64 batches with constant number of filters) loses validation accuracy more slowly with increasing filter size than the other tested hyperparameter pairings. In the context of overfitting, this would be expected behaviour. A smaller network will have a smaller number of weights to train, reducing the potential for overfitting. With the smaller network, the deleterious effects on validation accuracy from increasing filter size (which increases the number of weights) is reduced due to the inherently smaller number of weights.

It would be interesting to see if this behaviour is replicated at 128x128 pixel size inputs. Unfortunately, the failure to train more complex networks at this resolution precludes this analysis. Assuming sufficient representative range has been achieved in the training datasets, it can be assumed that overfitting prevention measures such as batch normalisation and dropout were successful. None of the networks display statistically significant worsening validation accuracy as network complexity increases (which increases risk of overfitting).

The overall low statistical significance of performance comparisons between hyperparameters in Figure 8-2 may be attributed to several factors:

1. Hyperparameter range selection too small
2. Large variance in measurement of validation accuracy for each hyperparameter pairing due to limited repeats
3. Large variance in measurement of validation accuracy due to small validation dataset size
4. Artefact of overfitting prevention mechanisms

It is likely a combination of the above factors, though it is which of these have the largest impact on high variance of accuracy measurements. Additional repeat testing and a larger validation accuracy may mitigate some variance in the results measurement, however the repeat testing in the next section shows that variance of validation accuracy measurements is already relatively low. This excludes factors 2 and 3 from contributing to the lack of variance. Factors 1 and 4 will both require significant additional testing to determine if other hyperparameters produce statistically significant differences in training results.

Moving on to loss measurements in Figure 8-3, broadly similar trends can be seen between loss and validation accuracy. Both perform somewhat worse with increasing convolutional filter size. However, the loss results have considerably more statistical significance across all input resolutions compared to the validation accuracy results. This is likely due to network training directly optimising for loss and significantly less variance inherent in the loss measurement due to the larger sample population in calculating the loss measurement.

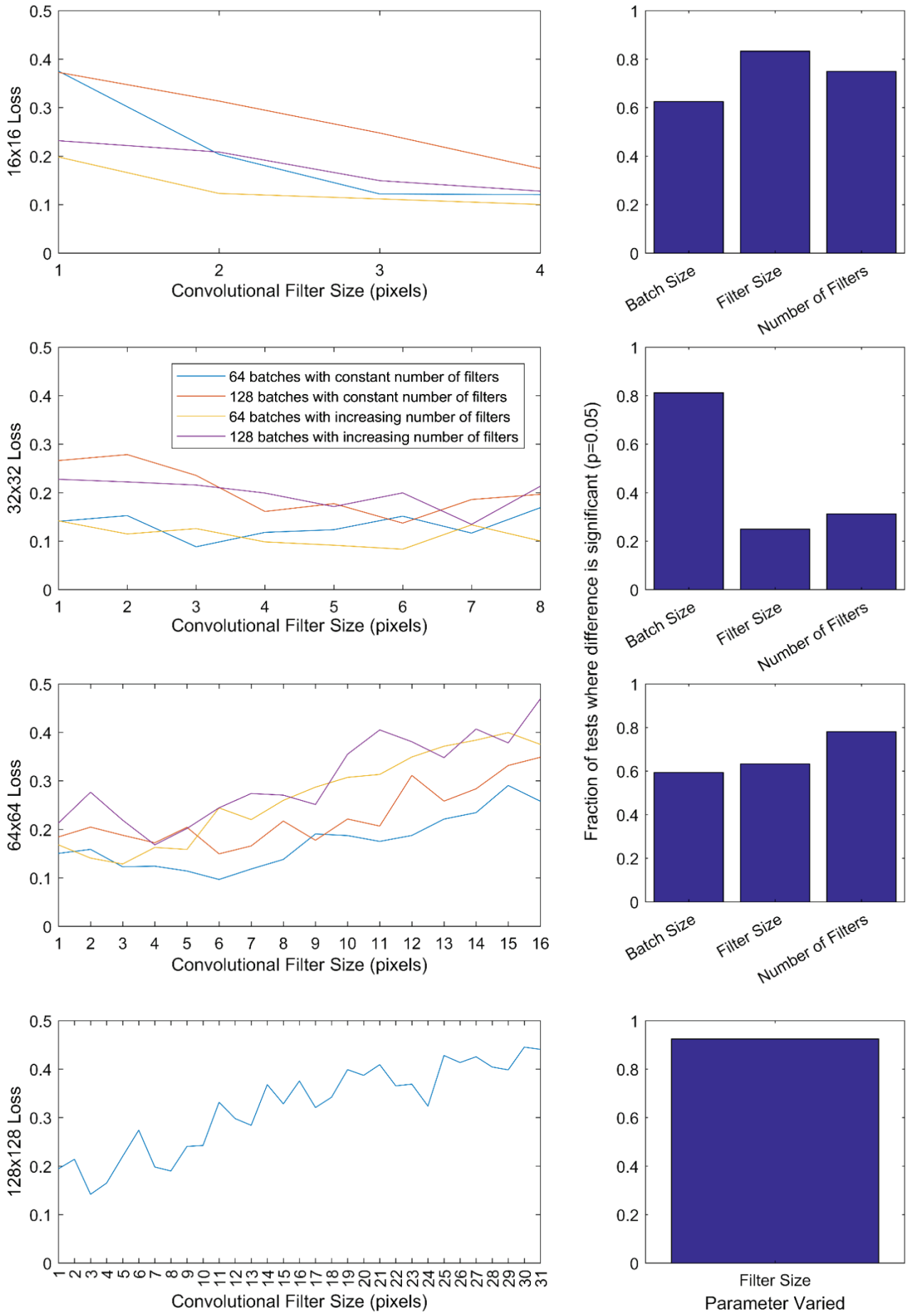


Figure 8-3: Loss and Significance of Hyperparameter Variance for All Input Resolutions

Similar to validation accuracy measurements, smaller networks tend to perform better, with lower loss as indicated in Figure 8-3. Unlike the validation accuracy measurements, many of the pairwise comparisons between hyperparameter testing results for loss have statistically significant results, allowing more meaningful discussion of results. Batch size, and to a lesser degree, number of filters, both have a large statistical significance at all resolutions where they were tested for. A smaller batch size results in better loss results for resolutions from 16x16 to 64x64. Number of convolutional filters has a lesser effect and presents contradictory results at 16x16 and 64x64 pixel resolutions.

At 16x16 an increasing number of filters leads to better loss results, while a constant number of filters improves loss results at 64x64. This may be due to interaction of textural and spectral features at these input resolutions. As previously postulated, it is likely that 16x16 input resolution neural networks will primarily use spectral features during training due to weak textural features. By increasing the complexity of the network at 16x16 pixels, it may allow for more accurate modelling of the spectral feature space. Conversely, at 64x64 pixels it is expected that textural features will be strong. With a more complex network, the high dimensional textural feature space will be more accurately modelled, however, this may be a drawback for classification in this case study.

The features extracted from the textural feature space may be too complex to effectively map to the four grade classes, leading to poorer classification performance overall. The spectral features which are shown to correlate very well to the grade classes in Section 8.1.3 are more easily modelled in the 16x16 neural network, leading to superior classification performance. Essentially a mild form of overfitting is occurring, more complex 16x16 neural networks can more effectively overfit the spectral features, while the 64x64 neural network cannot do the same to the textural features. The effect of this overfitting on validation accuracy is decreased by overfitting mitigation strategies used.

Examining the fraction of significant differences in Figure 8-3 further shows that filter size has strong significance at 16x16 and 128x128 pixel input size. The effect of better loss measurements with increasing filter size at 16x16 is likely for the same reasons that more convolutional filters result in better performance. The more complex the network at 16x16 pixel input, the more capable it is of modelling the spectral feature space. The results at 128x128 pixel inputs should be taken with some more caution. It is possible that the large significance of varying the filter size is abnormally large due to the lack of repeated runs for this resolution. The single test results are likely to not have representative means for loss measurements of the various hyperparameters as indicated by the high variability of means during repeated testing in Figure 8-4.

The final point of interest in the loss measurement results is the optimal point at filter size 6 for 64x64 input networks. This may indicate the presence of the textural property in the image which is similar in size to 6 pixels which correlates well with the grade classes. While tempting to propose this relationship, literature

indicates that convolutional filter sizes often do not conform to textural properties within the images being trained (Goodfellow, et al., 2016).

There is a lack of consistent results across all input resolutions. No single hyperparameter change always had a large change in network performance for all input resolutions and the lack of successful training results at 128x128 input size further limits additional analysis. One parameter which is consistent in providing better performance is a batch size of 64.

Regardless of the lack of significant between the various results, all the results are within a narrow range of good performance. This indicates that the range of hyperparameters and network architecture may have been too conservative. The network architecture may be too complex for the dimensionality of the data and thus any variance from different hyperparameters is offset by compensation from excessive network weights and biases.

It may also be that the network relies too heavily on spectral features rather than textural features. As analysis in Section 8.1.3 shows, there are spectral features being evaluated within the network, and the final soft-sensor results indicate that spectral features are excellent for classification of the current dataset. Unless all spectral information is removed from the images, it is likely that the neural networks will continue to use that in conjunction with textural information and train well regardless of selected hyperparameters.

8.1.2 Selected Networks Repeat Results

From each input resolution size, the best neural network model was selected for five repeated runs to evaluate training repeatability and relative performance between input resolutions. The selected networks were:

- 16in-32c(b)3-64c(b)3-128c(b)3-512fc(d)-16b(d)-4c-64batches_2017-03-03_16-56-54.tflearn9212
- 32in-32c(b)6-64c(b)6-128c(b)6-512fc(d)-16b(d)-4c-64batches_2017-03-03_21-27-50.tflearn9445
- 64in-32c(b)11-32c(b)11-32c(b)11-512fc(d)-16b(d)-4c-64batches_2017-04-13_04-03-00.tflearn9562
- 128in-32c(b)5-32c(b)5-32c(b)5-512fc(d)-16b(d)-4c-64batches_2017-04-13_11-09-54.tflearn9241

To indicate similarity (or lack of it) between repeat runs, boxplots were used to show the performance metrics. The central red mark indicates the median with the 25^{th} and 75^{th} percentiles represented at the box ends, the whiskers indicate outliers not considered extreme. Extreme outliers are plotted separately. The notches in the boxplot indicate the intervals of the data. If notches between boxplots overlap, then they are not significantly different at 5% significance level.

Overall, the results summarised in Figure 8-4 show reasonable repeatability of loss and validation accuracy measurements. Results at 128x128 pixel input size show the greatest variance of all repeat groups, however this may be due to poor selection of hyperparameters as full testing could not be completed at this resolution. Variance within measurements of the loss results are generally higher than for validation accuracy.

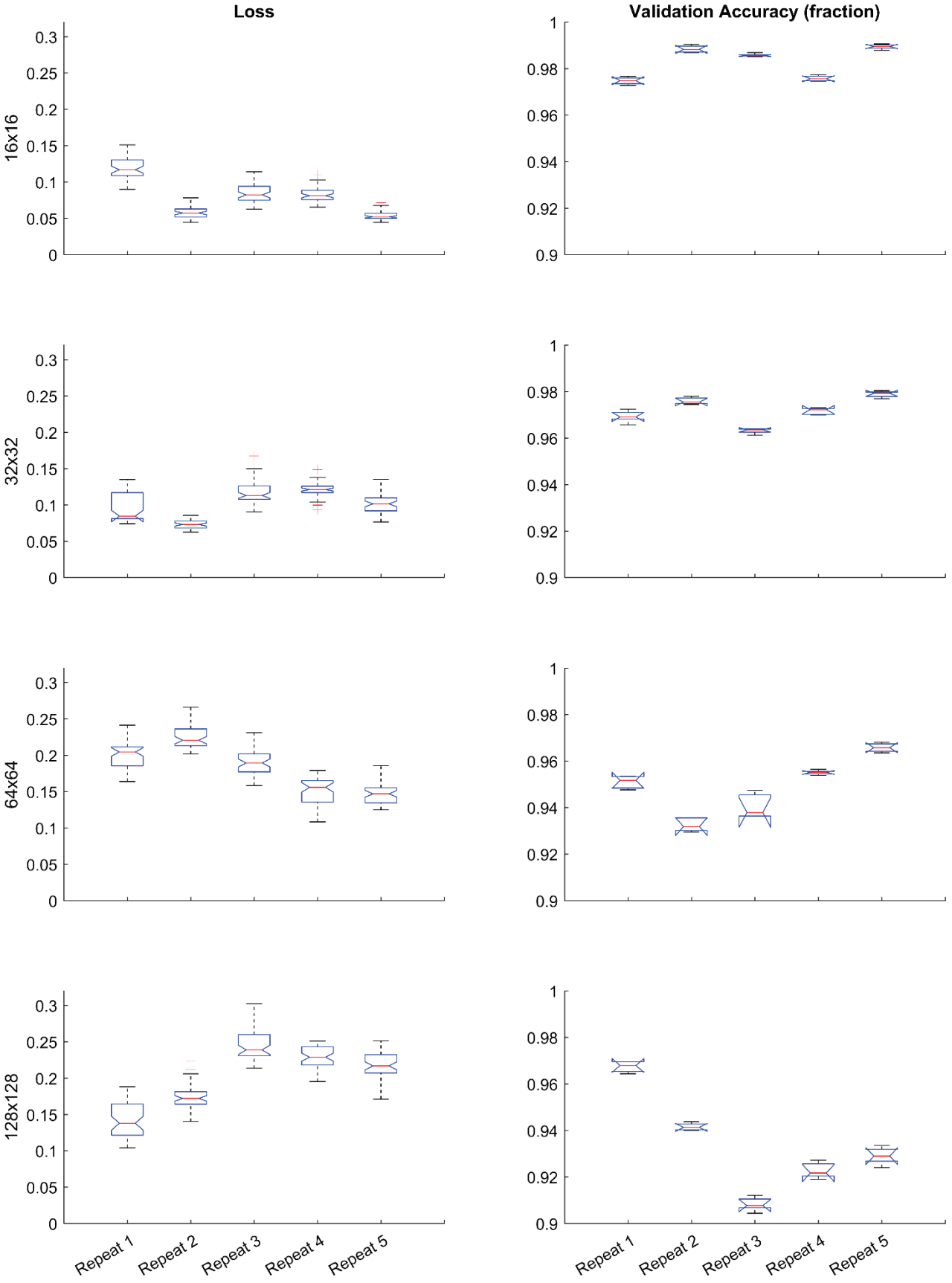


Figure 8-4: Loss and Accuracy Measurements for Repeat Testing

It is difficult to directly compare significance intervals between loss and validation accuracy measurements as the measurements do not conform to the same measurement ranges. Validation accuracy ranges from 0 to 1 in a linear scale while loss is a logarithmic measurement ranging from 0 to infinity. However, the upper limit of loss is usually calculated from random chance classification, which in the case of the four-grade case froth flotation case study would be approximately 0.6. More classes result in a higher upper bound for the loss value of a classifier.

Starting with 16x16 input resolutions, there is variability between repeated tests, only two of any tests at best being statistically similar to each other for both loss and validation accuracy. However, the total validation accuracy range is quite small, from 0.972 to 0.992 with variability between repeats is much lower than that observed between hyperparameters during 16x16 input testing. The interval of the loss measurement is smaller but with larger outliers compared to validation results.

Repeated accuracy measurement for 32x32 pixel images closely follows that of the 16x16 pixel images, with at best two of any repeats being similar. However, the repeated loss measurements in Figure 8-4 show somewhat more variance with only repeats 3 and 4 having statistically similar means. The increased variance may be attributed either to increased overfitting potential of the larger number weights in the 32x32 pixel neural network, or to increased variability of textural features which will feature more strongly at 32x32 pixel size than 16x16.

Unlike the smaller input resolutions, only repeats 1 and 4 of accuracy measurements are statistically similar for 64x64 pixel inputs. It is likely that the increased impact of textural features of this higher input resolution is increasing variability in the accuracy measurements. Conversely, the variability in loss measurements for 64x64 pixel inputs is similar to the 32x32 pixel measurements, only two repeats have statistically similar means. Again, this is likely due to increased impact of textural features from the larger input resolution However, the larger input size may present clearer textures, allowing for better classification and reduced outliers in the loss measurement compared to 32x32 pixel input images.

For 128x128 pixel input images, only two of the repeats have statistically similar validation accuracy, with similar variance within each repeat to other input resolutions. The trend continues in higher resolution repeats with the loss measurements for 128x128 pixel inputs. Only two of the repeats are statistically similar and variability within measurements have increased once again.

The overall trend among the repeated runs for each resolution is some variance between repeats, but not as great as that between hyperparameters for those resolutions. Variability increases with input resolution which reflects the increased number weights which must be retrained to similar final performance. This indicates that there is reasonable repeatability within testing and that the results in Section 8.1.1 may be considered valid representations. However, additional repeats are recommended for future testing to reduce variability within each test run.

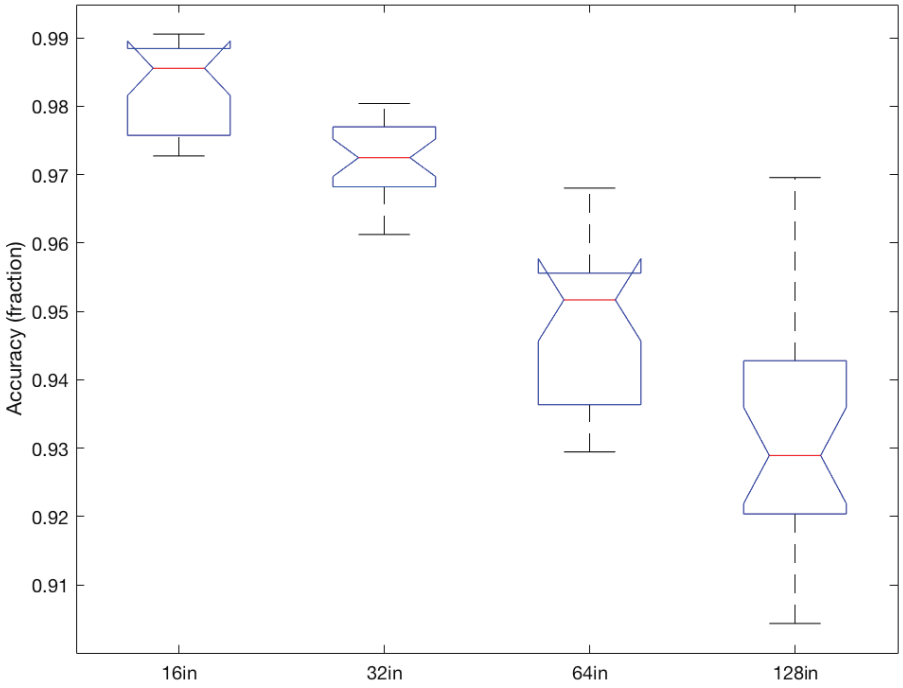


Figure 8-5: Input Resolution Mean Accuracy Results Comparison

Aggregating the repeated data for each resolution shows a clear and statistically significant trend of decreasing validation accuracy with increased resolutions. As Figure 8-5 shows, this is coupled with a general increase of variability and outliers with increasing resolution. The trend for reduced performance with increasing input resolution is also shown in the loss measurement collated in Figure 8-6. Variability within the loss results is more consistent across input resolutions

These trends are likely due to interaction of spectral and textural features, and increasing network complexity with input resolution. At lower resolutions spectral features are expected to dominate as the quality of textural features will be low or insignificant. Increasing the input image resolution also increases the clarity of textural features within the images. These complex textures are likely to form a complex feature space which describes information beyond that of the four platinum grade classes, increasing classification difficulty.

Further examining Figure 8-6, variance within each resolution's loss measurement is more consistent across runs, but still increases with increasing resolutions. The similarity in variance between loss measurements compared to accuracy measurement is likely due to the much larger number of images under consideration during loss measurement than during accuracy validation. The larger population of images results in a broader distribution of measurements and extreme outliers in the accuracy measurement likely fall within expected variances of the larger results population in loss measurements.

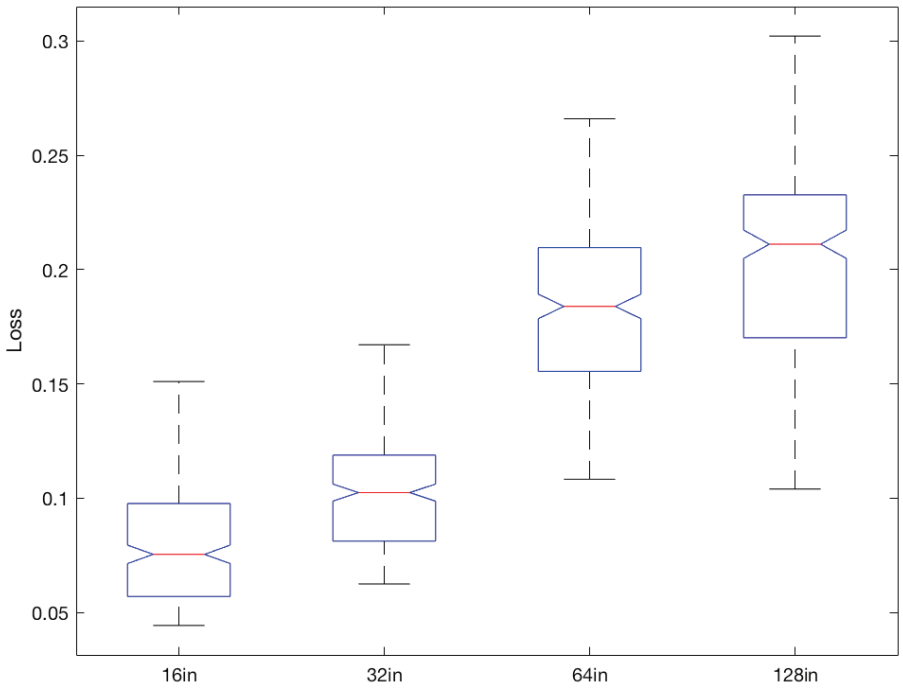


Figure 8-6: Input Resolution Mean Loss Results Comparison

8.1.3 Visualising Network Parameters

To better understand the properties of the neural network, visualisation strategies were applied to a selected network. To increase legibility of the visualisations, they were only applied to the 128x128 pixel size input networks as they produce the largest outputs for the visualisations and accuracy and loss was reasonably similar for all networks tested. This also has the benefit of ensuring that strong textural features are presented. It is expected that the visualisations would be broadly similar across all resolutions, with textural features being less apparent at lower resolutions, and spectral features dominating.

Figure 8-7: Base Image Used for Network Visualisations

There are two visualisations used to explore network properties: deepdream visualisations perform image space gradient descent and develop an image which maximally activates a convolutional filter, and intermediate layer outputs which shows the effects of convolution on a sample image. These visualisations are limited the convolutional layers, as the data output is two dimensional in nature and produces meaningful visualisations. The deepdream visualisations may be applied to later, fully-connected, layers, though this requires that max-pooling layers and ReLU activations are not used in the convolutional layers as they cause gradient collapse.

For visualising layer outputs, the first image of the 128x128 pixel testset was fed into the network, the original image can be seen in Figure 8-7. Deepdream visualisation starts with an image consisting of random noise with a uniform distribution and then calculates an optimal image.

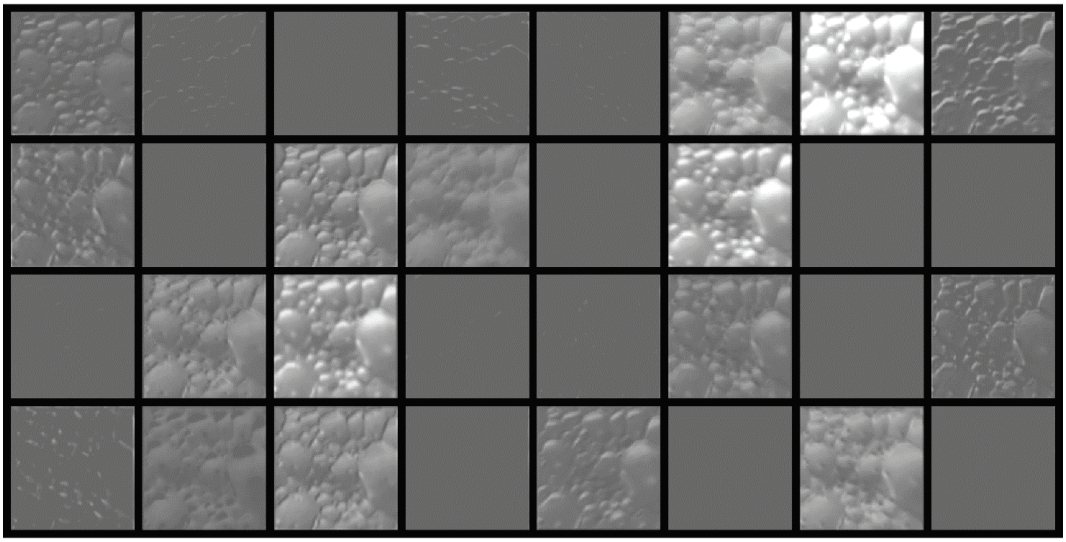


Figure 8-8: Output of Filters from First Convolutional Layer

Starting with Figure 8-8, the output of each convolutional filter in the first convolutional layer can be seen. At this point, approximately half of the filters output images of reasonably strong intensity with minimal textural enhancement, the textural appearance of output images is not markedly different than the original input image. The greatest change appears to be related to intensity and possibly colour channel extraction.

Examining the deepdream images for the first convolutional layer in Figure 8-9 corroborates the first layer outputs. The ideal inputs for each filter appear to be only spectral, indicating that minimal textural searching occurs within the first layer of the neural network and confirms that the layer outputs are largely a result of spectral rather than textural transforms.

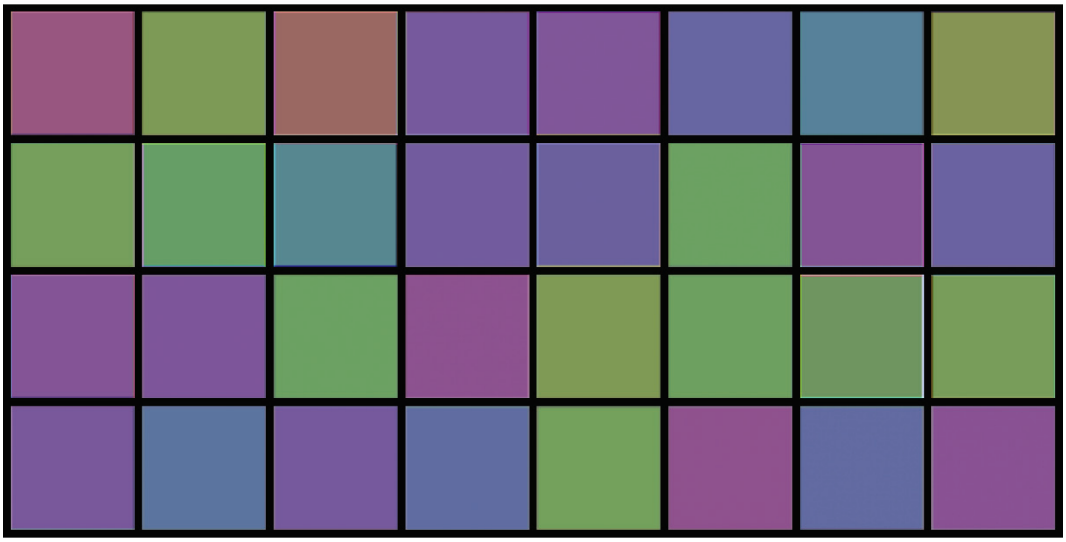


Figure 8-9: Deepdream Optimal Inputs for Filter Activation in First Convolutional Layer

It should be noted that, while the individual images in both figures correspond to the same filters, similar results in one do not necessarily correspond to similar results in the other. This is due to the random variability in generating the deepdream images. While the intermediate outputs are stochastic and repeatable in the final network, the deepdream generation relies on a random image as a starter seed. Fluctuations within the random distribution results in slightly different results for each filter each time a deepdream image is generated. The deepdream results should be interpreted as a general guide to what causes strong activations within a specific layer. For the first convolutional layer, this would be spectral features.

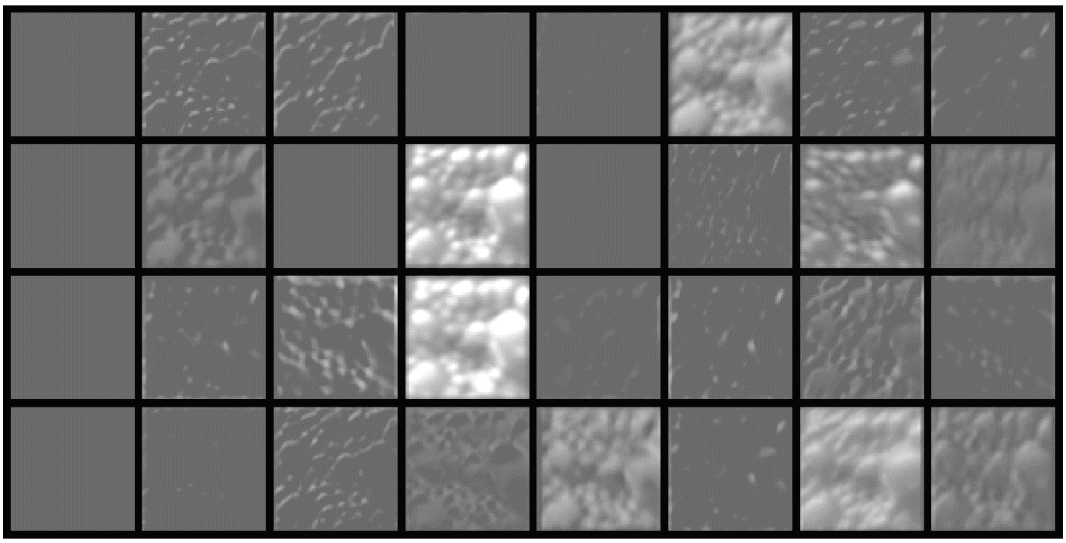


Figure 8-10: Output of Filters from Second Convolutional Layer

Moving onto the second convolutional layer, it can be seen in Figure 8-10 that more significant textural changes are occurring in the filter outputs, the textural appearance of the outputs are visibly different from the original input image. There appears to be directionality differences between filters as well as rudimentary

edge detection occurring within some. Additionally, the light intensity levels become more extreme, either very bright or very dark features are being amplified in the layer outputs.

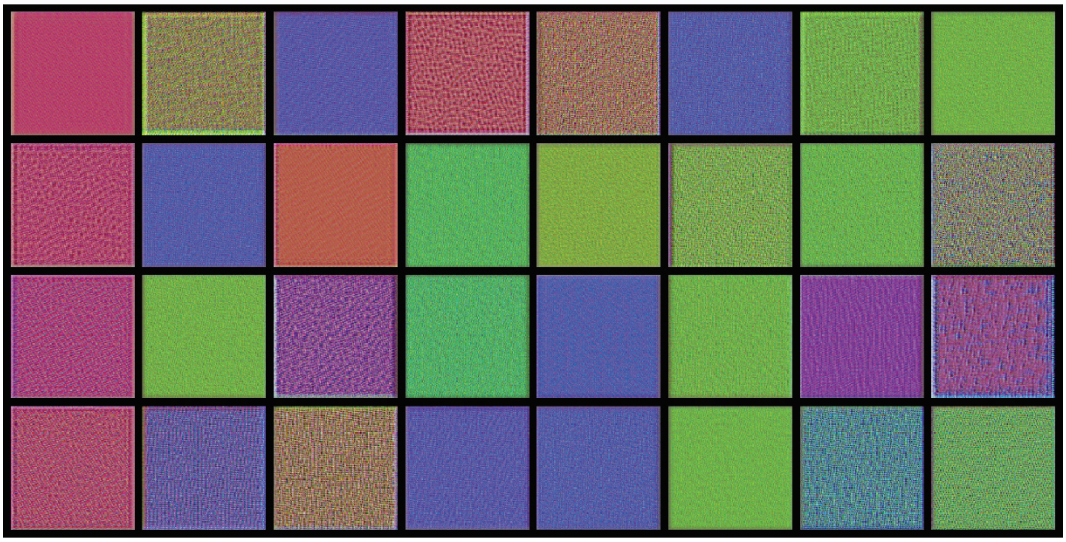


Figure 8-11: Deepdream Optimal Inputs for Filter Activation in Second Convolutional Layer (Contrast was enhanced in this image for visualisation purposes)

Comparing the deepdream visualisations for the second layer in Figure 8-11 with those from the first layer shows a marked increase in fine textural structure. The second layer has begun to evaluate textural features in additional to spectral ones, with maximum activation occurring with mottled (e.g. column 4, row 1) or rippling textures (e.g. column 8, row 3). As the next layer results show, these smaller textural features combine to form larger structures as predicted in the theory of convolutional network operation (Fukushima, 1980).

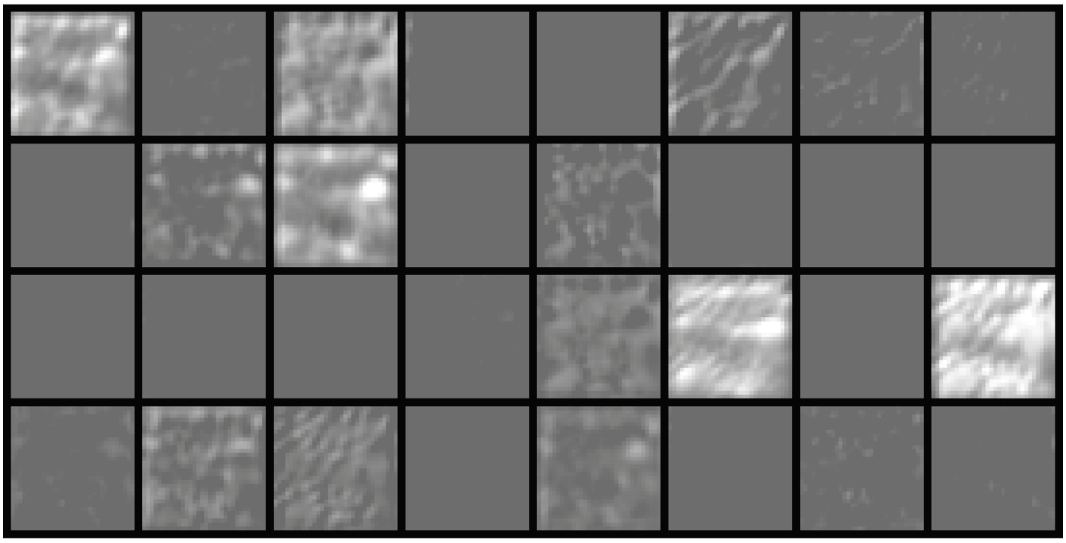


Figure 8-12: Output of Filters from Third Convolutional Layer

In the third and final convolutional layer, further abstraction of the textural features occurs as indicated in Figure 8-12. Approximately half of the filters produce a significant output which is difficult to recognise from

the initial input image. These are the hidden visual features which define major differences between classes and are fed into the fully-connected layers where the feature space is mapped and classification is learned.

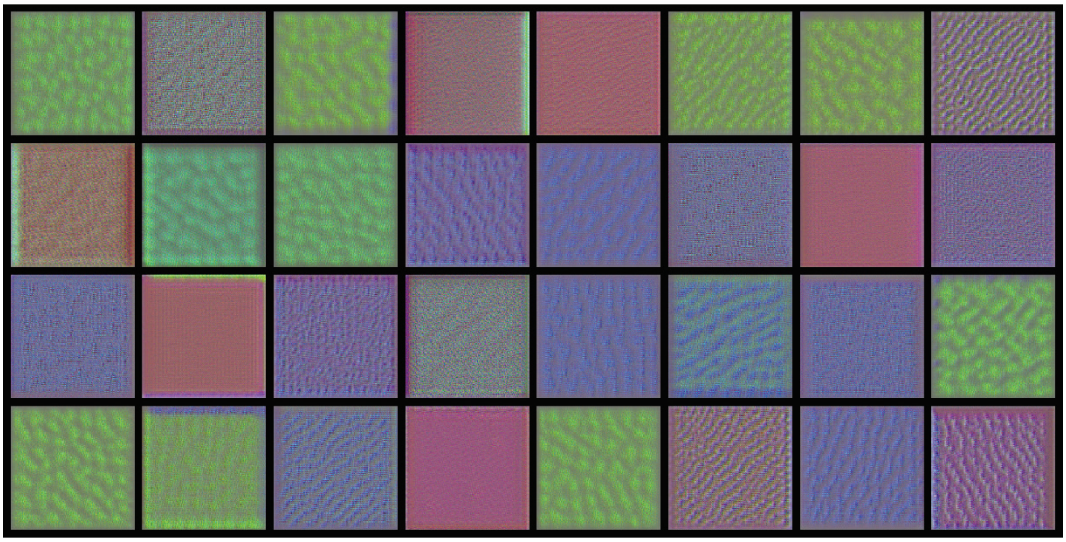


Figure 8-13: Deepdream Optimal Inputs for Filter Activation in Third Convolutional Layer

As expected from convolutional neural network theory, the ideal features in the final, third convolutional layer appear to be textures with large structures derived from smaller structures in previous layers. Figure 8-13 shows that the final layer expects textural and spectral features to activate layer neurons.

While there are also strong indicators of directionality sensing within the final layer, as initially described, the deepdream images are variable depending on bias within the randomly initialised seed images. It is not possible to determine if there is a specific direction in the textures which is of interest for classification, but only that directionality is encoded in the network. It is possible that this may cause some variance in convolutional features as directionality within the input images changes.

These visualisations have confirmed that both textural and spectral features were evaluated by the network during training, and so these features can be expected to play a role in classification. The difficulty with mineral processing images and these visualisation tools is that often it is hard to ascribe meaningful identities to the outputs.

Convolutional neural networks have mostly been used to identify objects within large image databases which are recognisable to humans casually examining the images (Mordvintsev, et al., 2015). When tools such as deepdream visualisation is used on networks trained for these tasks, individual filters may show distinct objects or portions of objects as their ideal input. As Figure 8-14 shows, the GoogLeNet architecture which classifies common objects, a generated flower-like structure is discernible in the filter which is strongly activated by flowers in the image (Szegedy, et al., 2015). The deepdream image here are more readily interpretable by humans than the images generated from the froth flotation CNN and shows that this filter is responsible for detecting flowers.

For flotation froths, the deepdream images are much more abstract, as has been indicated in the prior visualisations.

Figure 8-14: Deepdream Optimal Input for Single Filter in Layer in a Benchmark Classifier (This filter detects the flowers within an image and produces flower-like structures when applying the deepdream algorithm)

8.2 Feature Properties and Extraction Speed Results

Using the four networks selected in Section 8.1.2, the networks were rebuilt with the originally trained weights to output 16 features as a feature set. Due to the nature of the hyperbolic tangent activation function used in the hidden layer from which the features are extracted, the values of each feature in the set range continuously from -1 to +1.

Once the neural networks are trained, performing feature extraction is very quick, and as Table 8-1 shows, the quickest of all textural methods.

Table 8-1: Feature Extraction Speed

Extractor	Feature Sets Per Second
CNN	1881
LBP	8
GLCM	713
Wavelets	848
Spectral	8569

The average number of feature sets extracted per second across all input resolutions was measured for each method using identical computer hardware. Convolutional neural networks provide competitive rates of feature extraction and are the fastest amongst the methods that consider textural features.

It should be noted that the highest input resolution, 128x128 pixels, is likely to produce features from traditional feature extraction methods which most closely matches the expected performance from prior work by (Kistner, 2013). The hyperparameters taken from the work of (Kistner, 2013) were optimised for an input resolution of 224x244. 128x128 pixel input is the closest size to that in the original work while still being small enough to be successfully trained with CNNs with the limited computational resources available.

A detailed example of dataflow and calculations through one of the trained convolutional neural networks is provided in Appendix A.

8.3 Soft-sensor Results

Using features generated from selected CNNs and traditional methods for comparison, the performance of these features can be evaluated within an inferential soft-sensor.

Both linear and non-linear classification models were used to evaluate the performance of various features and evaluate whether neural network features are suitable for monitoring of flotation froth.

The networks are trained using features extracted from the training dataset. Final classification performance is evaluated with the testing dataset, which has not been used to train any feature extractor or soft-sensor model.

It is important to note that features generated by the convolutional neural network are influenced by the classification methods used during their training. As the fully-connected layers are large and thus capable of complex correlation, it is expected that the convolutional features will also be complex. This may explain the difficulty of LDA models in accurately predicting grade for CNN features as shown in the next section.

8.3.1 Linear Discriminant Analysis Classification

Using linear discriminant analysis results in uniformly poor soft-sensor performance across all feature extractors as shown in Figure 8-15. Note that the classification loss for each model listed here is distinct from the neural network loss which is a measurement of cross entropy. Classification loss is the weighted fraction of images misclassified.

The uniformly poor performance indicates that the feature space is not linearly separable, and that good performance is likely limited by the soft-sensor model and not the features. Despite the poor performance, some interesting observations can be made from this data. CNNs perform on par or slightly better to the other feature extraction methods. This validates the methodology of using features from a hidden layer in the network to eliminate potential bias from the non-linear modelling capabilities of neural networks. If

linearization occurred in addition to dimensionality reduction in the network, it would be expected that even a linear soft-sensor model would provide good classification performance when using the CNN features.

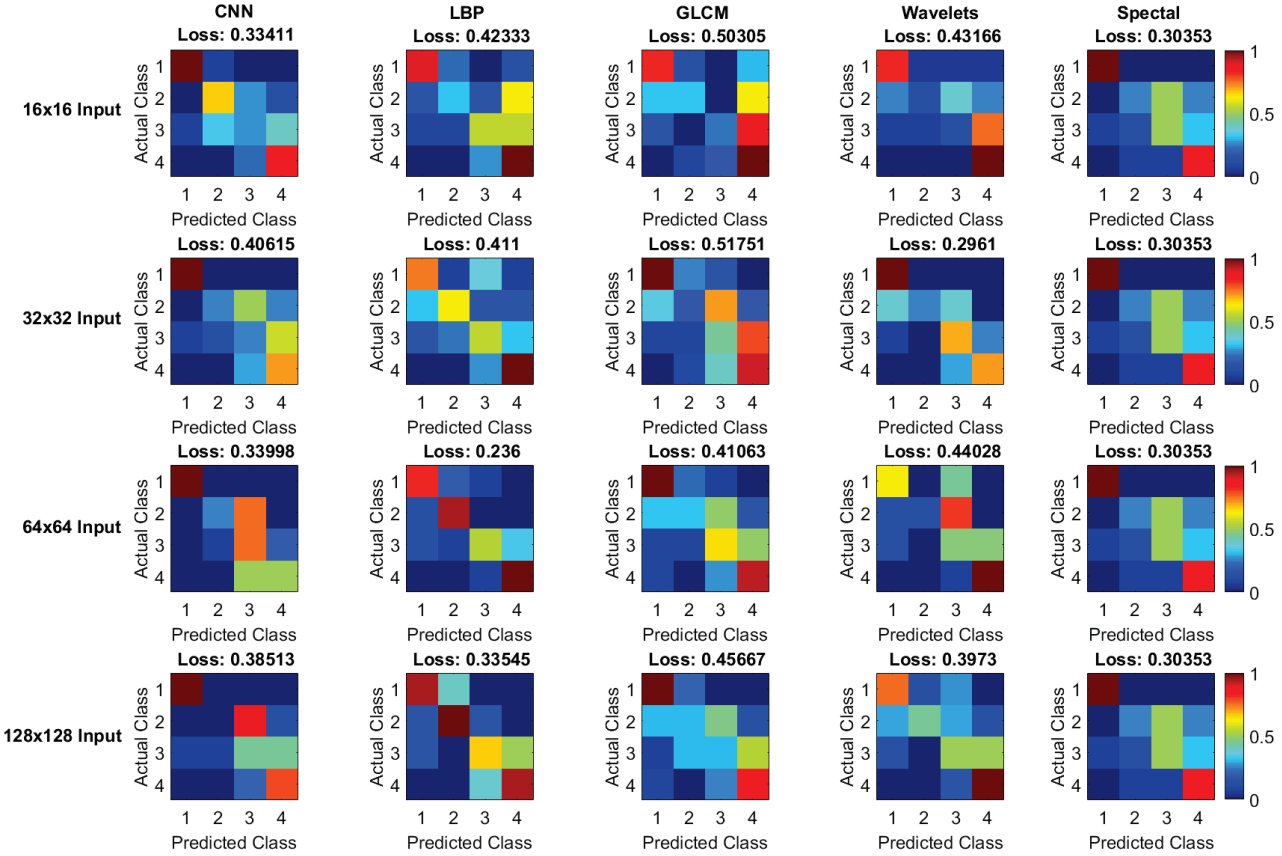


Figure 8-15: Classification Results for Features with Linear Soft-Sensor Model

Also, note the uniformity of the spectral features. Not only is the loss identical between input resolutions, but so are the specific misclassifications. This confirms that the dataset for each resolution was correctly downscaled from the larger resolutions and that coherency was maintained between images in each resolution. The assurance of uniformity between an image scaled to different resolutions is due to the method of scaling used, bicubic interpolation, which attempts to maintain both relative spatial relationships and spectral levels within the scaled images.

8.3.2 k-Nearest Neighbour Classification

In contrast to the linear classification in Figure 8-15, the results for k-Nearest Neighbour classification as summarised in Figure 8-16 show much better performance, and variance between feature extractors.

As with LDA, spectral features show uniform performance across input resolutions and have the best classification results for 64x64 and 16x16 input resolutions, indicating that spectral features have a strong impact on classification.

Local binary patterns beat all feature extraction methods at 16x16 and 32x32 pixel sizes, though these results should be considered within a broader context including computational resources required for implementation. While the performance is the best, it is also at the cost of very slow extraction speed and a high number of features within each feature set. For 16x16 pixels the number of input variables is reduced from 256 to 243, quite poor as a method of dimensionality reduction despite the good classification results.

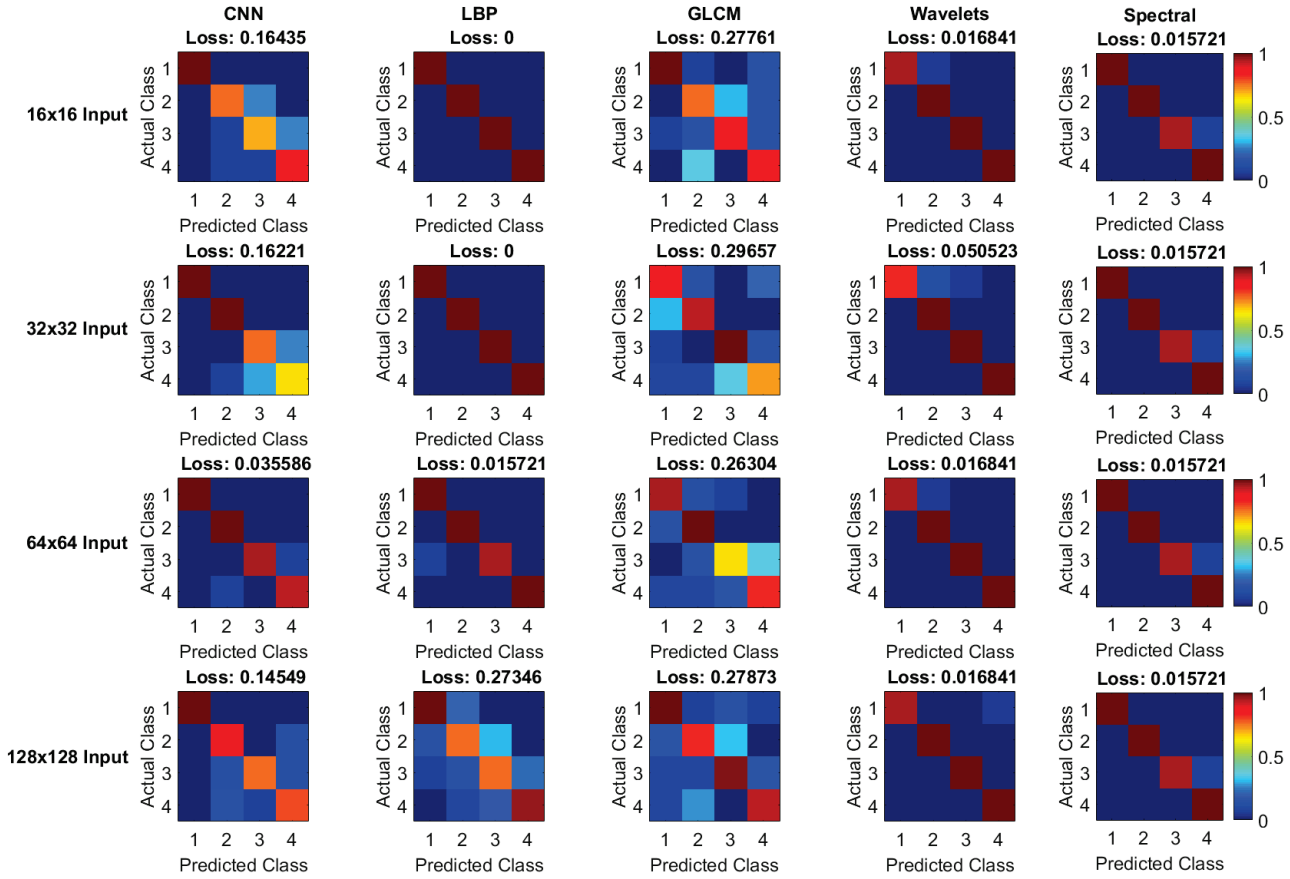


Figure 8-16: Classification Results for Features with Non-Linear Soft-Sensor Model

Grey-level co-occurrence matrices show better performance than with linear soft-sensor models, but their performance is worse (considerably so at lower resolutions) than all feature extractors. It may be that the hyperparameters for GLCMs are more sensitive to input resolution, however, in the original work of Kistner GLCMs were also found to be the worst of all textural feature extractors (Kistner, 2013).

Wavelets produce good classification results across all input resolutions and are the fastest traditional texture feature extractors. Comparing the convolutional neural network features to them shows worse but still reasonable results.

8.3.3 CNN Features

The CNN features are neither the worst nor the best feature extractor for the platinum froth case study. They outperform LBPs at higher resolutions and are capable of extracting features faster than any other textural method tested in this work. They can extract both textural and spectral features for classification and have

hyperparameters which are independent of input resolution (aside from valid convolutional filter sizes). With further training and additional training data, increased performance may be achieved.

A factor not evaluated in the platinum flotation case study is the resilience of the feature extractors to changes in imaging conditions. It is expected that CNNs would perform well under a variety of imaging conditions, however, within the limited dataset of the case study it was not possible to evaluate this for any of the feature extractors. There were concerns of overfitting in the CNN features with the small dataset from the case study, but in the final results limited to no overfitting occurred. However, the small dataset may have contributed to the disparity of performance between the larger and smaller network architectures. Batch normalisation and dropout both contributed to validation accuracy remaining steady in the network training results.

As a proof of concept for their application in flotation froth soft sensing, these results are sufficient to warrant further work and deeper evaluation.

8.4 CNN Soft-Sensor System Development

The performance and imaging requirements of vision-based soft-sensors form part of objective 3 where industrial requirements for these systems must be evaluated. These are interdependent on the type of feature extractor used, data availability, and expectations from industry.

An idealised soft-sensor for primary monitoring would poses the following properties:

- Measure process conditions without error
- Have robust and valid output for all possible process conditions
- Update inferred variables as quickly as possible
- Reliable operation and fail-safe error modes
- Cheap and fast to implement

However, not all of these properties are feasible and may be mutually exclusive in a realisable soft-sensor (Fortuna, et al., 2007). Additionally, high performance soft-sensors require significant ancillary equipment and computational infrastructure where they are to be implemented. A more realisable soft-sensor with currently used methods would have the following properties:

- Have a reasonably low error associated with the method of sensing measured process conditions
- Have valid output for most commonly encountered process conditions
- Update inferred variables at a rate comparable or better than process dynamics
- Have graceful failure states which can be detected early
- Cheap enough to implement at critical process units/streams

These are more realistic goals, but within the context of minerals processing, even these may be a challenge. Mineral processes have a large number of unmeasured variables which affect process performance, any soft-sensor model must be resilient to these changes which cannot easily be measured with existing process sensors (González, 2010). Image-based soft-sensors attempt to measure some of these unmeasured variables, but image-based sensing comes with its own set of challenges.

Image data is significantly larger than most univariate process data used in process control. They present a challenge not only to data storage requirements, but consistent acquisition, storage, and processing of these images present new technical challenges on a process plant. As has been discussed in prior sections of this work, feature extraction as a method of dimensionality reduction is required to use image data in traditional soft-sensor frameworks.

8.4.1 Imaging Requirements

The challenge with imaging process conditions is ensuring high quality images are produced under controlled imaging conditions. Traditional texture feature extractors are sensitive to changes in lighting conditions, and unless specially controlled or accounted for in soft-sensor models it will result in reduced accuracy or erroneous results (Tan & Triggs, 2010). Additionally, these feature extractors may not be variant to rotation or scaling which requires specific installation of image capture equipment or re-optimising feature extractors for each installation.

In traditional soft-sensor implementations, this meant fixed camera positions with controlled lighting. However, this is not feasible in the long term for some processes. Ambient light might be too difficult to control on large flotation units or controlled lightning might be too expensive to run continuously.

CNNs alleviate some of these issues. They are also sensitive to changes in lighting conditions, but less so than traditional methods and are easier to train to operate effectively under changing imaging conditions. As has been shown in Section 8.1.3, CNN features may make use of both textural and spectral properties within images, which may allow mitigation of illumination bias within one feature property or the other.

As insufficient data was available of differing lighting conditions in the flotation case study, this property of CNN feature extractors could not be evaluated. Within the context of flotation froths, process dynamics are typically on the order of 10s of seconds to several minutes, any image capture and feature extraction must be performed faster than this in order to sufficiently monitor the process (Zanin, et al., 2009). All of the feature extractors evaluated in the work can extract features within these timeframes.

8.4.2 Soft-Sensor Requirements

The traditional feature extractors and soft-sensor models evaluated in this work fulfil some of the requirements of a realistic soft-sensor implementation. Graceful failure cannot be evaluated due to limited

datasets. In terms of cheap implementation however, some of them fail. Each of the traditional methods evaluated are computationally intensive on modern desktop workstations. In a process plant, simpler and more cost effective soft-sensor implementation are preferred when monitoring is done locally. Imaging may also be performed remotely, with feature extraction and soft-sensor correlation done at a central computational resource. In this case efficiency is still preferred as many soft-sensors may have to be updated simultaneously.

Convolutional neural networks (and other neural networks) are unique in that their heavy computational requirements are mostly limited to their training stage. Once trained, the neural networks may perform classification at very high speed as shown in Table 8-1 on a workstation. Conversely, trained neural networks may be implemented on low-cost local soft-sensors while still maintaining sufficient classification performance (Alonso, et al., 2010).

With sufficient training data, CNN models can approach some of the idealised soft-sensor capabilities more easily than traditional feature extractors. They respond well to large and non-linear changes in input data and can potentially be trained to perform well over a wide range of conditions and report on fault conditions (Goodfellow, et al., 2016). CNNs are also capable of graceful performance degradation, being resilient to noise in input images which is common in industrial vision systems in harsh conditions such as mineral processing (Xie, et al., 2012).

Chapter 9: Conclusions and Recommendations

The original objectives of this work as summarised in Chapter 1 are:

1. Develop and evaluate a deep learning system for froth flotation sensing.
2. Compare deep learning feature extractors with traditional ones and evaluate the performance requirements for a soft-sensor in froth flotation
3. Contextualise results of deep learning system development with industry requirements and challenges for vision-based soft-sensing systems

These were successfully achieved to varying degrees. For the first objective, an in-depth analysis and development of a deep learning soft-sensor for froth flotation was documented in this work. The deep learning methods used was convolutional neural networks and their performance was comparable to traditional feature extractors and soft-sensors. As part of the second objective, a framework was developed which successfully compared textures extracted with CNNs to those from other feature extractors in an unbiased manner. During this development, several features and requirements for a soft-sensor in mineral processing were evaluated. Finally, through the course of achieving the first two objectives, a framework for producing image-based datasets was defined and several datasets were generated for fair comparison between feature extraction methods. Industrial imaging requirements were evaluated to determine the optimum imaging conditions and sensing requirements for image-based soft-sensing in mineral processing with an awareness of practical limitations applicable in industry.

The usage of convolutional neural networks for feature extraction in flotation froths presented a unique challenge. The tools and use cases for CNNs have focussed overwhelmingly on classification tasks in consumer multimedia applications. As such these tools depend on a human designer being able to recognise everyday objects in neural network visualisations or to be able to sensibly label data. These are currently of low utility to process engineering, requiring substantial evaluation as many of the image properties inherent to mineral processes are not readily discernible by humans.

9.1 Training CNNs on Flotation Froths

Training convolutional neural networks on mineral processes was relatively straightforward once batch normalisation was applied. Results were good and resilient to changes in hyperparameters, indicating a robust network architecture or a limited hyperparameter search space.

Testing repeatability of training was successful, with inter-repeat variance being lower than variance seen in hyperparameter testing. In future work, it is suggested that multiple repeats are carried out for each hyperparameter tested. This should be done not to ensure repeatability of results, but to reduce the variance in network training performance metrics.

The neural networks tested in the course of this work all performed within a relatively narrow windows of good validation accuracy and training loss. However, the variance in performance metrics often exceeded changes in performance due to hyperparameter selection. While this provided for robust feature extraction performance, it limited determining which hyperparameters had the largest effect on training performance. Smaller batch sizes and smaller input resolutions were the most statistically significant changes for improved training performance. The unexpected result of smaller input resolutions leading to better training performance was likely due to trade-offs at that resolution in preferring spectral features over textural features for classification, or that the small dataset was insufficient to effectively train the larger models.

As the visualisations in Section 8.1.3 indicate, spectral features play a large role in the features extracted from the CNNs. Unlike traditional feature extractors, the neural networks could utilise these in addition to the expected textural features. Additional testing indicated that naïve spectral features very closely correlated with expected classes for the froth flotation case study. Neural networks with a smaller input resolution could more effectively utilise these features and not be affected by greater variance in the textural features, resulting in their superior performance during training.

9.2 CNNs as Textural Feature Extractors

CNNs exhibit good performance as texture feature extractors within a soft-sensor. They can produce features for a large number of images rapidly, and their classification performance was comparable to features derived from traditional methods. In contrast to expectations from training, 64x64 pixel input size features gave the best results and not the smallest input resolution features.

While training in a supervised manner, the neural network may have been more effective at correlating features from smaller input resolutions to class data. In the soft-sensor implementation, features more dependent on textural properties from the higher resolutions may have been easier to model using the k-NN classifier. It may also be due to overfitting which has occurred at the smaller input resolutions leading to worse performance with the previously unseen testing data.

However, it was most likely due to method of selecting networks for each input resolution, as only the loss in the last step of training was considered during selection. Regardless, the performance across all input resolutions was sufficient for process monitoring purposes and greater performance is likely with further training and optimisation. As a proof of concept, convolutional neural networks have been successfully applied in correlating images with process data in a minerals processing context.

9.3 Outlook and Further Work

While the primary objectives of this work were achieved, numerous questions remain. Training results for neural networks and subsequent analyses were limited by the small dataset available in the platinum froth flotation case study.

Deep learning methods can intuitively detect and learn complex dependencies in data without requiring prior knowledge of data structure or additional hyperparameters during training. However, this also makes training deep methods difficult with limited data as training will rapidly overfit with small datasets. Additional methods such as stochastically generating variants may be explored to further enhance existing datasets but diminishing returns are expected with additional variants from those already tested. When prior knowledge is available it may be applied to help model the dataset with traditional feature extraction methods and facilitate effective soft-sensors when training data is limited. Additionally, deep learning methods are more sensitive to datasets which do not reflect a wide range of process conditions than traditional feature extractors and soft-sensors. To ensure that underlying relations within the dataset are learned, a representative dataset must be developed for the process. A small dataset or one which is representative of only a small portion of expected process conditions is unlikely to generalise well and may have undefined operating characteristics when encountering abnormal process data after training.

Many of the advantages of deep learning methods over traditional methods are thus only realisable with sufficient training data. Industry awareness of image-based monitoring is growing, and images of processes and various process measurements are now actively being stored as data capacity costs drop. However, datasets remain limited due to difficulty in reconciling various process data streams and sensible handling of time-based data from sequential process units. Researchers also face the difficulty of locally accessing and processing these large datasets which may require computational and storage resources out of reach for the average researcher.

Future work on deep learning systems in the froth flotation will have to rely on new and better methods of data enhancement until larger process datasets can be established. However, now that a proof of concept has been established, more complex implementations of deep learning systems can be considered. Alternatives such as autoencoders can be further considered for unsupervised training with sufficient computational power and data augmentation can be performed in the feature space to generate new images for training.

Of interest is the development of complete neural network soft-sensors, eliminating the feature extraction and secondary training step of traditional soft-sensors. As mentioned in Section 8.3 the features developed by the convolutional neural network are influenced by the classification method, running the network with the same classifier used to train it can potentially result in superior performance. Another advantage is easily combining other, related process data into the neural network sensor, potentially creating improved soft-sensors. The neural network sensors may outperform traditional soft-sensors and provide advanced fault detection in addition to key performance indicator monitoring and data for process control.

Chapter 10: References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenburg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X., 2015. *Large-Scale Machine Learning on Heterogeneous Systems.* [Online] Available at: http://tensorflow.org/

Aggarwal, C. C., Hinneburg, A. & Keim, D. A., 2001. *On the Surprising Behaviour of Distance Metrics in High Dimensional Space.* Heidelberg, Springer Berlin, pp. 420-434.

Aldrich, C., Marais, C., Shean, B. J. & Cilliers, J. J., 2010. Online Monitoring and Control of Froth Flotation Systems with Machine Vision: A Review. *International Journal of Mineral Processing,* 96(1-4), pp. 1-13.

Alonso, G. A., Istamboulie, G., Ramírez-García, A., Noguer, T., Marty, J.-L. & Muñoz, R., 2010. Artificial Neural Network Implementation in Single Low-Cost Chip for the Detection of Insecticides by Modeling of Screen-Printed Enzymatic Sensors Response. *Computers and Electronics in Agriculture,* 74(2), pp. 223-229.

Al-Thyabat, S., 2008. On the Optimization of Froth Flotation by the use of an Artificial Neural Network. *Journal of China University of Mining & Technology,* 18(3), pp. 418-426.

Balakrishnama, S. & Ganapathiraju, A., 1998. Linear Discriminant Analysis - A Brief Tutorial. *Institute for Signal and Information Processing,* Volume 18.

Baldassi, C., Borgs, C., Chayes, J. T., Ingrosso, A., Lucibello, C., Saglietti, L. & Zecchina, R., 2016. Unreasonable Effectiveness of Learning Neural Networks: From Accessible States and Robust Ensembles to Basic Algorithmic Schemes. *Proceedings of the National Academy of Sciences,* 113(48), pp. 7655-7662.

Baldi, P., 2012. Autoencoders, Unsupervised Learning, and Deep Architectures. *Journal of Machine Learning Research,* Volume 27, pp. 37-50.

Barbian, N., Hadler, K., Ventura-Medina, E. & Cilliers, J. J., 2005. The Froth Stability Column: Linking Froth Stability and Flotation Performance. *Minerals Engineering,* 18(3), pp. 317-324.

Bengio, Y., 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning,* 2(1), pp. 1-127.

Bengio, Y., 2013. Deep Learning of Representations: Looking Forward. In: A. Dediu, C. Martín-Vide, R. Mitkov & B. Truthe, eds. *Statistical Language and Speech Processing.* Berlin: Springer, pp. 1-37.

Bharati, M. H., Liu, J. J. & MacGregor, J. F., 2004. Image Texture Analysis: Methods and Comparisons. *Chemometrics and Intelligent Laboratory Systems,* Volume 72, pp. 57-71.

Bharati, M. H. & MacGregor, J. F., 1998. Multivariate Image Analysis for Real-Time Process Monitoring and Control. *Industrial & Engineering Chemistry Research,* 37(12), pp. 4715-4724.

Boureau, Y., Ponce, J. & LeCun, Y., 2010. *A Theoretical Analysis of Feature Pooling in Vision Algoritms.* Haifa, International Conference on Machine Learning.

Chan, T., Jia, K., Gao, S., Lu, J., Zeng, Z. & Ma, Y., 2015. PCANet: A Simple Deep Learning Baseline for Image Classification?. *IEEE Transactions on Image Processing,* 24(12), pp. 5017-5032.

Chidzanira, T., 2016. *Invesitgation of the Effect of Particle Size on Froth Stability, MScEng Thesis,* Cape Town: University of Cape Town.

Ciregan, D., Meier, U. & Schmidhuber, J., 2012. Multi-Column Deep Neural Networks for Image Classification. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 3642-3649.

Cubero, S., Aleixos, N., Moltó, E., Gómez-Sanchis, J. & Blasco, J., 2011. Advances in Machine Vision Applications for Automatic Inspection and Quality Evaluation of Fruits and Vegetables. *Food and Bioprocess Technology,* 4(4), pp. 487-504.

Damien, A., 2016. *TFLearn,* s.l.: GitHub.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M. & Brox, T., 2014. *Discriminative Unsupervised Feature Learning with Convolutional Neural Networks.* Montréal, Advances in Neural Information Processing Systems.

Duchesne, C., 2010. Multivariate Analysis in Mineral Processing. In: D. Sbárbaro & R. del Villar, eds. *Advances Control and Supervision of Mineral Processing Plants.* London: Springer, pp. 85-142.

Duchesne, C., Liu, J. J. & MacGregor, J. F., 2012. Multivariate Image Analysis in the Process Industries: A Review. *Chemometrics and Intelligent Laboratory Systems,* Volume 117, pp. 116-128.

Duchi, J., Hazan, E. & Singer, Y., 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimisation. *Journal of Machine Learning Research,* 12(7), pp. 2121-2159.

Elshenawy, L. M., Yin, S., Naik, A. S. & Ding, S. X., 2010. Efficient Recursive Principal Component Analysis Algorithms for Process Monitoring. *Industrial & Engineering Chemistry Research,* 49(1), pp. 252-259.

Eslamimanesh, A., Gharagheizi, F., Mohammadi, A. H. & Richon, D., 2011. Artificial Neural Network Modelling of Solubility of Supercritical Carbon Dioxide in 24 Commonly Used Ionic Liquids. *Chemical Engineering Science,* 66(13), pp. 3039-3044.

Fathi, A., Monadjemi, A. H. & Mahmoudi, F., 2012. Defect Detection of Tiles with Combined Undecimated Wavelet Transform and GLCM Features. *International Journal of Soft Computing and Engineering,* 2(2), pp. 30-34.

Fortuna, L., Graziani, S., Rizzo, A. & Xibilia, M. G., 2007. *Soft Sensors for Monitoring and Control of Industrial Processes.* London: Springer-Verlag.

Fuerstenau, M. C., Jameson, G. & Yoon, R., 2007. *Froth Flotation: A Century of Innovations.* Colorado: Society for Mining, Mettalurgy, and Exploration, Inc.

Fukushima, K., 1980. Neocognitron: A Self-organising Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics,* 36(4), pp. 193-202.

Gallant, S. I., 1993. *Neural Network Learning and Expert Systems.* Massachusetts: Bradford Books.

González, G. D., 2010. Soft Sensing. In: D. Sbárbaro & R. del Villar, eds. *Advanced Control and Supervision of Mineral Processing Plants.* London: Springer-Verlag, pp. 143-214.

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning.* Cambridge: The MIT Press.

Gupta, A. & Yan, D. S., 2006. Flotation. In: *Mineral Processing Design and Operation.* Amsterdam: Elsevier Science, pp. 555-603.

Hall-Beyer, M., 2017. *GLCM Texture: A Tutorial.* [Online] Available at: https://prism.ucalgary.ca/handle/1880/51900 [Accessed August 2017].

Haralick, R. M., Shanmugam, K. & Dinstein, I., 1973. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-3(6), pp. 610-621.

Hecht-Nielsen, R., 1988. Theory of The Backpropagation Neural Network. *Neural Networks,* 1(1), pp. 445-448.

He, K., Zhang, X., Ren, S. & Sun, J., 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance of Imagenet Classification. *Proceedings of the IEEE Internation Conference on Computer Vision,* pp. 1026-1034.

Himmelblau, D. M., 2000. Applications of Artificial Neural Networks in Chemical Engineering. *Korean Journal of Chemical Engineering,* 17(4), pp. 373-392.

Hinton, G. E., Osindero, S. & Teh, Y., 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation,* 18(7), pp. 1527-1554.

Hinton, G., Srivastava, N. & Swersky, K., 2012. RMSProp: Divide the Gradient by a Running Average of its Recent Magnitude. *Neural Networks for Machine Learning, Coursera Lecture 6e.*

Holtham, P. N. & Nguyen, K. K., 2002. On-line Analysis of Froth Surface in Coal and Mineral Flotation using JKFrothCam. *International Journal of Mineral Processing,* 64(2-3), pp. 163-180.

Hubel, D. H. & Wiesel, T. N., 1968. Receptive Fields and Functional Architecture of Monkey Striate Cortex. *The Journal of Physiology,* 195(1), pp. 215-243.

Jaeger, H., 2013. *A Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF, and the "Echo State Network" Approach,* Bremen: Fraunhofer Institute for Autonomous Intelligent Systems.

Jolliffe, I., 2014. Principal Component Analysis. In: *Wiley StatsRef: Statistics Reference Online.* s.l.:John Wiley & Sons.

Kadlec, P., Gabrys, B. & Strandt, S., 2009. Data-driven Soft Sensors in the Process Industry. *Computers & Chemical Engineering,* 33(4), pp. 795-814.

Keys, R., 1981. Cubic Convolution Interpolation for Digital Image Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 29(6), pp. 1153-1160.

Kim, S., Kano, M., Hasebe, S., Takinami, A. & Seki, T., 2013. Long-Term Industrial Applications of Inferential Control Based on Just-In-Time Soft-Sensors: Economical Impact and Challenges. *Industrial & Engineering Chemistry Research,* 52(35), pp. 12346-12356.

Kingma, D. & Ba, J., 2014. Adam: A Method for Stochastic Gradient Optimisation. *arXiv preprint arXiv:1412.6980.*

Kingsbury, N., 2006. *Rotation-Invariant Local Feature Matching with Complex Wavelets.* Florence, IEEE Signal Processing Confence.

Kistner, M., 2013. *Image Texture Analysis for Inferential Sensing in the Process Industries, MEng Thesis.* Stellenbosch : Stellenbosch University.

Kistner, M., Jemwa, G. T. & Aldrich, C., 2013. Monitoring of Mineral Processing Systems by Using Textural Image Analysis. *Minerals Engineering,* Volume 52, pp. 169-177.

Kittisupakorn, P., Thitiyasook, P., Hussain, M. A. & Daosud, W., 2009. Neural Network Based Model Predictive Control for a Steel Pickling Process. *Journal of Process Control,* 19(4), pp. 579-590.

Kluge, A., 2014. *The Acquisition of Knowledge and Skills for Taskwork and Teamwork to Control Complex Technical Systems.* Dordrecht: Springe Science+Business Media.

Krizhevsky, A., Sutskever, I. & Hinton, G. E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. In: F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds. *Advances in Neural Information Processing Systems 25.* s.l.:Curran Associates, Inc, pp. 1097-1105.

Latif-Amet, A., Ertüzün, A. & Erçil, A., 2000. An Efficient Method for Texture Defect Detection: Sub-Band Domain Co-Occurence Matrices. *Image and Vision Computing,* 18(6), pp. 543-553.

LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep Learning. *Nature,* 521(7553), pp. 436-444.

LeCun, Y., Huang, F. J. & Bottou, L., 2004. *Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting.* Washington DC, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Lee, D., 2013. *Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks.* Atlanta, Challenges in Representation Learning.

Leja, J., 1982. *Surface Chemistry of Froth Flotation.* New York: Plenum.

Li, C., Runge, K., Shi, F. & Farrokhpay, S., 2016. Effect of Flotation Froth Properties on Froth Rheology. *Powder Technology,* Volume 294, pp. 55-65.

Lin, B., Recke, B., Knudsen, J. K. H. & Jørgensen, S. B., 2007. A Systematic Appriach for Soft Sensor Development. *Computers & Chemical Engineering,* 31(5-6), pp. 419-425.

Liu, J. J. & MacGregor, J. F., 2008. Froth-based Modelling and Control of Flotation Processes. *Minerals Engineering,* 21(9), pp. 642-651.

Liu, J. J., MacGregor, J. F., Duchesne, C. & Bartolacci, G., 2005. Flotation Froth Monitoring Using Multiresolutional Multivariate Image Analysis. *Minerals Engineering,* 18(1), pp. 65-76.

Marais, C., 2010. *Estimation of Concentrate Grade in Platinum Flotation Based on Froth Image Analysis, MEng Thesis.* Stellenbosch: Stellenbosch University.

Marais, C. & Aldrich, C., 2011. Estimation of Platinum Flotation Grades from Froth Image Data. *Minerals Engineering,* 24(5), pp. 433-441.

Mathe, Z. T., Harris, M. C., O'Connor, C. T. & Franzidis, J.-P., 1998. Review of Froth Modelling in Steady State Flotation Systems. *Minerals Engineering,* 11(5), pp. 397-421.

MathWorks, 2017. *Statistics and Machine Learning Toolbox: fitcknn (R2017b).* [Online] Available at: https://www.mathworks.com/help/stats/fitcknn.html [Accessed 7 November 2017].

McCullough, W. S. & Pitts, W., 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology,* 5(4), pp. 115-133.

Mehrota, R., Namuduri, K. R. & Ranganathan, N., 1992. Gabor Filter-based Edge Detection. *Pattern Recognition,* 25(12), pp. 1479-1494.

Metso, 2007. *A Correlation Between Visiofroth Measurements and the Performance of a Flotation Cell.* [Online] Available at: http://www.metso.com/miningandconstruction/mct_service.nsf/WebWID/WTB-120118-22576-F1987/$File/136.pdf [Accessed 7 August 2017].

Mintek, 2015. *Flotation Control & Optimisation.* [Online] Available at: http://www.mintek.co.za/wp-content/uploads/2015/10/Flotation-Brochure-V2.pdf [Accessed 8 August 2017].

Monadjemi, A., 2004. *Towards Efficient Texture Classification and Abnormality Detection,* Bristol: University of Bristol Department of Computer Science.

Moolman, D. W., Aldrich, C., van Deventer, J. S. J. & Stange, W. W., 1995. The Classification of Froth Structures in a Copper Flotation Plant by Means of a Neural Net. *International Journal of Mineral Processing,* 43(3-4), pp. 193-208.

Mordvintsev, A., Olah, C. & Tyka, M., 2015. *DeepDream - A Code Example for Visualising Neural Networks.* [Online] Available at: https://research.googleblog.com/2015/07/deepdream-code-example-for-visualizing.html [Accessed 7 November 2017].

Nguyen, P. T. & Nguyen, A. V., 2009. Validation of the Generalised Sutherland Equation for Bubble-Particle Encounter Efficiency in Flotation: Effect of Particle Density. *Minerals Engineering,* 22(2), pp. 176-181.

Oquab, M., Bottou, L., Laptev, I. & Sivic, J., 2014. Learning and Transferring Mid-Level Image Representations. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition.* Washington DC: IEEE Computer Society, pp. 1717-1724.

Owen, T. V., 1988. On-Line Analysis at Rooiberg Tin Limited. *Journal of the South African Institute of Mining and Metallurgy,* 88(10), pp. 345-347.

Özdemir, S., Baykut, A., Meylani, R., Erçil, A. & Ertüzün, A., 1998. *Comparative Evaluation of Texture Analysis Algorithms for Defect Inspection of Textile Products.* Brisbane, International Conference on Pattern Recognition.

Pietikaeinen, M., Ojala, T., Nisula, J. & Heikkinen, J., 1994. Experiments with Two Industrial Problems Using Texture Classification Based on Feature Distributions. *3D Vision, Product Inspection, and Active Vision,* 2354(13), pp. 197-204.

Prasetyo-Wibowo, E., Khalid, M., Yusof, R. & Meriaudeau, F., 2010. *A Comparative Study of Feature Extraction Methods for Wood Texture Classification.* Kuala Lumpur, Signal-Image Technology and Internet-Based Systems.

Rosenblatt, F., 1962. *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms.* Washington: Spartan Books.

Sadr-Kazemi, N. & Cilliers, J. J., 1997. An Image Processing Algorithm For Measurement of Flotation Froth Bubble Size and Shape Distrbutions. *Minerals Engineering,* 10(10), pp. 1075-1083.

Schmidhuber, J., 2015. Deep Learning in Neural Networks: An Overview. *Neural Networks,* Volume 61, pp. 85-117.

Shean, B. J. & Cilliers, J. J., 2011. A Review of Froth Flotation Control. *International Journal of Mineral Processing,* 100(3-4), pp. 57-71.

Siegelmann, H. T. & Sontag, E. D., 1991. Turing Computability with Neural Nets. *Applied Mathematics Letters,* 4(6), pp. 77-80.

Sonka, M., Hlavac, V. & Boyle, R., 2015. *Image Processing, Analysis, and Machine Vision.* 4th ed. Stamford: Cengage Learning.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research,* 15(1), pp. 1929-1958.

Stanford, R. L., Meredith, D. L. & Spears, D. R., 1992. *Computer Vision Applications in MIneral Processing Research.* Houston, IEEE Industry Applications Society Annual Meeting.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A., 2015. *Going Deeper with Convolutions.* Boston, Computer Vision and Pattern Recognition.

Tan, X. & Triggs, B., 2010. Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions. *IEEE Transactions on Image Processing,* 19(6), pp. 1635-1650.

Tian, D. P., 2013. A Review on Image Feature Extraction and Representation Techniques. *International Journal fo Multimedia and Ubiquitous Engineering,* 8(4), pp. 385-396.

Tukey, J. W., 1949. Comparing Individual Means in the Analysis of Variance. *Biometrics,* 5(2), pp. 99-114.

Turan, N. G., Mesci, B. & Ozgonenel, O., 2011. The use of Artifical Neural Networks (ANN) for Modelling of Adsorption of Cu(II) from Industrial Leachate by Pumic. *Chemical Engineering Journal,* 171(3), pp. 1091-1097.

Van Der Maaten, L., Postma, E. & Van Den Herik, J., 2009. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research,* Volume 10, pp. 66-71.

Vathavooran, A., Batchelor, A., Miles, N. J. & Kingman, S. W., 2006. Applying Froth Imaging Techniques to Asses Fine Coal Dewatering Behaviour. *Coal Preparation,* 26(2), pp. 103-121.

Wang, J., Han, S., Shen, N. & Li, S., 2014. Features Extraction of Flotation Froth Images and BP Neural Network Soft-Sensor Model of Concentrate Grade Optimized by Shuffled Cuckoo Searching Algorithm. *The Scientific World Journal,* 2014(11).

Weston, J., Ratle, F., Mobahi, H. & Collobert, R., 2012. Deep Learning via Semi-Supervised Embedding. In: G. Montavon, G. B. Orr & K. Müller, eds. *Neural Networks: Tricks of the Trade.* Berlin: Springer, pp. 639-655.

Wills, B. A. & Napier-Munn, T., 2005. Froth Flotation. In: *Wills' Mineral Processing Technology.* 7th ed. Oxford: Butterworth-Heinemann, pp. 267-352.

Xie, J., Xu, L. & Chen, E., 2012. Image Denoising and Inpainting with Deep Neural Networks. In: F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds. *Advances in Neural Information Processing Systems 25.* s.l.:Curran Associates, pp. 341-349.

Xie, X., 2008. A Review of Recent Advances in Surface Defect Detection Using Texture Analysis Techniques. *Electronic Letters on Computer Vision and Image Analysis,* 7(3), pp. 1-22.

Yu, H. & MacGregor, J. F., 2004. *Multivariate Image Analysis for Inferential Sensing: A Framework.* Cambridge, 7th International Symposium on Dynamics and Control of Process Systems.

Zanin, M., Wightman, E., Grano, S. R. & Franzidis, J.-P., 2009. Quantifying Contributions to Froth Stability in Porphyry Copper Plants. *International Journal of Mineral Processing,* Volume 91, pp. 19-27.

Appendix A: Example of Single Image Processing in CNN

A single image has been selected from the dataset utilised in this work to visually and mathematically demonstrate the transforms which occur within one of the convolutional neural networks developed and used in this work. Specifically, an image from the 64x64 dataset was fed through this network:

- 64in-32c(b)11-32c(b)11-32c(b)11-512fc(d)-16b(d)-4c-64batches_2017-04-13_04-03-00.tflearn9562

The meaning of this network identified is summarised in Figure 7-8, but the relevant technical parameters will be described in the following breakdown of processing in the network. While this network contains dropout and batch normalisation layers, these are only active during training, not prediction of class. As such they will be omitted from the sample calculations.

All images presented in this appendix have been normalised across the entire filter set, i.e. all images from the first convolutional layer have been normalised for the range of output values for all the images from that layer. This shows the relative activation strength of one filter compared to another in the same layer.

A.1 Input Image to First Convolution

The first step in the neural network takes the input image and splits it into its constituent colour channels. The image's data structure is a three-dimensional matrix with dimensions (64, 64, 3) where the third dimension is the colour channels. By slicing this matrix along the colour dimension, three resulting (64, 64) sized matrices are produced for each colour. The individual colour channels shown in Figure A-1 show the intensity of each colour in each pixel as monochrome images.

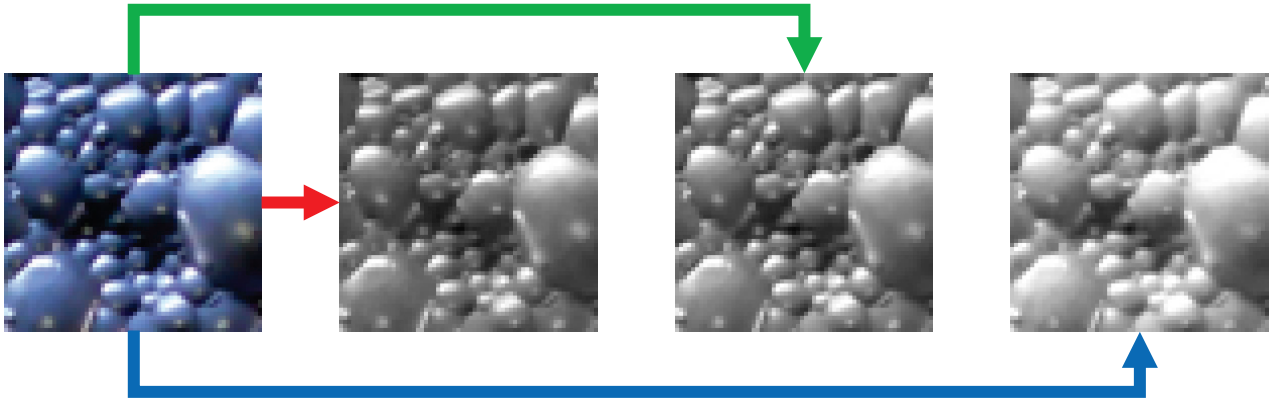
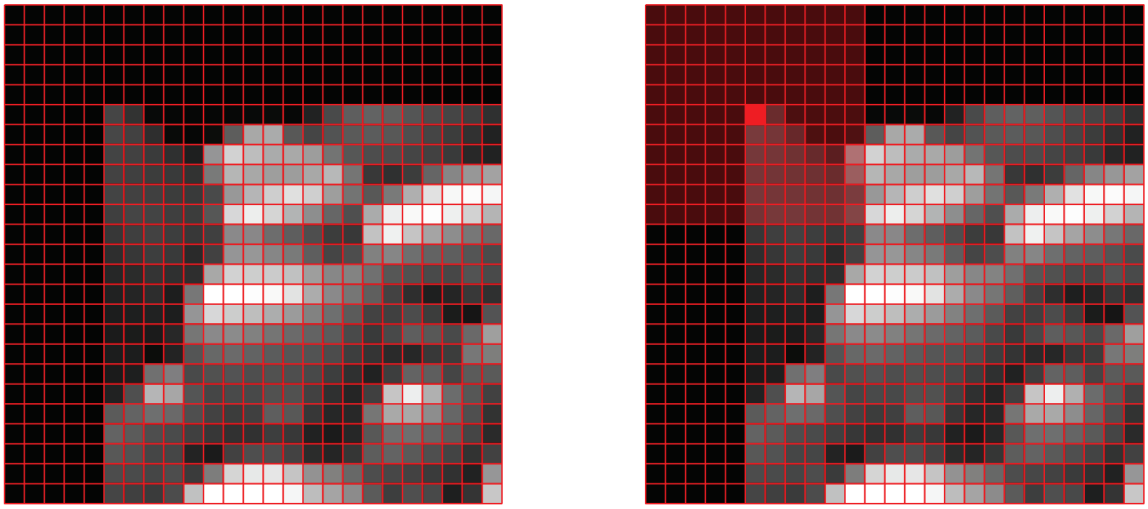


Figure A-1: Colour Channels Extracted from Input Image (Original, Red, Green, Blue)

In the convolutional layer, the input data passes through four sequential processing steps:

- Convolution of the image with a convolution kernel (called filter in this work)
- Addition of bias term to convolution output
- Activation function is applied (ReLU)
- Max-pooling of pixels

In each convolutional layer for the selected network there are 32 convolutional filters which have a size of (11, 11). In reality, these filters also have a depth which corresponds to the number of channels in the input image, so the convolution kernel has size (11, 11, 3) for the first convolutional layer as there are three colour channels in the input image. Convolution applied in this work uses padding on the input images to ensure the output size of the convolutional stage is equal to that of the input size. For an (11, 11) sized convolutional filter five pixels of value zero are padded along all sides of the input image. The upper-left 25x25 pixel region of the blue colour channel is shown in Figure A-2 after padding is applied for a total number of 30x30 pixels. The red grid shows the position of individual pixels in the image. The shaded area shows the coverage of the convolutional filter (11, 11) when convolving the first pixel (highlighted in red) in the image.



i) Padding added to image edges *ii) Coverage of convolutional filter*

Figure A-2: Upper Left Corner of Blue Channel Showing Padding and Filter Size

The actual numeric values for the pixels in the top left (11, 11) corner which are the first to be convolved are given in Figure A-4, it can be seen that the padded pixels in grey all have a value of zero, while the image pixels have values in the range from 0 to 255.

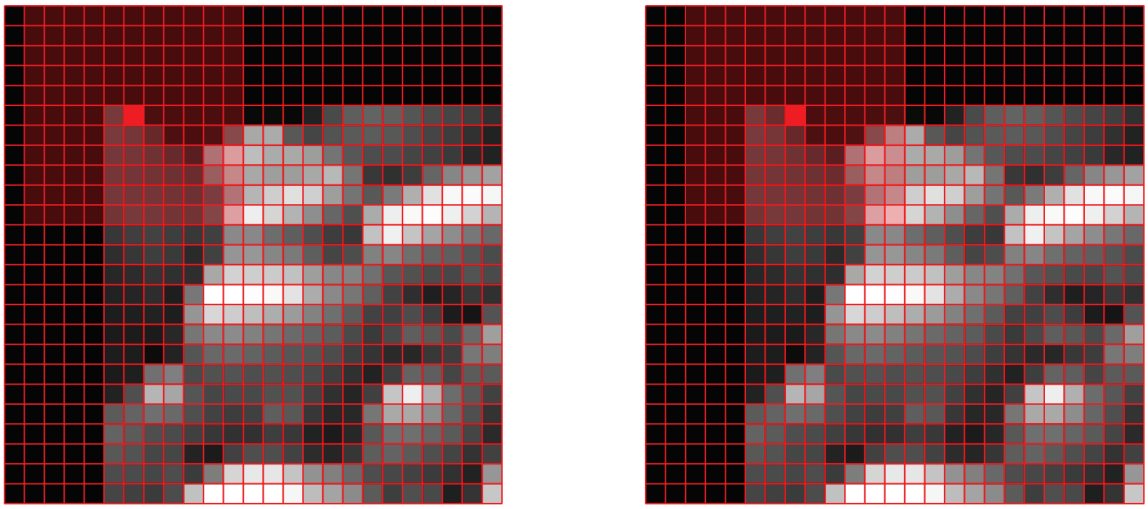


Figure A-3: Subsequent Pixel Convolution Order

Convolution with the same weights occurs across the entire image as demonstrated in Figure A-3. The convolutional window moves along the pixels of the image until all pixels have been convolved, giving an output image the same as the original (unpadded) input resolution of (64, 64).

Figure A-5, shows the value of the convolutional filter weights (for the blue channel) for the fifth filter in the first convolutional layer. In the first step of convolution, the pixels are multiplied element-wise with those in the filter, i.e. each weight in the filter corresponds with a specific pixel in the area convolved.

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	80	66	35	14	14	17
0	0	0	0	0	82	77	63	41	30	44
0	0	0	0	0	79	77	72	64	54	152
0	0	0	0	0	78	76	73	71	66	126
0	0	0	0	0	79	77	76	73	71	84
0	0	0	0	0	75	80	79	75	73	95

Figure A-4: Actual Pixel Values of Upper Left (25, 25) Corner of Padded Blue Channel

The output of this convolution on the first blue pixel in the image gives the output shown in Figure A-6. As expected from multiplication of zeros, the padded pixels give an output of zero when convolved. This results in them having a negligible impact on the final output aside from maintaining the input resolution.

0.031	-0.091	0.012	-0.015	-0.083	-0.069	-0.066	-0.115	-0.078	-0.049	0.063
-0.107	-0.145	-0.107	-0.068	-0.112	-0.139	-0.064	-0.072	-0.018	-0.090	-0.036
-0.121	-0.165	-0.115	-0.065	-0.043	0.032	0.021	-0.029	-0.117	-0.054	-0.045
-0.067	-0.173	-0.066	0.020	-0.048	-0.081	0.044	-0.062	-0.083	-0.015	0.022
-0.001	-0.067	-0.130	-0.055	-0.088	0.015	-0.036	-0.090	0.021	0.023	-0.063
-0.008	-0.132	-0.129	-0.049	-0.117	-0.035	-0.147	-0.022	-0.090	0.032	-0.073
-0.139	-0.097	-0.098	-0.059	-0.060	-0.047	-0.095	-0.100	-0.061	-0.084	-0.092
0.008	0.035	-0.044	0.020	-0.055	-0.013	-0.127	-0.003	-0.018	-0.054	-0.021
-0.099	0.058	0.067	-0.031	-0.041	-0.085	-0.025	-0.028	-0.111	0.056	-0.069
0.012	0.092	-0.018	0.066	0.033	-0.026	-0.046	-0.008	-0.047	-0.048	0.036
-0.035	0.002	-0.071	0.034	0.007	-0.042	-0.023	-0.095	-0.063	0.015	-0.038

Figure A-5: Values of Convolutional Filter Weights for Blue Channel from Filter 5

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	-2.824	-9.692	-0.777	-1.262	0.442	-1.238
0.000	0.000	0.000	0.000	0.000	-3.839	-7.305	-6.324	-2.483	-2.515	-4.043
0.000	0.000	0.000	0.000	0.000	-1.008	-9.773	-0.235	-1.165	-2.889	-3.195
0.000	0.000	0.000	0.000	0.000	-6.618	-1.879	-2.041	-7.857	3.668	-8.751
0.000	0.000	0.000	0.000	0.000	-2.040	-3.522	-0.584	-3.467	-3.380	3.021
0.000	0.000	0.000	0.000	0.000	-3.168	-1.840	-7.531	-4.693	1.078	-3.566

Figure A-6: Result of Element-Wise Multiplication of Pixels from Upper Left Blue Channel with Filter 5

All elements in the resulting convolution are then summed to give an overall convolutional output of -113.29, as the padded pixels which were convolved are zero, this does not contribute to the overall output. In matrix operations this is also known as the Frobenius inner product and is summarised in Equation A-1:

$$\sum_{i=1}^{11}\sum_{j=1}^{11} input_{i,j,blue} \times filter_{i,j,blue,5} = -113.29 \qquad (A-1)$$

The value for the next two convolved blue pixels as shown in Figure A-3 are -136.07 and -148.84 respectively. However, the red and green channels have yet to be considered. Once they are calculated, the values from each channel are summed for a final pixel output, giving an output of 318.14 for the first pixel in the image. This can be expressed as the following equation, where k represents the index of the three colour channels:

$$\sum_{i=1}^{11}\sum_{j=1}^{11}\sum_{k=1}^{3} input_{i,j,k} \times filter_{i,j,k,5} = 318.14 \qquad (A-2)$$

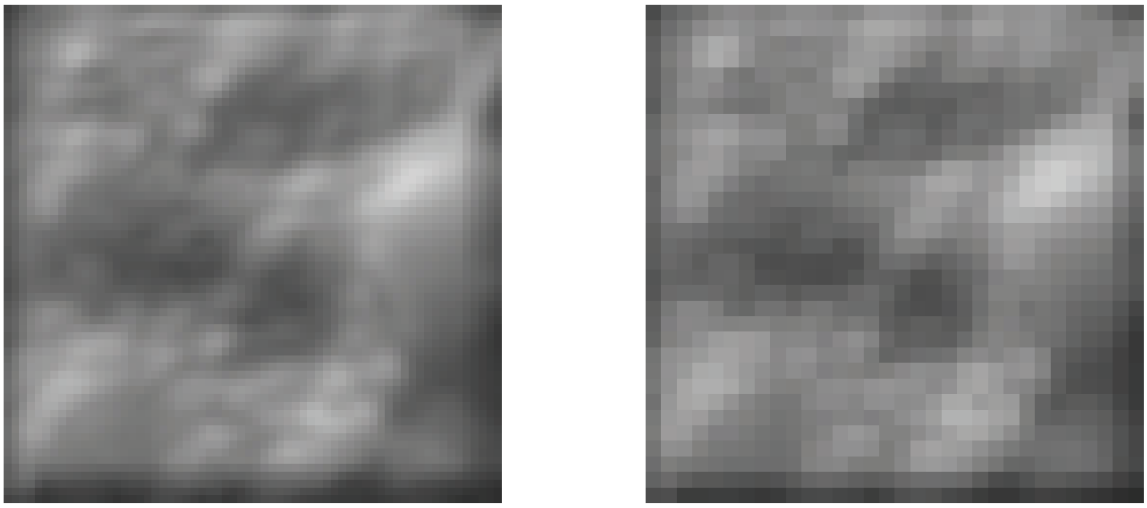
The bias term for artificial neurons as described in Section 5.1 of this work also applies to the convolutional layer. In this case, each filter has a bias term which is applied uniformly across all pixels. For the fifth filter this is 0.0226, giving a final pixel output of 318.16, this can be expressed as:

$$b_5 + \sum_{i=1}^{11}\sum_{j=1}^{11}\sum_{k=1}^{3} input_{i,j,k} \times filter_{i,j,k,5} = 318.16 \qquad (A-3)$$

It should be noted that this has a small impact for this specific filter and pixel combination, but the effect varies between pixels and filters. It is at this point that the first activation function is used, ReLU. The function is described in detail in Section 5.1.1, but to reiterate and simplify, all negative outputs are discarded and set to zero. The process can be expressed as Equation A-4 across all pixels:

$$relu\ output_{i,j} = \max(0, input_{i,j}) \qquad (A-4)$$

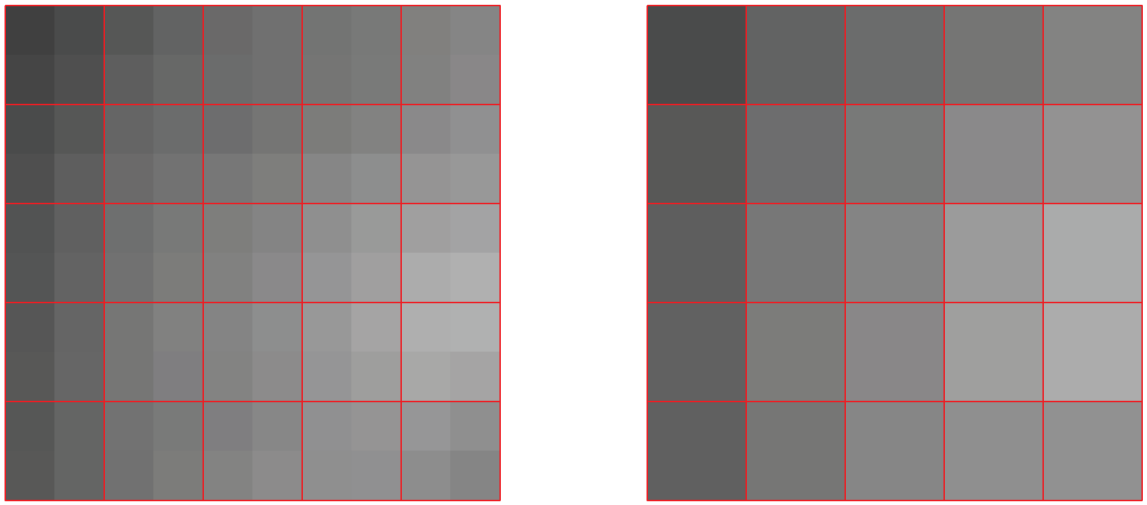
For the specific image and filter combination used this has no effect as none of the convolved pixels have a negative value. The next and final stage of convolutional layer processing is max-pooling, the effect of which is demonstrated in Figure A-7. The figures have been normalised to the standard greyscale interval to highlight the textural information in the image, relative pixel intensities are still maintained.



i) Output after ReLU activation *ii) Final output after max-pooling*

Figure A-7: ReLU and Max-Pooling Outputs of Fifth Convolutional Filter in First Layer (Enhanced)

Max-pooling examines a small region of pixels (in this case 2x2) and selects the maximum intensity pixel as the output for the region being examined. This is repeated across the entire image, reducing the number of pixels by a factor of 4 and increasing the non-linearity of the data.



i) Input image showing outline of pooled pixels (10, 10) *ii) Final output (5, 5)*

Figure A-8: Effect of Max-Pooling on (10, 10) Region of Input Image and Resulting Output

The effect of max-pooling on a smaller region of the input is shown in Figure A-8 where a (10, 10) corner from the top left of the output from ReLU activation has been taken. The brightest of the four pixels within each red bounded region is selected and output in subfigure (ii), giving an output size of (5, 5).

In all, the single filter has taken an input of size (64, 64, 3) and produced an output of (32, 32). With all 32 filters the output of the first convolutional layer is (32, 32, 32). The overall process is summarised in Figure A-9 below:

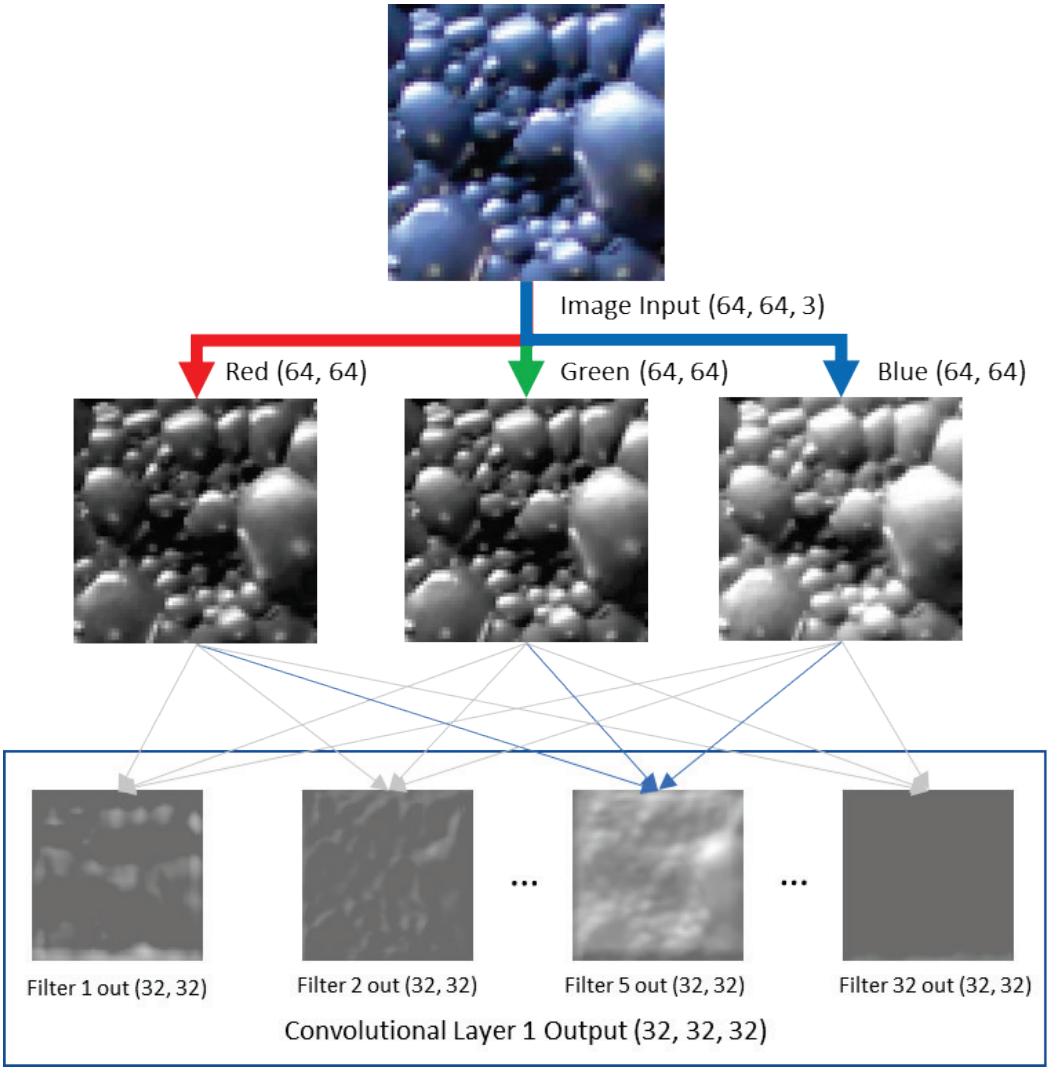


Figure A-9: Overall Process of Input Image to First Convolutional Layer

A.2 Convolutional Layer to Convolutional Layer

The second convolutional layer in the network functions identically to the first layer, though instead of three input images (for the original colour channels), each filter accepts 32 input images (for each filter in the previous convolutional layer). Thus, the overall convolutional equation between convolutional layers can be expressed as follows for arbitrary filter f (assuming 32 filters in the previous layer):

$$relu\left(b_f + \sum_{i=1}^{11}\sum_{j=1}^{11}\sum_{k=1}^{32} input_{i,j,k} \times filter_{i,j,k,f}\right) = output_{i,j,f} \qquad (A-5)$$

As with the first convolutional filter, the second layer reduces the number of pixels in each filter output by a factor of four compared to the input, taking (32, 32, 32) input and transforming it to (16, 16, 32) for the network used in this example.

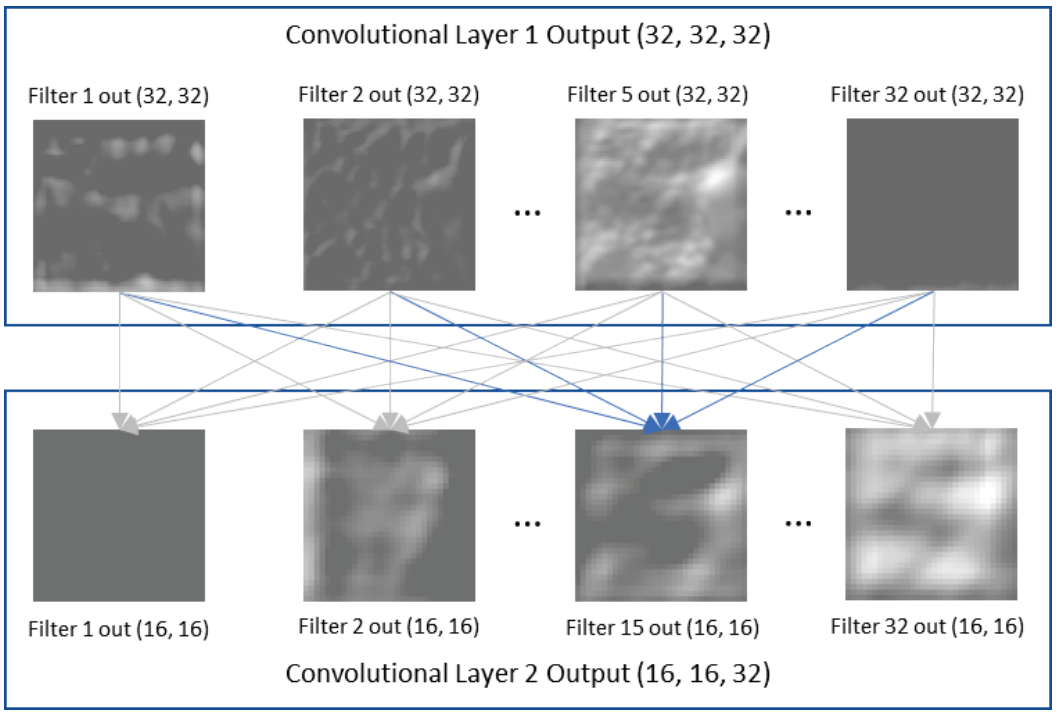


Figure A-10: First Convolutional Layer to Second Convolutional Layer Data Flow

Figure A-10 shows a simplified representation of the data flow between the first and second convolutional layers. It should be noted that the omitted filters in the first layer still contribute to each filter in the second layer as indicated by Equation A-5.

The operation between the second and final convolutional layer is again virtually identical to that between the first and second layer. The only difference being that the input size is (16, 16, 32) and the output size is (8, 8, 32). Otherwise, the exact same algorithmic procedure as described in Section A.1 is followed for all subsequent convolutional layers.

The output of the final convolutional layer in the neural network is the last point at which it can be meaningfully expressed as an image. In all subsequent fully-connected layers the outputs are single dimensional vectors, unlike the two-dimensional matrices of the convolutional layers.

A.3 Final Convolution to Fully-Connected

Unlike the convolutional layers, the transforms between fully connected layers can be more efficiently expressed as matrix multiplications. If there are n neurons in a fully connected layer with i inputs, then the weights will be a matrix of shape (i, n). The input vector of shape $(1, i)$ is multiplied with the weight matrix (i, n) to produce each neurons output $(1, n)$.

However, the output from the last convolutional layer is a two-dimensional matrix and not a vector. Thus, the first step in processing is to unwrap, or flatten, the output of the final convolutional layer. The output of the last convolutional layer has a shape of (8, 8, 32), giving 2048 individual elements. They are remapped to a vector of shape (1, 2048) by sequentially reallocating the original matrix elements. i.e. the pixels of the first filter (8, 8) are remapped to positions (1, 1) through (1, 64), the second filter to positions (1, 65) through (1, 128) etc.

$$(input_1 \quad \cdots \quad input_{2048}) \begin{pmatrix} weight_{1,1} & \cdots & weight_{1,512} \\ \vdots & \ddots & \vdots \\ weight_{2048,1} & \cdots & weight_{2048,512} \end{pmatrix} = (output_1 \quad \cdots \quad output_{512}) \quad (A-6)$$

Once flattened, weights are applied to the input vector as summarised in Equation A-6. A bias term of shape (1, 512) is added to the output term before the hyperbolic tangent activation function is applied as shown in Equation A-7.

$$\tanh\left(I_{1,2048} \cdot W_{2048,512} + b_{1,512}\right) = O_{1,512} \quad (A-7)$$

This output vector is then fed to subsequent layers. Due to the large number of vectors in this output, it will not be shown in this section.

A.4 Fully-Connected Layer to Fully-Connected Layer

Between fully-connected layers the process followed is identical to that summarised in Equations A-6 and A-7. As the output of a fully-connected layer is already one-dimensional, there is no requirement to flatten the input before processing as with convolutional layer outputs.

The intermediate fully-connected layer in this network only has 16 neurons compared to 512 of the previous neighbour, slightly modifying Equation A-7 into Equation A-8:

$$\tanh\left(I_{1,512} \cdot W_{512,16} + b_{1,16}\right) = O_{1,16} \quad (A-8)$$

The sixteen outputs generated are shown in Figure A-11, this is the feature set used in model comparisons:

-0.679	1.000	-1.000	-1.000	-0.999	-0.997	1.000	-1.000	0.546	-1.000	0.995	0.981	-1.000	-0.860	0.585	1.000

Figure A-11: Intermediate Fully-Connected Layer Output (1, 16)

A.5 Fully-Connected to Classification

From the intermediate fully-connected layer to the final classification layer again the same mathematical operations are applied as used in Equation A-6. However, unlike the first and second fully-connected layers, hyperbolic tangent is not the activation function, instead softmax activation is used giving Equation A-9:

$$\text{softmax}\left(I_{1,16} \cdot W_{16,4} + b_{1,4}\right) = O_{1,4} \quad (A-9)$$

Softmax activation, as described in Section 5.1.1, is unique in that it is interdependent on all of the output neurons. Each neuron in the output represents a single class and outputs a probability of that class being the correct one for the input image, the outputs of all neurons in the classifier layer sum to 1.

The limited number of neurons allows the information flow to be visualised for this final layer in Figure A-12:

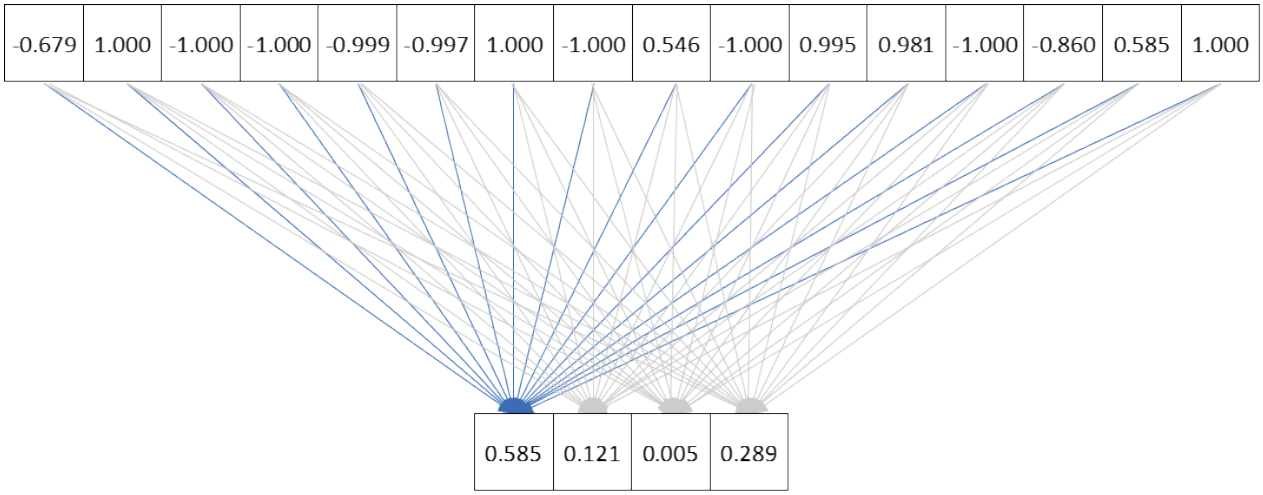


Figure A-12: Information Flow to Classification Layer Highlighting Class One as Highest Probability

Each probability consists of the weighted inputs of all previous layer neurons with a requisite bias value added followed by the softmax activation. The weights used in the final layer are shown in Figure A-13 below:

-0.462	0.243	0.193	0.208
0.291	0.591	-0.391	-0.349
-0.349	-0.571	0.392	0.366
-0.327	0.382	0.385	-0.279
-0.329	0.370	0.373	-0.296
0.221	0.198	0.136	-0.449
-0.477	0.325	0.040	0.270
-0.354	0.384	0.396	-0.283
0.480	-0.214	-0.184	-0.199
-0.331	-0.608	0.313	0.282
-0.221	-0.198	-0.128	0.450
-0.480	0.229	0.181	0.190
-0.328	-0.542	0.416	0.345
0.467	-0.227	-0.204	-0.202
0.470	-0.230	-0.179	-0.200
0.203	-0.276	-0.215	0.427

Figure A-13: Weights for Inputs to Classification Layer

A.6 Summary

In review, the neural network for the 64x64 input consisted of three convolutional layers, each with 32 filters having a size of (11, 11). After processing through the convolutional layers, an output of (8, 8, 32) is fed into three successive fully-connected layers, with the third being a classifier layer.

The size of the inputs, outputs, and number of trained parameters in each layer are as follows:

1. Convolution 1 – Input = 12288 (64, 64, 3), Output = 32768 (32, 32, 32), Parameters = 11648
2. Convolution 2 – Input = 32768 (32, 32, 32), Output = 8192 (16, 16, 32), Parameters = 123936
3. Convolution 3 – Input = 8192 (16, 16, 32), Output = 2048 (8, 8, 32), Parameters = 123936
4. Fully-Connected 1 – Input = 2048 (1, 2048), Output = 512 (1, 512), Parameters = 1049088
5. Fully-Connected 2 – Input = 512 (1, 512), Output = 16 (1, 16), Parameters = 8208
6. Classifier – Input = 16 (1, 16), Output = 4 (1, 4), Parameters = 68

From this summary it can be seen that there are a total of 1316884 parameters to be trained in the neural network, of which 79.7% are in the first fully-connected layer. This demonstrates the curse of dimensionality inherent in fully connected networks. The procedures described in this section can be repeated for any input image and combinations of convolutional and fully connected layers.

It should be noted that the processes for training a deep neural network such as used in this work are too complex and consist of too many parameters to effectively be described in this appendix. Rather it is suggested to refer to Chapter 5 and the referenced sources for mechanistic knowledge of the training process.