Unsupervised Feature Learning for Speech Using Correspondence and Siamese Networks

Petri-Johan Last

Report submitted in partial fulfilment of the requirements of the module
Project (E) 874 for the degree Master's in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Dr H. Kamper and Prof. H.A. Engelbrecht

March 2020

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

Eerstens wil ek dankie sê aan my ma. Sonder mamma se opofferinge en motivering sou nie ek of Jolanie wees waar ons vandag is nie. Aan my studieleiers, Herman en Herman, baie dankie vir die (amper) weeklikse terugvoersessies. Aan Prof. Engelbrecht, dankie vir die laaste paar jaar se studieleiding. Ons het deur 'n hele paar meestersonderwerpe gegaan, maar ek is bly hierdie laaste een het darem gewerk. Aan Dr Kamper, dankie dat jy my onder jou vlerk ingeneem het hierdie laaste jaar, anders het ek en Prof. Engelbrecht heel moontlik nog 'n onderwerp sit en uit dink vir volgende jaar. Aan G-J van Custos, dankie vir speler nommer drie wees op die studieleidingspan en die ondersteuning deur die jaar. Dan aan Lize-Marié, my verloofde, my alles. Dankie dat ek by jou kon kla oor my studieleiers. Dankie dat jy saam met my opgewonde was wanneer dinge gewerk het en my ondersteun het wanneer dinge nie gewerk het nie.

Stellenbosch University https://scholar.sun.ac.za

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / Plagiarism *Declaration*

1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.

2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.

3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.

4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Copyright © 2020 Stellenbosch University
All rights reserved

P Last **Voorletters en van / *Initials and surname***	March 2020 **Datum / *Date***

Stellenbosch University https://scholar.sun.ac.za

Abstract

English

In automatic speech recognition systems, speaker characteristics, such as gender, pitch, and talking speed, can affect the performance of the system. Humans, however, are able to understand what is being said, regardless of these speaker characteristics. There are therefore features in speech that make up word identities, regardless of the speaker characteristics. Being able to learn such features from speech would be beneficial to downstream speech processing tasks.

In this thesis, we perform three experiments. Throughout all three experiments, networks are trained on unlabelled speech data so that their applicability in a zero resource environment can be evaluated. Terms are automatically discovered using an unsupervised term discovery system, and the training procedure as a whole is unsupervised. The networks learn frame-level acoustic features, which are then evaluated using a word discrimination task that also measures speaker independence.

In the first experiment, we perform a comparison between the correspondence autoencoder (CAE) and Triamese networks. We show that, under the described training conditions, features produced by the CAE outperform those produced by the Triamese network.

In the second experiment, we investigate the effect speaker conditioning has on features produced by the CAE. A speaker matrix is constructed with randomised speaker representations for each speaker in the training set. By using access to the speaker labels, a speaker embedding is extracted from this matrix and concatenated to the input of the decoder half of the network. These embeddings are fully trainable, which gives the network more parameters to manipulate and, in theory, make the encoder half of the network less prone to keep speaker-specific information. We show that speaker conditioning produces mixed results, as it worsens performance on one dataset while increasing performance on another.

In the final experiment, we develop a novel CAE-Triamese hybrid network, the CTriamese network. By applying the contrastive loss of the Triamese network to the middle layer of the CAE, the intermediate representations of the CAE face an extra constraint. We show that this network produces features that outperform features produced by both the CAE and Triamese networks on the evaluation task. We also show that, unlike the CAE, the CTriamese network produces features that score higher on the evaluation task when speaker conditioning is introduced.

Stellenbosch University https://scholar.sun.ac.za

Afrikaans

In outomatiese spraakherkenningstelsels kan sprekerkenmerke, soos geslag, toonhoogte en geselssnelheid, die werking van die stelsel beïnvloed. Mense kan egter verstaan wat gesê word, ongeag hierdie sprekerkenmerke. Daar is dus kenmerke in spraak wat woordidentiteite beïnvloed, wat onafhanklik van die sprekerkenmerke is. Om sulke eienskappe uit spraak te leer kan voordelig wees vir spraakverwerkingstake.

In hierdie tesis voer ons drie eksperimente uit. In al drie eksperimente word netwerke opgelei op ongemerkde spraakdata, sodat die toepaslikheid daarvan in 'n nulhulpbronomgewing beoordeel kan word. Terme word outomaties ontdek deur 'n termontdekkingstelsel sonder toesig, en die opleidingsprosedure as geheel het geen toesig nie. Die netwerke leer akoestiese kenmerke op raamvlak, wat dan geëvalueer word met behulp van 'n woorddiskriminasie-taak wat ook die sprekersonafhanklikheid meet.

In die eerste eksperiment voer ons 'n vergelyking uit tussen die korrespondensie outoenkodeerder (KOE) en *Triamese* netwerke. Ons toon aan dat kenmerke wat deur die KOE vervaardig is, beter vaar in die evalueringstaak as die wat deur die Triamese netwerk vervaardig is, onder die bogenoemde opleidingsomstandighede.

In die tweede eksperiment ondersoek ons die effek wat sprekerkondisionering het op die kenmerke wat deur die KOE vervaardig word. 'n Sprekermatriks word saamgestel met lukrake sprekervoorstellings vir elke spreker in die opleidingstel. Deur toegang tot die spreker identiteite te gebruik, word 'n sprekervoorstelling uit hierdie matriks onttrek en gekoppel aan die ingang van die dekodeerderhelfte van die KOE. Hierdie voorstellings kan deur die netwerk aangepas word, wat die netwerk meer parameters gee om te manipuleer en, in teorie, die kodeerderhelfte van die netwerk minder geneig maak om sprekerspesifieke inligting te hou. Ons toon aan dat sprekerskondisionering gemengde resultate lewer, aangesien dit die prestasie op een datastel vererger en die prestasie op 'n ander verhoog.

In die laaste eksperiment ontwikkel ons 'n nuwe KOE-Triamese basternetwerk, die KTriamese netwerk. Deur die kontrasverlies van die Triamese netwerk op die middelste laag van die KOE toe te pas, het die intermediêre voorstellings van die KOE 'n ekstra beperking. Ons toon aan dat hierdie netwerk kenmerke lewer wat beter vaar in die evalueringstaak as kenmerke wat deur die KOE en Triamese netwerke vervaardig word. Ons toon ook aan dat die KTriamese-netwerk, anders as die KOE, kenmerke produseer wat beter vaar in die evalueringstaak wanneer sprekerkondisionering toegepas word.

Stellenbosch University https://scholar.sun.ac.za

Contents

Abstract iii

List of Figures viii

List of Tables x

Nomenclature xi

1. Introduction 2

1.1. Motivation . . . 3
1.2. Background . . . 4
1.3. Literature synopsis . . . 4
1.3.1. Autoencoders . . . 4
1.3.2. The correspondence auto-encoder . . . 5
1.3.3. The Siamese network . . . 5
1.3.4. The Triamese network . . . 5
1.4. Objectives . . . 5
1.4.1. Comparing the CAE and Triamese networks . . . 6
1.4.2. Improving the CAE with speaker conditioning . . . 6
1.4.3. Build a CAE-Triamese hybrid network . . . 6
1.5. Contributions . . . 6
1.6. Overview of this work . . . 7

2. Literature review 9

2.1. Zero resource environments . . . 9
2.2. Mel-frequency cepstral coefficients . . . 10
2.3. Dynamic time warping . . . 10
2.3.1. Segmental dynamic time warping . . . 11
2.4. Unsupervised term discovery . . . 11
2.5. Evaluation . . . 12
2.5.1. Precision and recall . . . 12
2.5.2. The same-different task . . . 14
2.5.3. The ABX task . . . 15
2.6. Neural networks . . . 16
2.6.1. Optimisation . . . 16

Stellenbosch University https://scholar.sun.ac.za

2.7. Activation functions . . . 19
2.7.1. Linear function . . . 19
2.7.2. Rectified linear unit . . . 20
2.8. The Siamese network . . . 21
2.9. The Triamese network . . . 23
2.10. The correspondence autoencoder . . . 24
2.11. Chapter summary . . . 24

3. Datasets 26
3.1. The Buckeye corpus of conversational English . . . 26
3.1.1. Training, validation and test sets . . . 26
3.2. The NCHLT speech corpus of South African languages . . . 27
3.2.1. Training and test sets . . . 27
3.3. Data processing . . . 28
3.3.1. Extract MFCCs with CMVN, deltas and delta-deltas . . . 28
3.3.2. Unsupervised term discovery (UTD) . . . 28
3.3.3. Voice activity detection . . . 29
3.3.4. Word pair alignment . . . 29
3.4. Chapter summary . . . 29

4. A comparison of the correspondence autoencoder and the Triamese network 30
4.1. The networks . . . 30
4.1.1. The correspondence autoencoder (CAE) . . . 30
4.1.2. The Triamese network . . . 31
4.1.3. Network comparison . . . 32
4.2. Experiments . . . 33
4.2.1. Purpose of experiment . . . 33
4.2.2. Datasets . . . 33
4.2.3. Unsupervised term discovery . . . 34
4.2.4. Evaluation metric . . . 34
4.2.5. Experimental setup . . . 35
4.2.6. Results . . . 36
4.3. Chapter summary and conclusions . . . 37

5. Introducing speaker information into the CAE 39
5.1. Hypothesis . . . 39
5.1.1. Adding speaker information . . . 39
5.2. Experiments . . . 40
5.2.1. Experimental setup . . . 40
5.2.2. Results . . . 41

Stellenbosch University https://scholar.sun.ac.za

5.3. Chapter summary and conclusions 42

6. The Correspondence Triamese Network 43

6.1. CAE and Triamese 43
6.2. Hypothesis 44
6.3. The correspondence Triamese network (CTriamese) 44
6.4. Speaker information 45
6.5. Experiments 46
6.5.1. Experimental setup 46
6.5.2. Training procedure 47
6.5.3. Results 47
6.6. Chapter summary and conclusions 49

7. Summary and conclusion 50

7.1. Comparison of the CAE and Triamese networks 50
7.2. Speaker conditioning in the CAE 51
7.3. The CTriamese network 51
7.4. Contributions 51
7.5. Future work 52

Bibliography 53

A. Network implementations in Python 57

A.1. CAE 57
A.2. Triamese 58
A.3. CAE with speaker conditioning 59
A.4. CTriamese with speaker conditioning 61

Stellenbosch University https://scholar.sun.ac.za

List of Figures

2.1. Example of MFCCs from two terms aligned using DTW [1]. Frames from one term that are similar to the other term are mapped to each other. . . . 11
2.2. Examples of precision-recall curves from [2]. 14
2.3. A linear activation function. 19
2.4. A ReLU activation function. 20
2.5. A leaky ReLU activation function. 21
2.6. A randomised leaky ReLU activation function. The different values of a are randomised for each unit. 22
2.7. A Siamese network. The two subnetworks are identical and share weights. An embedding is extracted from each input and compared. Depending on whether or not the embeddings are representing the same type of input, a distance function is minimised or maximised. 22
2.8. A Triamese network. The three subnetworks are identical and share weights. An embedding is extracted from each input. The loss function attempts to minimise the distance between two inputs while simultaneously maximising the distance to the third input. 23
2.9. The CAE is trained using word pairs consisting of similar words spoken by a variety of speakers [1]. The network consists of two halves: an encoder and a decoder. Feature embeddings are extracted from the middle layer. . 24

4.1. The correspondence autoencoder (CAE) is trained to reconstruct one acoustic frame in a discovered word from another [1]. 31
4.2. The Triamese network is trained so that the embeddings $\boldsymbol{e}_a$ and $\boldsymbol{e}_b$ of the same type are more similar by a margin m than embeddings $\boldsymbol{e}_a$ and $\boldsymbol{e}'$ of different types. 32

5.1. Speaker information is introduced to the CAE by manner of an $n \times m$ matrix that is trained alongside with the network. 40

6.1. A CAE next to a Triamese network. As shown by the dotted box, a single branch of a Triamese network is identical to the encoding half of a CAE. . 44
6.2. The final network architecture of the CTriamese network. Each branch of the Triamese network can be extended to a full CAE network, thus employing the constraints of both networks on the embedding layer. 45

Stellenbosch University https://scholar.sun.ac.za

6.3. The final network architecture of the CTriamese network. Each branch of the Triamese network can be extended to a full CAE network, thus employing the constraints of both networks on the embedding layer. 46

Stellenbosch University https://scholar.sun.ac.za

List of Tables

3.1. Speaker distribution in the Buckeye subsets. 27
3.2. Speaker distribution in the Xitsonga subsets. 28

4.1. Results from the evaluation task show that, when trained on UTD terms, features produced by the CAE outperforms those produced by the Triamese network. The Triamese network also failed to outperform the baseline MFCCs on the English test set, possibly due to being sensitive to the selection of training pairs. 36
4.2. Results from the evaluation task show that, when trained on ground truth terms, features produced by the CAE outperforms those produced by the Triamese network. Using an Adam optimiser instead of stochastic gradient descent worsened the performance of the Triamese network. 37

5.1. Each network is trained using discovered terms. The network is then used as a feature extractor, and the features are evaluated using the same-different task. 41
5.2. Each network is trained using ground truth terms. The network is then used as a feature extractor, and the features are evaluated using the same-different task. 41

6.1. Results from the evaluation task with each network trained on UTD terms. For both AP and PRB a higher value is better. 48
6.2. Results from the evaluation task with each network trained on ground truth terms. For both AP and PRB a higher value is better. 48

Stellenbosch University https://scholar.sun.ac.za

Nomenclature

Variables and functions

f	Frequency.
P	Precision.
R	Recall.
W	Set of words.
w	Single word.
τ	Threshold parameter.
$P(\tau)$	Precision with respect to threshold τ.
$R(\tau)$	Recall with respect to threshold τ.
$\boldsymbol{\theta}$	Parameter vector of a neural network.
θ	Specific parameter in a neural network.
G	Sum of squared gradients.
g	Gradient of a parameter.
RMS	Root Mean Squared Error.
m_t	Decaying average of past gradients.
v_t	Decaying average of past squared gradients.
β	Decay rate.
$\mathbf{e}$	Embedding Vector.
γ	Margin Paremeter.
s	Speaker.

Stellenbosch University https://scholar.sun.ac.za

Acronyms and abbreviations

AP	Average Precision
ASR	Automatic Speech Recognition
CAE	Correspondence Auto-encoder
CTriamese	Correspondence Triamese
DCT	Discrete Cosine Transform
DTW	Dynamic Time Warping
MFCC	Mel-Frequency Cepstral Coefficient
PLP	Predictive Linear Prediction
PRB	Precision-Recall Breakeven
SDTW	Segmental Dynamic Time Warping
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Unit
RReLU	Random Rectified Linear Unit
UTD	Unsupervised Term Discovery
VAD	Voice Activity Detection

Stellenbosch University https://scholar.sun.ac.za

Chapter 1

Introduction

Not all languages in the world are written [3]. In some cases, collecting labelled speech data for these languages can be nearly impossible. We refer to these languages for which resources are scarce or non-existent as *zero resource* languages. However, just because these languages are not written does not mean they are seldomly used. Various Chinese and Arabic variations have no written form but are spoken by millions of people [4]. A language does not have to be unwritten for it to be classified as a zero resource language, and being zero resource does not mean it is nearly extinct. Zero resource simply means no labelled speech resources are available. However, our most advanced methods of developing speech technologies, such as automatic speech recognition, rely on large quantities of labelled data. We refer to this method of training that requires labelled data as *supervised learning* [5]. The requirement of labelled training data presents a problem for zero resource languages and for the people who speak them. The lack of labelled training data means that large populations of the world cannot benefit from the technologies we develop. Researchers have therefore also started to develop methods of developing speech technologies without labelled data, and we refer to it as *unsupervised learning* [5]. Supervised learning methods generally far outperform their unsupervised counterparts [6], but in zero resource environments, supervised learning is not an option.

Zero resource speech processing has piqued the interest of a variety of fields. The field of cognitive science has an interest in modelling infant speech acquisition [7]. Children can acquire speech long before they learn to write, and they learn to speak by constant exposure to spoken language, other social and visual cues, and interaction in their environments. From the child's perspective, they are operating in a zero resource environment. Similarly, if a computer is tasked to learn a language in an unknown environment, it should be equipped to operate in a zero resource environment. Linguists require tools that can work with zero resource languages if they mean to study the language [8]. In the Zero Resource Speech Challenge [9], researchers around the world every year attempt to solve a specific task in the more significant problem of language acquisition for machines.

The use of neural networks has seen great success in the handling of these tasks [10], and it is these neural networks that we will be focusing on in this thesis. Specifically, we compare two neural networks that have seen success in the area of unsupervised speech processing, specifically for feature extraction. We attempt to improve upon one of these

Stellenbosch University https://scholar.sun.ac.za

models by using our access to speaker identities during training. We also present a novel hybrid network that merges these two approaches and show that the resulting features score higher on a word discrimination task that also measures speaker independence.

1.1. Motivation

Speech can sound significantly different depending on the speaker. Women tend to have higher pitch compared to men, and individuals talk at different speeds at different times. All of these characteristics are reflected in the speech data itself. The same word spoken by two different people can appear vastly different when looking at the data alone. However, humans are able to understand what is being said, regardless of the speaker. This task is not quite as simple in the field of speech processing. Given a spoken word, we want to classify this word by its type, regardless of the speaker. We have to extract information from these words that maintains the identity of the word while disregarding information such as the speaker of the word. This feature extraction task is fundamental to other downstream speech processing tasks, such as query by example search or automatic speech recognition [11–14].

The work presented in this thesis builds upon previous work conducted by Kamper et al. in the development of the correspondence autoencoder (CAE) [6]. They developed the CAE to act as a generic feature extractor. The goal is to extract features from speech data that keep meaningful linguistic contrasts that make up the word identity while disregarding speaker-specific features, like the speaker's pitch, gender, and speech tempo. The purpose of this work is to build upon their work and improve the performance of downstream speech processing tasks in zero resource environments as a whole. We investigate whether speaker conditioning during training can improve the features produced by the CAE.

While several networks are developed every year, researchers use different evaluation techniques on different datasets, making a fair comparison between networks that perform similar tasks difficult or even impossible. We compare the CAE with another promising network: the Triamese network [15]. The Triamese network, like the CAE, was also developed for speaker-independent feature extraction. However, to our knowledge, these networks have never been compared using the same evaluation technique on the same data.

Lastly, we investigate whether or not these networks have aspects that are complementary to one another. The CAE and Triamese networks propose different methods of feature extraction, and both have seen success with their methods. However, the fact that the methods are different can be beneficial, as this means that both networks are possibly doing something the other network is not. If these networks can be combined to work together in some way and produce features that outperform the features produced by the individual networks, we can show that the networks are indeed complementary.

Stellenbosch University https://scholar.sun.ac.za

1.2. Background

The following is a list of concepts that are necessary to understand the rest of this chapter. We explain each of these concepts more thoroughly in Chapter 2.

- **Mel-frequency cepstral coefficients** (MFCCs) refer to a specific set of features extracted from audio files. MFCCs have proven to be very useful in speech processing, and many researchers use it as the base input they use to train their neural networks.
- The **ABX task** is an evaluation task in which an input (X) is presented and classified as either belonging to a group A or B. From the results, an error rate is calculated which is used as a performance metric.
- The **same-different task** is an evaluation task in which a distance is calculated between all pairs in a set. Pairs which are closer than some specified threshold are classified as belonging to the same set. From the classifications, two metrics are produced: precision (how many classifications were correct) and recall (what fraction of actual correct classifications were considered). By varying the threshold, a precision-recall curve is produced. The area under this curve, called the average precision (AP), is used as a performance metric.
- **Autoencoders** refer to a specific type of neural network that attempts to replicate its input at the output. These have various applications, including the denoising of data or finding alternative representations of data (such as compression or feature extraction).
- **Unsupervised term discovery** (UTD) refers to a system in which repeated utterances are automatically detected and labelled in a large body of unlabelled speech data. A UTD system is valuable in an unsupervised setting as it labels otherwise unlabelled speech data, allowing us to make use of supervised machine learning techniques.

1.3. Literature synopsis

1.3.1. Autoencoders

Autoencoders [5] are neural networks that are trained to replicate their input at their output. Autoencoders consist of two halves: an encoder and a decoder. There is a hidden layer in the middle which connects the encoder and the decoder. By varying the size of this hidden layer and other training conditions, autoencoders can be trained to perform a variety of tasks, such as compression [16], denoising input [5] and finding alternative

Stellenbosch University https://scholar.sun.ac.za

feature representations of data [17]. Once an autoencoder is trained, the encoder and decoder halves can be split and used independently.

1.3.2. The correspondence auto-encoder

Kamper et al. [6] proposed the correspondence auto-encoder, which uses weak top-down supervision in the form of word pairs discovered by an unsupervised term discovery system. By using pairs of similar words spoken by different speakers as an input-output pair, the CAE learns feature representations in a middle bottleneck layer. Since the speakers of the word pair differ but the word identity is the same, the network disregards speaker information so that the output target can be reconstructed as closely as possible. Kamper et al. [6] showed that speech features produced by the CAE nearly match those produced by a supervised neural network.

1.3.3. The Siamese network

A Siamese network [18] consists of two identical subnetworks which share weights. The network is trained on pairs of inputs, which can be labelled positive or negative. A positive input pair represents two inputs of the same type, while a negative input pair consists of inputs of different types. Each subnetwork extracts a feature set from each input, and then, depending on the label of the pair, the loss function attempts to minimise or maximise a distance function. These networks have seen a wide variety of applications, including signature verification [18] and speech processing [15].

1.3.4. The Triamese network

Zeghidour et al. [15] proposed the Triamese network, which is a derivative of the Siamese network. The Triamese network is trained on word triplets, where two of the words are the same word spoken by different speakers, and the third word is a different word spoken by one of the other two speakers. This network compares the features extracted from the words during training and adjusts itself so that the features of the two identical words are more similar and the features of the third word are less similar. Zeghidour et al. [15] found that the Triamese network performs better than the baseline mel filterbanks spectral coefficients on an ABX discrimination task.

1.4. Objectives

The objectives of the thesis is three-fold:

- We conduct a fair comparison between the CAE and Triamese networks, where we train both networks on the same datasets and evaluate them using the same

Stellenbosch University https://scholar.sun.ac.za

same-different evaluation task.

- We want to improve upon the CAE by using speaker conditioning during training.
- We build a novel hybrid network that combines aspects of both the CAE and Triamese networks and evaluate how it performs in comparison to the other two networks on the same-different evaluation task.

1.4.1. Comparing the CAE and Triamese networks

There are multiple methods researchers use to evaluate the performance of speech features [2, 19]. There are also multiple datasets available which researchers use to train their datasets on. Our goal is to select a single evaluation method and use it to evaluate the quality of the features produced by both networks when both networks are trained using the same training data. For the datasets, we choose the Buckeye corpus of conversational English [20] and the Xitsonga language from the NCHLT dataset [21]. These are large datasets with multiple speakers, with word labels for each dataset. For the evaluation task, we use the same-different task [2], a word discrimination task that also measures speaker independence.

1.4.2. Improving the CAE with speaker conditioning

Speaker identities are much easier to obtain than word labels. It simply requires that the speech data be categorised according to speaker, something that can be done when the data is recorded. We want to use this information during training to reduce the speaker dependence of the features produced by the CAE.

1.4.3. Build a CAE-Triamese hybrid network

The CAE and Triamese networks differ in how the embedding features are constrained during training. The Triamese network uses a contrastive loss function directly on the feature embeddings, which makes embeddings of the same type more similar and embeddings of different types less similar. The CAE uses a reconstructive loss, where the feature embedding from one speaker is reconstructed into another speaker by passing it through a decoder. We want to use this difference to our advantage and produce a network that utilises both to produce features that are less speaker dependent.

1.5. Contributions

By building upon Kamper et al. [6] and Zeghidour et al.'s [15] work on the CAE and Triamese network respectively this thesis makes the following contributions to the field of

Stellenbosch University https://scholar.sun.ac.za

unsupervised speech feature extraction:

- For the first time, to the best of our knowledge, the CAE and the Triamese network are compared on the same dataset using the same evaluation technique. We show that, under the same conditions, features produced by the CAE outperforms features produced by the Triamese network.

- We introduce speaker data to the CAE by learning speaker representations during training and use these representations in the decoding half of the CAE so that the encoding half, which acts as the feature extractor, can be more speaker agnostic. However, tests showed that speaker conditioning slightly hinders the performance of the CAE instead.

- We propose a novel neural network that combines aspects of both the CAE and the Triamese networks, called the correspondence Triamese (CTriamese) network. We evaluate the features produced by the CTriamese network using the same evaluation techniques and find it outperforms the CAE and Triamese networks on all datasets used in the evaluation. We also use the speaker conditioning proposed for the CAE on the CTriamese network and find that it leads to minor improvements.

The work in this thesis has also been submitted as an article to the *IEEE Signal Processing Letters*.

1.6. Overview of this work

The rest of the thesis is structured as follows.

In Chapter 2 we look at relevant literature in the field. Concepts that are necessary to understand the rest of the thesis, like mel-frequency cepstral coefficients, are explained here. We also look at work presented by others in the field, such as variations on the CAE, the use of Siamese networks and other neural network based approaches.

Chapter 3 describes the datasets we use to train our systems. It describes the contents of the Buckeye corpus and the NCHLT Xitsonga dataset. We describe how we use an unsupervised term discovery task and MFCCs to extract pairs from a large body of speech data. By using this approach to set up our data we ensure that our process as a whole is unsupervised and allows us to use techniques that are usually supervised in the training of our models. We also explain how we organise our data into training, validation and test sets to ensure a fair evaluation environment.

Chapters 4 to 6 contain the bulk of our contribution.

In Chapter 4 we compare the performance of the CAE with that of the Triamese network on a word discrimination task that also measures speaker independence. We show that the CAE outperforms the Triamese network on all datasets considered here.

Stellenbosch University https://scholar.sun.ac.za

In Chapter 5 we describe our hypothesis of using speaker conditioning during the training of the CAE to improve the speaker independence of the features it produces. The results show that speaker conditioning hindered the performance of the features instead.

In Chapter 6 we introduce the correspondence Triamese network. This is a novel hybrid network created by combining the CAE and Triamese networks. We evaluate the CTriamese network on the same data as the other two networks and show that it manages to outperform both using the same evaluation technique.

Finally, in Chapter 7, we conclude the thesis. We give a summary of our main findings and we make some recommendations for future work.

Stellenbosch University https://scholar.sun.ac.za

Chapter 2

Literature review

We start this chapter by explaining some relevant concepts that are crucial to the rest of the thesis. We give a brief overview of the definition of zero resource environments, as the main focus of our research relates to them. Next, we describe unsupervised term discovery, which allows us to turn an otherwise unsupervised environment into a supervised environment. We also explain what dynamic time warping is and how it is used to make two inputs comparable that would otherwise not be comparable. We describe what mel-frequency cepstral coefficients are and how they are extracted. After this, we review some evaluation tasks that are often used in the evaluation of speech features in zero resource environments, namely the same-different task and the ABX task. After that, we give a brief overview of neural networks, optimisers, and often used activation functions. Finally, we give an overview of the neural networks that are relevant to this thesis, namely the Siamese network, the Triamese network, and the correspondence autoencoder.

2.1. Zero resource environments

The term “zero resource environment” [22] refers to a problem space in which annotated data is not available. In speech processing, we use this term to refer to languages for which annotated speech is scarce or impossible to come by. Very few languages in the world have proper scientific or technological coverage. This lack of coverage causes a vast technological divide between the languages that are covered and the languages that are not. There are also thousands of languages which are hardly documented at all, leaving these languages at risk of going extinct within the next few hundred years [23]. A large portion of the world’s everyday speech stems from languages that are mostly unwritten [8], making actual documentation of the languages difficult or impossible. Linguists need technologies that can help them navigate these zero resource environments if they were to stand any chance of preserving some of these languages.

Apart from linguists, cognitive scientists are interested in modelling language acquisition in humans [7]. Infants can acquire full mastery of a language with access only to raw language data (spoken by the people around them), social cues, and interaction in their environments.

Stellenbosch University https://scholar.sun.ac.za

The Zero Resource Speech Challenge (ZRSC) [9] tasks researchers around the world with a series of challenges in an attempt to answer a specific research question: how can a system autonomously acquire language? The challenge limits researchers to resources that would be available to a language learning infant. The ZRSC is split into two tracks. Track one is unsupervised subword modelling. In this track, researchers try to discover alternate representations of sounds that emphasise linguistically relevant features, while de-emphasising the irrelevant features. Track two is spoken term discovery. In this track, researchers try to automatically discover word-like phrases in a continuous stream of speech.

Progress in the field of zero resource speech research can have far-reaching implications in a variety of connected fields.

2.2. Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients [24] (MFCCs) have been one of the best generic features for speech for a long time and is generally the baseline against which new features are measured. Networks also frequently use MFCCs as their input features and then further improve upon the MFCCs by producing features from it. MFCCs are produced by windowing a signal into 20-40 ms time frames. The power spectrum of each time frame is then calculated. Using the mel scale, these power spectrums are summed together in bins, which produces mel filterbanks. These bins get wider as the frequencies increase. This step is motivated by the workings of the human ear, as it is more difficult for the human ear to distinguish between frequencies as the frequencies increase. Mel filterbanks are also sometimes used instead of MFCCs. The discrete cosine transform (DCT) of the logarithms of the filterbank energies are then computed. The use of the logarithm is also motivated by the human ear, as to increase the perceived loudness of a sound, the energy has to be increased exponentially. The mel filterbanks overlap, making the filterbank energies correlated. The DCT decorrelates these energies, and the resulting coefficients are what make up the MFCCs.

2.3. Dynamic time warping

Dynamic time warping (DTW) is a dynamic programming technique in which two linear sequences are compared and aligned by warping the time axis of one of the sequences. DTW has seen many applications in audio, video and graphics data. People tend to speak at varying speeds at different times, and the same utterance can vary depending on how fast or slow it was spoken. By applying DTW to two utterances of the same word, the DTW-aligned pair becomes comparable. The words are separated into small time windows called frames, and DTW aligns these frames so that the same segment of speech in one

Stellenbosch University https://scholar.sun.ac.za

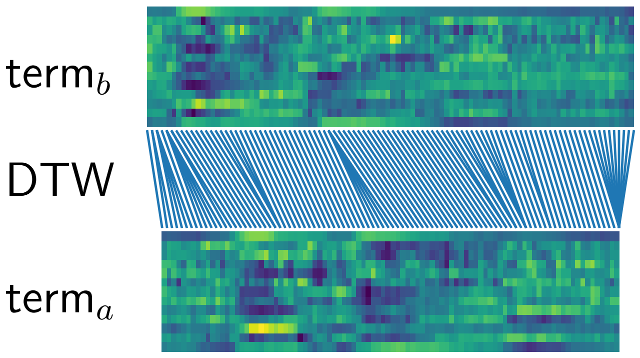

Figure 2.1: Example of MFCCs from two terms aligned using DTW [1]. Frames from one term that are similar to the other term are mapped to each other.

word aligns with the same segment in the other. Given two sequences of length n and m, and some distance function d, the DTW alignment algorithm is as follows:

1. Create an $n \times m$ matrix, G, with a value of 0 at index $(0, 0)$ and a value of infinity along the rest of the 0-indexed axes.
2. Calculate the value at index $(1, 1)$ as $d(1, 1)$ (the distance between each sequence at index 1).
3. For each other index, (i, j), the value is calculated by $G(i, j) = \min(G(i, j-1), G(i-1, j), G(i-1, j-1)) + d(i, j)$. Keep track of each index that contributed to the next (the result of $\min(G(i, j-1), G(i-1, j), G(i-1, j-1))$).
4. Starting at index (n, m), work back, selecting each index that was kept track of in step 3. This produces a minimum distance path between the two sequences, with the indexes of the path being the aligned indexes of the two sequences.

The input frames can also be MFCCs. Figure 2.1 shows the alignment of two MFCC frames from the same word type.

2.3.1. Segmental dynamic time warping

Segmental dynamic time warping (SDTW) differs from regular DTW. Where regular DTW aligns two entire sequences, SDTW compares and aligns subsections of the sequences, and instead tries to find an optimal alignment among these subsections.

2.4. Unsupervised term discovery

The training of supervised systems relies on large quantities of annotated data. Speech focused systems, like automatic speech recognition, require large datasets of annotated

Stellenbosch University https://scholar.sun.ac.za

speech. In zero resource environments, these annotated datasets could be difficult or impossible to come by. Collecting unlabelled data, on the other hand, is far easier, as all that is required is a speaker and a recording setup. We need to train unsupervised systems if we want to operate in a zero resource environment.

Unsupervised term discovery (UTD), sometimes referred to as lexical discovery or spoken term discovery, refers to the discovery of repeated words or phrases in speech without the help of any speech labels or pronunciation dictionaries. Using a UTD system, similar sections can be grouped and labelled under the same label. A UTD system allows us to use otherwise supervised techniques in an unsupervised capacity. Park and Glass [25] proposed an unsupervised pattern discovery system. They show that lexical entities can be extracted directly from unannotated speech audio. Using SDTW, they find all acoustically similar audio segments in a speech corpus, which they then cluster together. They show that these clusters correspond to relevant words and phrases in the audio from which they are extracted. However, the exhaustive SDTW search is inefficient and does not scale well for practical use. Jansen and Van Durme [26] showed that using randomised algorithms, they can increase the efficiency with which spoken terms are discovered. Kamper et al. [27] devised a full-coverage system that segments and clusters conversational speech audio. In a full-coverage system, word boundaries and lexical categories are predicted for the entire input [28]. Theirs is the first attempt at an evaluation of a full-coverage zero resource system on multi-speaker large-vocabulary data.

2.5. Evaluation

Evaluation tasks are used to measure how effectively a task was performed. In the case of speaker-independent feature extraction, we use it to measure how effectively the feature extractor managed to disregard speaker-specific information while maintaining features that make up the word identity. In this section we first look at precision and recall, two metrics used in the evaluation of retrieval tasks. After that, we look at two tasks that are often used to evaluate the performance of speech features, namely the *same-different task* and the *ABX task*.

2.5.1. Precision and recall

In the context of information retrieval, precision and recall are two measures that describe the success of an information retrieval task [29]. Given a set of items, S, and an imperfect task with the purpose of retrieving a subset of these items that match a certain criteria. Within S, there are a subset of items, S_{relevant}, that exactly match the criteria specified. However, since the retrieval task is not perfect, a subset $S_{\text{retrieved}}$ is retrieved instead. Given these subsets, we define two measures with which we measure the success of the

Stellenbosch University https://scholar.sun.ac.za

retrieval task: precision and recall.

Precision

Precision measures how accurate the retrieval task was in retrieving relevant items. Given a set of retrieved items, the precision of the task is defined to be the fraction of the retrieved items that match the fetching criteria. Precision (P) is calculated by the following equation:

$$P = \frac{|S_{\text{relevant}}| \cap |S_{\text{retrieved}}|}{|S_{\text{retrieved}}|}, \tag{2.1}$$

where $|S|$ is the number of elements in a set S.

Recall

Recall measures how successful the task was in retrieving relevant items. Given the set of all possible items, there are a number of relevant items in that set that exactly matches the retrieval criteria. Recall measures the fraction of these relevant items that are successfully retrieved. Recall (R) is calculated by the following equation:

$$R = \frac{|S_{\text{relevant}}| \cap |S_{\text{retrieved}}|}{|S_{\text{relevant}}|}. \tag{2.2}$$

Precision-recall curve

A retrieval task can have perfect precision but imperfect recall and vice versa. If exactly one item is retrieved that matches the retrieval criteria, the precision will be 1. However, if there are 100 relevant items in the set, the recall will be 0.01. Similarly, if 100 items are retrieved, but there is only 1 relevant item in the set, the recall will be 1 but the precision will be 0.01. Given a retrieval task $f(\tau)$ that can be tuned via some parameter τ, τ can be varied and the precision and recall of the task measured at each value. Plotting the precision against the recall for each value produces a precision-recall curve. An example of such a curve is illustrated in Figure 2.2.

There are two characteristics of this precision-recall curve that are of relevance: the precision-recall breakeven (PRB) and the average precision (AP). The PRB is the value at which the precision and recall are equal. In a perfect retrieval task, this value would be 1. The AP is the area under the precision-recall curve, which can be used as a performance measure of either the retrieval task or the feature set being searched (a good feature set would score a higher AP on the same retrieval task than a bad feature set).

Stellenbosch University https://scholar.sun.ac.za

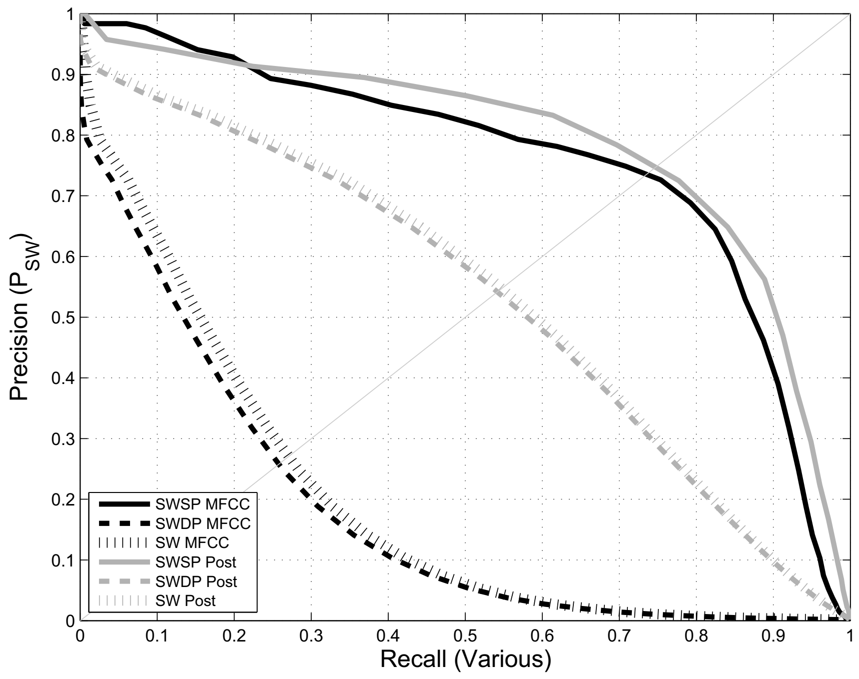

Figure 2.2: Examples of precision-recall curves from [2].

2.5.2. The same-different task

Carlin et al. [2] proposed the same-different task as an evaluation task to be used in zero resource settings. The purpose of the same-different task is to indicate how well a set of features might perform in a downstream task that requires speaker-independence and a high level of word discrimination. Given a set of words W, we define four sets of word pairs $(w_i, w_j) \in W \times W, i \neq j$:

- C_1: Same word, same speaker (SWSP)
- C_2: Same word, different speakers (SWDP)
- C_3: Different words, same speaker (DWSP)
- C_4: Different words, different speakers (DWDP)

For a threshold τ, a word pair is considered as belonging to the same word if $f_{\text{distance}}(w_i, w_j) \leq \tau$, where f_{distance} is a distance function that measures the similarity of the two words. In the case of the same-different task, Carlin et al. selected f_{distance} to be the minimum DTW alignment cost between the word pair. DTW alignment requires a distance metric to be specified. Carlin et al. [2] found the cosine distance produces the best results. By varying the threshold τ, a precision-recall curve is produced. From this curve, we derive the PRB and the AP. Given the task that wants to classify two words as

Stellenbosch University https://scholar.sun.ac.za

belonging to the same word identity regardless of speaker, we define

$$N_k(\tau) := |\{(w_i, w_j) \in C_k : DTW(w_i, w_j) \leq \tau\}| \tag{2.3}$$

to be the number of word pairs in C_k that are considered to be the same word by the DTW alignment cost. The precision and recall at a given threshold τ can then be defined as

$$P(\tau) = \frac{N_1(\tau)}{\Sigma_{k=1}^{4} N_k(\tau)} \tag{2.4}$$

and

$$R(\tau) = \frac{N_1(\tau) + N_2(\tau)}{|C_1| + |C_2|} \tag{2.5}$$

respectively. The AP is the metric used to measure the performance of a feature set. This evaluation task can be used to compare feature extractors. By extracting a set of features from a test set, the AP of that feature set can be calculated. This value can then be compared with the AP of another feature set extracted from the same test set. The extractor that produces the better performing features can be classified as the better extractor, as Carlin et al. [2] showed that a higher AP leads to better performance in downstream speech processing tasks.

2.5.3. The ABX task

Munson and Gardner [30] developed the ABX task as a method to minimise the variability in auditory testing. In their original experiment, a subject is presented with three sounds. The first two sounds, A and B, represent two similar but different signals. The third sound, X, corresponds to either sound A or sound B. The subject is then tasked to select which sound X matches closest. From the results, an error rate can be produced. This error rate measures the performance of the subject. In speech processing, ABX tasks can be used to evaluate speech representations. Schatz et al. [19] proposed the minimal-pair ABX task as an extension to Carlin et al.'s same-different task [2] to evaluate speech representations in a zero resource setting. Given three words w_a, w_b and w_x, we compute the DTW distances, as in [2], $DTW(w_a, w_x)$ and $DTW(w_b, w_x)$. The sign of $DTW(w_a, w_x) - DTW(w_b, w_x)$ is then used to classify w_x as belonging to B or A if the sign is positive or negative and from this, an error rate can be determined. Given two feature extractors, the extractor which produces features that score a lower error rate on the ABX task can be classified as the better extractor.

Stellenbosch University https://scholar.sun.ac.za

2.6. Neural networks

Neural networks are a subsection of machine learning. McCulloch and Pitts [31] modelled neurons in the human brain as a linear threshold gate. By stimulating the modelled neuron through a number of inputs, the neuron would be switched on if the combined inputs surpassed some threshold. This neuron could then be connected to other neurons, with each connection having a weighted value that stimulates the next neuron if the previous neuron is turned on. In doing so, a neural network is created. The McCulloch-Pitts neuron was a very simplistic representation of the neuron, as it could only produce binary output. In modern neural networks, the neurons (usually referred to as *units*) are defined by their activation functions. Various activation functions are discussed later in this chapter.

The term *deep learning* refers to a special subset of neural networks in which multiple neural network layers are connected to one another. This allows the networks to model complicated behaviours that a regular single-layer neural network could not.

The weights between each unit are referred to as the *network parameters*. These parameters are adjusted by *training* the network. Training refers to the process in which example data is fed to the network. By comparing the output of the network to the expected output of the example data, the network parameters can be updated to minimise the discrepancy between the predicted output and the wanted output. There are many methods of training a neural network, some of which are discussed in the next section of this chapter.

2.6.1. Optimisation

Neural networks are trained via a loss function (also called a cost function). When training a neural network, the goal is to minimise this loss function, as the minimum (specifically the global minimum, as there can be many local minima) should be the point at which the network implements the desired behaviour. Optimisers are algorithms that update the weight parameters of the network in order to minimise the network's loss function.

Gradient descent

Gradient descent is an iterative optimisation algorithm used to find the minimum of a function. Neural networks are usually initialised by having all of the weights set to random values that are close to zero. Weights are then updated iteratively by calculating the partial derivative of the loss function with respect to weight:

$$\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_{\tau} - \alpha \frac{\partial \ell}{\partial \boldsymbol{\theta}}, \tag{2.6}$$

where $\boldsymbol{\theta}$ is the weight vector of the network, α is the learning rate and ℓ is the loss function. The learning rate affects how aggressively the function is minimised. A higher learning

Stellenbosch University https://scholar.sun.ac.za

rate can converge quicker, but can also cause the function to never converge at all as the minimum will always be overshot. As mentioned before, a minimum is not necessarily the global minimum of the function. Gradient descent does not account for multiple minima and reaching the global minimum is dependent on how the weights were initialised, or it might even be impossible.

Batch gradient descent

Batch gradient descent [5] calculates gradients for the entire training set before adjusting the network weights by the average of the calculated gradients. This leads to a relatively smooth training curve, but a very large number of calculations has to be performed to take even a single training step, which can be infeasible for very large datasets.

Stochastic gradient descent

Stochastic gradient descent [5] (SGD) is a variation of gradient descent. *Stochastic* refers to the fact that it only uses a single example at a time. By using only a single example at a time, the gradient does not have to be calculated using the entire training set, thereby greatly reducing training time and allowing convergence even when the training set is very large. However, since the weights are updated very frequently, the cost function fluctuates a lot, which can lead the descent to a sub-optimal local minimum.

Mini-batch gradient descent

Mini-batch gradient descent is a combination of batch gradient descent and SGD. Instead of considering the entire dataset or just a single training sample during a training step, a subset of the training set is used. This leads to less frequent weight updates, causing the cost function to fluctuate less than it would have with SGD. The smaller batch size also means that weight updates will be more frequent than it would have been with batch gradient descent, leading to faster convergence.

AdaGrad

AdaGrad [32], or the adaptive gradient algorithm, is an optimisation algorithm proposed by Duchi et al. The term *adaptive* refers to the fact that the optimiser has a per-parameter learning rate that changes based on the parameters. This means each weight in the network has an individual learning rate, which is adjusted as the network reaches a minimum. Each parameter is updated as follows:

$$\theta_{j,t+1} = \theta_{j,t} - \frac{\eta}{\sqrt{G_{j,j,t}}} g_{j,t}, \tag{2.7}$$

Stellenbosch University https://scholar.sun.ac.za

where η is the base learning rate, $g_{j,t}$ is the current gradient of the parameter and $G_{j,j,t}$ is the sum of squared gradients over the training period. Because of the division by $\sqrt{G_{j,j,t}}$ in Equation 2.7, the learning rate for each parameter will decrease as training progresses, depending on the size of the gradients for that parameter.

ADADELTA

The ADADELTA [33] optimiser is an extension of the AdaGrad optimiser. The AdaGrad optimiser is very aggressive in its learning rate attenuation. Learning rates are reduced proportionally to the sum of their respective gradients throughout the entire training period. ADADELTA mitigates this by using a rolling window period for learning rate attenuation. Optimisations are done so that the entire rolling window of gradients do not have to be stored and updated. The final update function is as follows:

$$\theta_{t+1} = \theta_t + \Delta\theta_t, \tag{2.8}$$

where

$$\Delta\theta_t = -\frac{\text{RMS}(\Delta\theta_{t-1})}{\text{RMS}(g_t)} \cdot g_t. \tag{2.9}$$

RMS is the root mean squared error, and, as with AdaGrad, g_t is the current gradient vector of the parameters. We omit j as the parameter identifier for brevity.

Adam

Adam [34], or Adaptive Moment Estimation, stores a decaying average of past squared gradients just like Adadelta, but Adam also stores a decaying average of past gradients. The decaying average of past gradients (m_t) and the decaying average of past squared gradients (v_t) are calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.11}$$

β_1 and β_2 are the decay rates of their respective property. The decaying averages are initialised to zero at the start of training, leading to initial updates that are biased towards zero. In order to correct biases in the early training steps of the network, a bias corrected version of these parameters are calculated using the results of Equations 2.10 and 2.11:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.12}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.13}$$

Stellenbosch University https://scholar.sun.ac.za

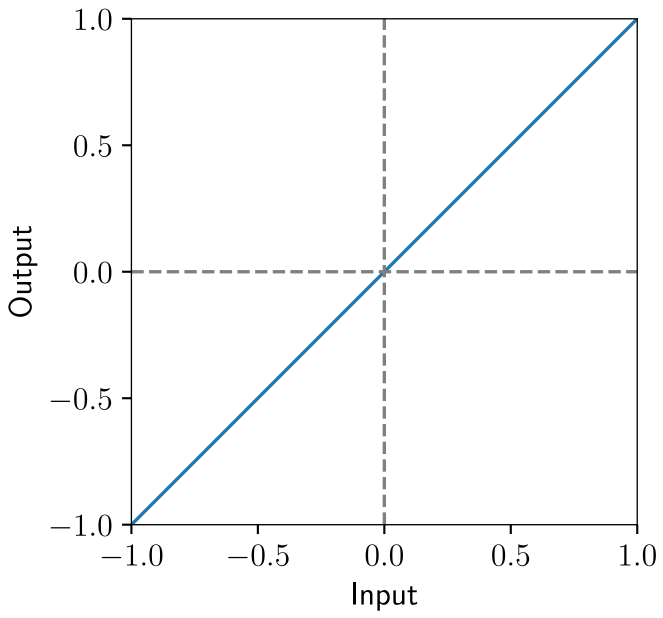

Figure 2.3: A linear activation function.

The final update rule for Adam is then

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{2.14}$$

where ϵ is a very small value, generally 10^{-8}, which prevents division by zero.

2.7. Activation functions

Activation functions determine the output of a unit in a neural network given an input. There are several activation functions that all serve different purposes.

2.7.1. Linear function

A linear activation function provides an output that is directly proportional to its input. A normal linear equation defines a linear activation:

$$f(x) = cx, \tag{2.15}$$

where c the gradient. Output units often use linear activations. In deep neural networks, if every layer were to use a linear activation function, it would defeat the purpose of having layers, as the end layer would end up being a linear scaling of the input, which could be achieved by a single layer.

Stellenbosch University https://scholar.sun.ac.za

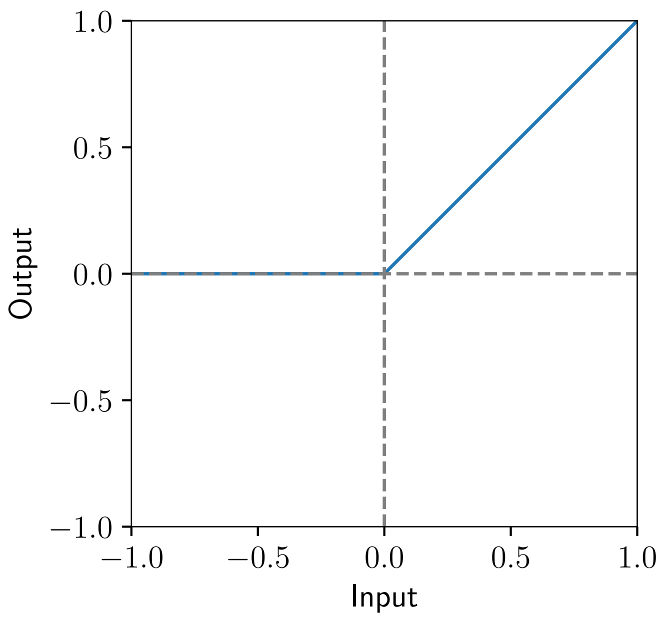

Figure 2.4: A ReLU activation function.

2.7.2. Rectified linear unit

The rectified linear unit [35] (ReLU) is one of the most popular activation function used in neural networks. The following equation defines a ReLU activation:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} . \tag{2.16}$$

While it looks like a linear function at first, it is not. The unit is switched off for any $x \leq 0$. This makes ReLU activations usable in deep neural networks ReLU activations are generally less computationally expensive, as no calculations have to be performed for a unit that is turned off. One issue with the ReLU activation function is that units can be permanently switched off. When this happens, the units will no longer respond to future inputs. Other variations of the ReLU function have been proposed to circumvent this issue.

Leaky ReLU

The following equation defines the leaky ReLU [36] (LReLU) activation function:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \end{cases} , \tag{2.17}$$

where a is usually around 0.01. It is very similar to the normal ReLU function, but LReLU has a very small gradient for negative values of x. This allows the network to recover if a

Stellenbosch University https://scholar.sun.ac.za

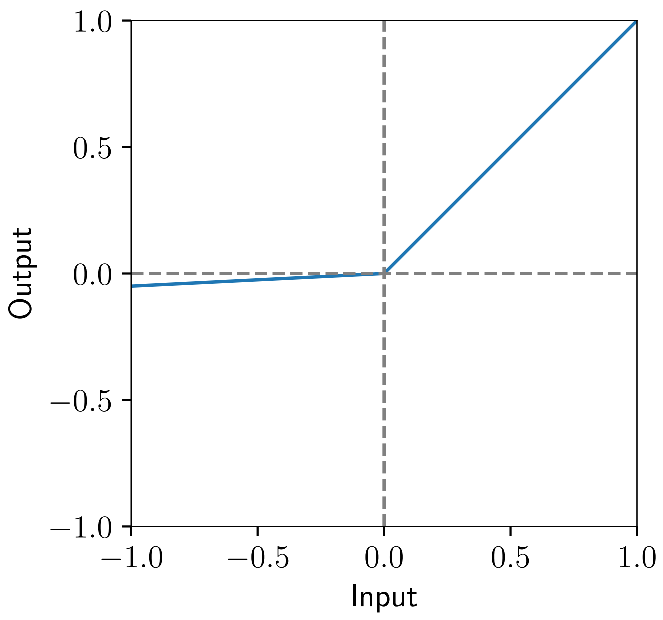

Figure 2.5: A leaky ReLU activation function.

unit is switched off.

Randomised Leaky ReLU

The randomised leaky ReLU (RReLU) is a variation of the RReLU that randomises the gradient for values less than zero. The following equation defines the LReLU activation function:

$$f(x_{ji}) = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji}x_{ji} & \text{if } x_{ji} < 0 \end{cases}, \tag{2.18}$$

where a_i is sampled from a uniform distribution between 0 and 1. Xu et al. [37] showed that the LReLU and RReLU perform better than the normal ReLU in classification tasks.

2.8. The Siamese network

Bromley et al. initially introduced the Siamese network [18] for signature verification. The network consists of two identical subnetworks with shared weights. The network is trained using pairs of inputs, where an input pair is classified as either positive or negative. A positive (or AA) input pair consists of two inputs which represent the same type of input (for example, two images that both represent a bowl). In contrast, a negative (or AB) input pair represents different types of input (such as one image depicting a bowl and the other a cup). During training, together with the input pair, the network is fed whether or

Stellenbosch University https://scholar.sun.ac.za

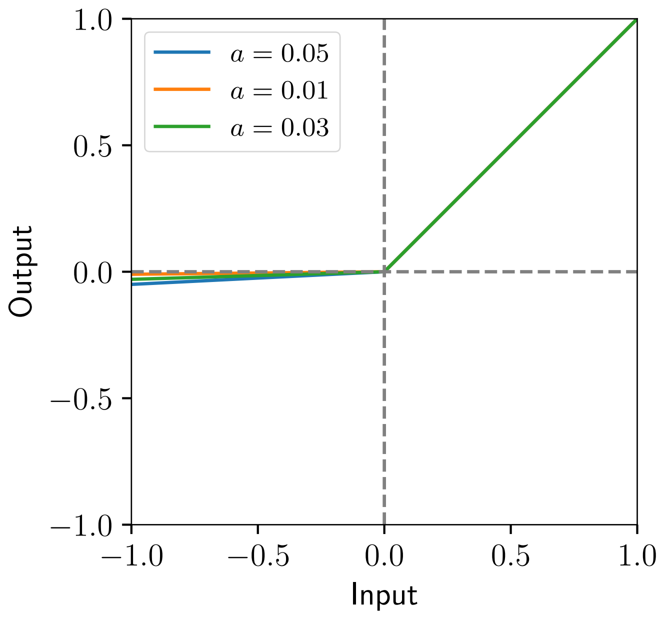

Figure 2.6: A randomised leaky ReLU activation function. The different values of a are randomised for each unit.

not the current pair is positive or negative.

Each subnetwork produces an result from its respective input, which we refer to as an embedding. Some distance function then measures how similar the two embeddings are, and depending on whether the input pair is positive or negative, the loss function attempts to minimise or maximise the distance function. An example of a Siamese network is depicted in Figure 2.7.

Thiollière et al. [38] trained a Siamese network on aligned frames from word pairs discovered through a spoken term discovery system. They found that the resulting feature extractor produced features that scored well on the minimal-pair ABX task [19].

Zeghidour et al. [15] evaluated the Siamese network's ability to disentangle speaker

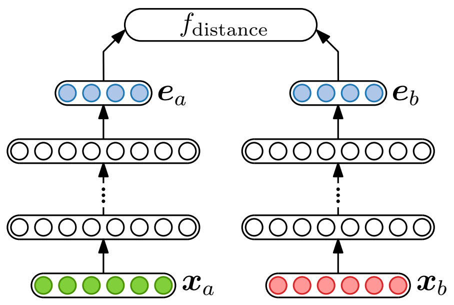

Figure 2.7: A Siamese network. The two subnetworks are identical and share weights. An embedding is extracted from each input and compared. Depending on whether or not the embeddings are representing the same type of input, a distance function is minimised or maximised.

Stellenbosch University https://scholar.sun.ac.za

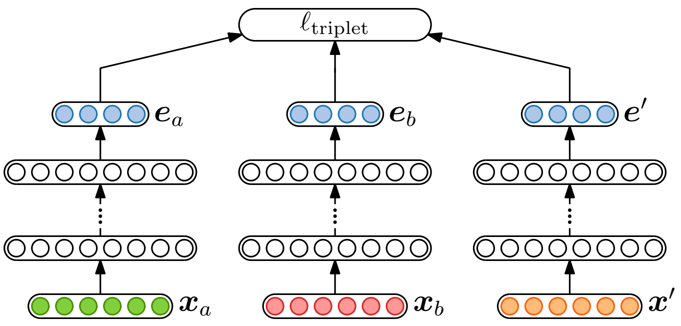

Figure 2.8: A Triamese network. The three subnetworks are identical and share weights. An embedding is extracted from each input. The loss function attempts to minimise the distance between two inputs while simultaneously maximising the distance to the third input.

and phonetic information in speech. The similarity of an embedding pair was measured by their cosine, and the pairwise loss function they proposed was

$$\ell_\gamma(\boldsymbol{e_a}, \boldsymbol{e_b}, y) = \begin{cases} -\cos(\boldsymbol{e_a}, \boldsymbol{e_b}) & \text{if } y = 1 \\ \max(0, \cos(\boldsymbol{e_a}, \boldsymbol{e_b}) - \gamma) & \text{if } y = 0 \end{cases}, \tag{2.19}$$

where γ is a margin hyperparameter that determines the minimum distance between the different word types. They used the Siamese network as the basis for their *Triamese* network.

2.9. The Triamese network

Zeghidour et al. [15] proposed the *Triamese* network. The Triamese network, which is a variation of the Siamese network, consists of three identical subnetworks, all with shared weights. A depiction of the network is illustrated in Figure 2.8.

The network uses AAB triplets, instead of the AA and AB pairs of the Siamese network, as input. Each subnetwork produces an embedding from its respective input, which is then fed to the network's loss function. Zeghidour et al. proposed the following loss function:

$$\ell_\gamma(\boldsymbol{e_a}, \boldsymbol{e_b}, \boldsymbol{e}') = \max(0, \gamma - \cos(\boldsymbol{e_a}, \boldsymbol{e_b}) + \cos(\boldsymbol{e_a}, \boldsymbol{e}'), \tag{2.20}$$

where γ is a margin hyperparameter. Given three inputs $(\boldsymbol{w}_a, \boldsymbol{w}_b, \boldsymbol{w}')$, we would like the network to minimise the distance between embeddings $\boldsymbol{e}_a$ and $\boldsymbol{e}_b$ and maximise the distance between $\boldsymbol{e}_a$ and $\boldsymbol{e}'$. Since $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ are phonetically similar, the loss function would try to emphasise the similarities between $\boldsymbol{e}_a$ and $\boldsymbol{e}_b$ so that they are always more similar than $\boldsymbol{e}_a$ and $\boldsymbol{e}'$ by some margin γ. Zeghidour et al. attempted to use the Triamese network to disentangle both speaker and phonetic information in a single network by extracting two embeddings from each input and then applying two loss functions, but it

Stellenbosch University https://scholar.sun.ac.za

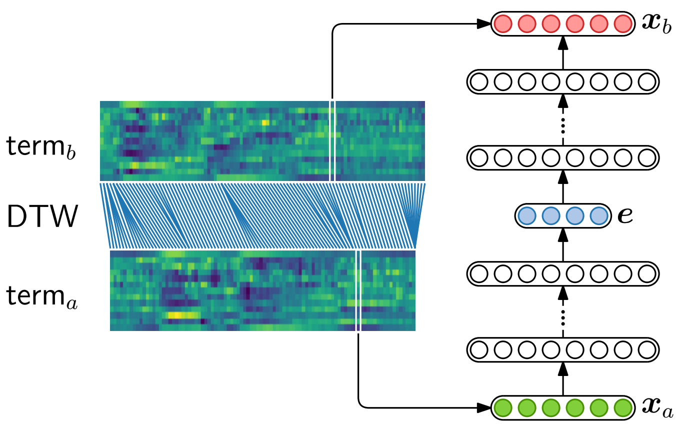

Figure 2.9: The CAE is trained using word pairs consisting of similar words spoken by a variety of speakers [1]. The network consists of two halves: an encoder and a decoder. Feature embeddings are extracted from the middle layer.

did not produce significantly better results.

2.10. The correspondence autoencoder

In [6], Kamper et al. proposed the correspondence autoencoder (CAE), an autoencoder-like network which, instead of using the same input as the output target, is trained on input-output pairs of similar words spoken by a variety of speakers. The network can be thought of consisting of two parts: an encoder and a decoder. In the middle, where the encoder and decoder connects, is an embedding layer from which features are extracted. Since the network requires weak top-down supervision in the form of word pairs, Kamper et al. paired the network with a UTD system so that it can operate in a zero resource environment. The network is trained on a frame-by-frame basis, so in order to make each frame in the input-output pair comparable, the two terms are DTW aligned. They found that the features produced by the network came close to matching the performance of a supervised network. An illustration of the network can be found in Figure 2.9.

Renshaw et al. [39] showed that the CAE could produce better features by first training the network in a supervised environment on a high-resource language, followed by unsupervised language adaptation to a zero resource language.

2.11. Chapter summary

In this chapter, several topics that are relevant to the rest of this thesis were discussed. We gave an overview of zero resource environments and discussed how the lack of annotated data in such environments present problems to conventional machine learning methods. We discussed how UTD systems can be used to annotate raw data, and, in doing so, allow otherwise unannotated data to be used by supervised learning methods. We gave an

Stellenbosch University https://scholar.sun.ac.za

overview of dynamic time warping and showed how mel-frequency cepstral coefficients can be derived. After discussing precision and recall, two measures often used to evaluate the performance of retrieval tasks, we gave an overview of two of the evaluation tasks often used to evaluate speech features: the same-different task and the ABX task. After a brief summary of neural networks, we discussed some often used optimisers and activation functions. Finally, we gave an overview of the Siamese, Triamese and correspondence autoencoders, as these networks form the basis for the rest of the thesis.

Stellenbosch University https://scholar.sun.ac.za

Chapter 3

Datasets

In this chapter, we give an overview of the datasets each model is trained and evaluated on. Firstly, we give an overview of the Buckeye corpus and the NCHLT dataset. After this, we explain how the datasets are split into their respective training, validation and test sets. We also give an overview of how these datasets are processed so that we can use them for training our neural networks.

3.1. The Buckeye corpus of conversational English

The Buckeye corpus of conversational English [20] consists of the recordings of forty long-time Caucasian residents from Columbus, Ohio — twenty male, twenty female; twenty young, twenty old — during an interview. Each speaker was interviewed for about one hour. The interviews were conducted in a quiet room, and the acoustic signals are clear of noise. Each interview has been transcribed. The word labels, together with the timestamps of where they occur in the file, have been saved in word label files. Each phone has also been labelled and stored in a file together with its timestamp. In total, about 300 000 words were collected. This is the English dataset we will use to develop our models. The large size of the dataset allows us to comfortably split it into subsets for training, validation and testing. The labelled data also allows us to test each network on actual words. The variety in speakers serves our purpose of validating how well each network extracts speaker-independent features from speech data.

3.1.1. Training, validation and test sets

The English dataset is divided into four subsets:

- a training set consisting of 12021 discovered terms spoken by 12 speakers,
- a training set consisting of 5106 ground truth terms spoken by 12 speakers,
- a validation set consisting of 2733 ground truth terms spoken by 8 speakers,
- and a test set consisting of 4052 ground truth terms spoken by 12 speakers.

Stellenbosch University https://scholar.sun.ac.za

Table 3.1: Speaker distribution in the Buckeye subsets.

Subset	Speaker list
Training	s02, s03, s04, s05, s06, s08, s10, s11, s12, s13, s16, s38
Validation	s17, s18, s19, s22, s34, s37, s39, s40
Test	s01, s20, s23, s24, s25, s26, s27, s29, s30, s31, s32, s33

Since our goal is to evaluate the speaker independence of the features produced by each network, the training, test and validation sets were chosen to have no overlap in speaker identities. The speaker identities in each set are displayed in Table 3.1. The identities are represented as "sXX", where XX is a numerical value. The ground truth terms are terms that correspond to actual words, instead of discovered words. A training set of these is included so that the performance loss of training on discovered terms can be measured. The speaker sets are the same as those used by Kamper et al. in [27]. There are 32 speakers used in total. Kamper et al. had a fourth set, but that set was not used. Each subset of speakers was randomly assigned, under the condition that each subset should have an equal number of old males, young males, old females and young females.

3.2. The NCHLT speech corpus of South African languages

The NCHLT speech corpus of South African Languages contains speech from all eleven official languages of South Africa, with approximately 200 speakers per language [21]. The NCHLT project is an expansion of the African Speech Technologies (AST) and Lwazi projects. The corpus was created to develop large-vocabulary speech recognition systems, such as voice search or indexing. We will only be using the Xitsonga language from this dataset, and it will be treated as a zero resource environment. During training, no access to word labels will be used. The Xitsonga set consists of 200 speakers: 95 men and 105 women. There are 142 hours of speech in total. As with the Buckeye corpus, the large dataset, access to word labels for testing, and the large variety of speakers makes it perfect for training each network to produce features that are speaker-independent, as well as measuring how well the features produced by each network performs on an evaluation task.

3.2.1. Training and test sets

We divide the Xitsonga dataset into two subsets:

- a training set consisting of 11705 discovered terms,

Stellenbosch University https://scholar.sun.ac.za

Table 3.2: Speaker distribution in the Xitsonga subsets.

Subset	Speaker list
Training and test	001m, 102f, 103f, 104f, 126f, 127f, 128m, 129f, 130m, 131f, 132m, 133f, 134m, 135m, 136f, 138m, 139f, 140f, 141m, 142m, 143m, 144m, 145m, 146f

- and a test set consisting of 6384 ground truth terms.

There is no validation subset for this dataset. The purpose of this dataset is to evaluate the performance of the produced features in a completely unsupervised manner, so we will not be able to fine tune the networks on this dataset at all. The purpose of this dataset is to evaluate features produced for a fixed set of speakers and, as such, the training and test sets consist of the same speaker identities. The speaker list is shown in Table 3.2.

3.3. Data processing

The following section describes how the datasets are processed so that they can be used in the training of the neural networks. The processing steps used were developed by Kamper et al. in [27].

3.3.1. Extract MFCCs with CMVN, deltas and delta-deltas

First we extract the MFCCs from the sound files. MFCCs are an excellent start for any system trained on speech data. If we were not worried about correlation in inputs influencing our data, we would use filter-banks instead. We also use cepstral mean and variance normalisation to normalise our data, as normalisation has been shown to improve the performance on a wide variety of neural networks. Cepstral mean and variance normalisation is applied on each utterance discovered by the UTD. It should be noted that utterance length can have an effect on the normalised features. Lastly, we append the first and second order derivatives of the MFCCs (deltas and delta-deltas) to the MFCCs, which has been shown to improve the performance of neural networks when included during training. This results in a 39-dimensional feature vector.

3.3.2. Unsupervised term discovery (UTD)

Next we employ the unsupervised term discovery (UTD) system proposed by Jansen and Van Durme [26] to discover repeated terms in the sound files[1]. From this, we also create a

[1]The UTD system was not actually used to generate the terms here. Instead, pre-discovered terms was made available and those terms were used. However, these terms were discovered using said UTD system, but not by the author himself.

Stellenbosch University https://scholar.sun.ac.za

list of word pairs. For both datasets, all possible terms discovered by the UTD is used.

3.3.3. Voice activity detection

Voice activity detection (VAD) is used to determine which parts of a sound file contains speech, as during natural conversation there can be short or long periods of silence in between actual spoken words. These silences add no value to our system, so we use a VAD system to remove these silences from each word pair, leaving only the sections in each pair that contain speech.

For the purposes of evaluating the features produced by each network an actual VAD system was not used. Instead, oracle speech regions obtained by force aligning the terms were used. An actual VAD system would introduce uncertainty into the experiment and not contribute anything to what we are actually trying to evaluate, which is the features each network produces. However, in a true zero resource environment a VAD systtem would be required.

3.3.4. Word pair alignment

Finally, in order to have comparable frames in the word pairs, we use dynamic time warping (DTW) alignment. The word pairs share a word identity, although not necessarily a speaker. For the CAE and Siamese networks, these are positive (AA) word pairs. By using DTW alignment, the word pair frames are re-indexed so that similar frames in each word correspond to one another. This is essential since the same word can be spoken at different speeds, which misaligns the comparable frames in each word. The networks are trained frame-by-frame and comparing different frames to one another could be very detrimental.

3.4. Chapter summary

In this chapter, the datasets used in the experiments in this thesis were described. We discussed the contents of the datasets and why they were chosen for these experiments. Finally, we described how these datasets were processed so that they could be used in the training of the neural networks.

Stellenbosch University https://scholar.sun.ac.za

Chapter 4

A comparison of the correspondence autoencoder and the Triamese network

In this chapter we describe an experiment in which we compare the performance of speech features produced by the CAE and Triamese networks using the same-different evaluation task [2], which is a word discrimination task that also measures speaker independence. Firstly, we give an overview of each network, followed by a discussion of some differences and similarities between the networks. Following this, we state the purpose of the experiment and what we aim to learn from the experiment. We briefly describe how each dataset is partitioned in order to perform the experiment. We give a short description of the evaluation metric used to evaluate the speech features. In the experimental setup we describe the specifics of each network, such as the architecture, hyperparameters, loss functions and optimisers. We explain how each network is trained and evaluated. Finally, we present the results of our experiment and discuss its implications. We conclude the chapter with a short summary of the experiment and our findings.

4.1. The networks

In this section we give an overview of the correspondence autoencoder and the Triamese network. A detailed literature study was given on these networks in Chapter 2. Here we focus on the workings of these networks, instead of their origin or the results of their respective experiments. Firstly, we examine the CAE, followed by an examination of the Triamese network. We then compare the two networks.

4.1.1. The correspondence autoencoder (CAE)

The correspondence autoencoder [6], as described in Section 2.10, is an autoencoder-like neural network. Unlike general autoencoders, the network is not trained by having the network reconstruct the input at the output. Instead, the network is trained using weak top-down supervision in the form of word pairs that share the same word identity, although they do not necessarily share the same speaker. Each word pair is aligned using dynamic programming and the network is trained frame by frame. The CAE consists of two halves:

Stellenbosch University https://scholar.sun.ac.za

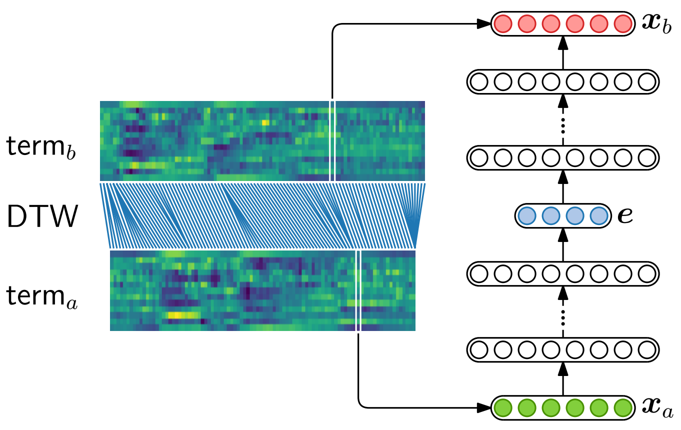

Figure 4.1: The correspondence autoencoder (CAE) is trained to reconstruct one acoustic frame in a discovered word from another [1].

an encoder and a decoder. During training, the encoder receives one of the word instances, $\boldsymbol{x}_a$ as input. The encoder transforms the provided input into an embedding $\boldsymbol{e}$. The decoder then takes this embedding and attempts to transform it into the second word instance, $\boldsymbol{x}_b$. The loss function requires that the network minimise the distance between the target output and the decoded embedding. Due to the word pairs not necessarily sharing a speaker, the network learns to maintain features that contribute to the core word identity, while disregarding features that are speaker-specific, like tone of voice. Once trained, the encoder can be separated from the rest of the network and functions as a generic feature extractor. The features extracted in this manner have proven to be less speaker-dependent than generic MFCCs and shows promise in improving the general performance of downstream tasks such as query by example search and indexing. The network is illustrated in Figure 4.1.

4.1.2. The Triamese network

The Triamese network [15], as described in Section 2.9, is an adaptation of the Siamese neural network [18]. The Siamese network consists of two branches of identical subnetworks with shared weights. The network is trained on AA (positive) or AB (negative) input pairs, with the network being told whether the current pair on which it is training is positive or negative. In the context of speech processing, a positive input pair would consist of two words of the same type, but not the same speakers. A negative pair would be words of different types, but from the same speaker. Each subnetwork produces an embedding from its corresponding input and a distance function is applied to measure the similarity of the two embeddings. A loss function then minimises the distance between positive embedding pairs and maximises the distance between negative embedding pairs. This kind of loss function is called a contrastive loss function. Features that contribute to speaker identity is de-emphasised by negative pairs sharing a speaker, while features maintaining word

Stellenbosch University https://scholar.sun.ac.za

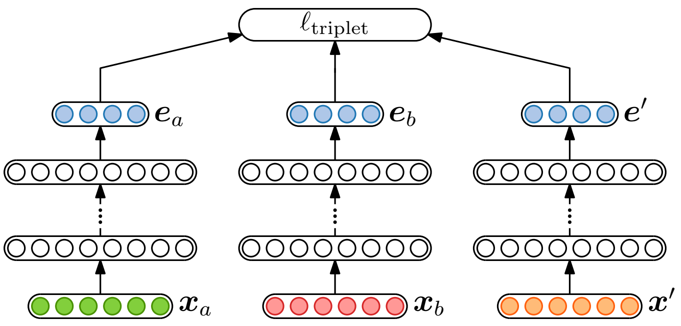

Figure 4.2: The Triamese network is trained so that the embeddings $\boldsymbol{e}_a$ and $\boldsymbol{e}_b$ of the same type are more similar by a margin m than embeddings $\boldsymbol{e}_a$ and $\boldsymbol{e}'$ of different types.

identity is emphasised by the positive word pairs.

The Triamese network, on the other hand, consists of three branches of identical subnetworks with shared weights. The network is trained on AAB triplets, which we represent as $(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}')$. $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ form a positive AA pair, and $\boldsymbol{x}_a$ and $\boldsymbol{x}'$ form a negative AB pair. In the context of speech processing, $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ would share a word identity but not necessarily a speaker identity. $\boldsymbol{x}_a$ and $\boldsymbol{x}'$ would not share a word identity, but would most likely share a speaker identity. Each subnetwork produces an embedding from its respective input. A distance function calculates the distances between the positive and negative pairs respectively. These distances are then passed to a contrastive loss function which simultaneously minimises the distance between the positive embedding pair, while maximising the distance between the negative embedding pair. As in the Siamese network, the positive pair encourages the network to maintain features that contribute to word identity, while the negative pair discourages features that contribute to speaker identity. In the Triamese network the positive and negative embedding pairs are handled simultaneously, whereas in the Siamese network they are handled separately. The Triamese network is illustrated in Figure 4.2.

4.1.3. Network comparison

Both the CAE and the Triamese networks were developed for the purpose of speaker-independent feature extraction. Both networks are also trained on frames of word pairs, where each frame shares a word identity but not necessarily a speaker identity. In the case of the Triamese network, there is a third frame as well, since the Triamese network uses a contrastive loss function which requires a negative example. While the Triamese network compares both inputs directly on the embedding level (by computing the distance between the produced feature embeddings), the CAE first decodes the feature embedding and attempts to construct the second input from the features extracted from the first input. Ignoring the negative negative input subnetwork of the Triamese network, the Triamese network can be seen as two encoders trying to produce the same feature embedding from

Stellenbosch University https://scholar.sun.ac.za

similar inputs, whereas the CAE is an encoder trying to produce a feature embedding that is generic enough so that a decoder can construct an output that is similar to the input.

4.2. Experiments

In this section we describe the process through we which we compared the CAE and Triamese networks. Firstly, we describe the purpose of this experiment and what we aim to learn from its results. We then describe the experimental setup: the datasets used, the hyperparameters chosen for each network, the training procedure and the evaluation metric used. Next we present the results from the experiment, discussing the outcomes and possible reasons for the results. Finally we summarise our findings and draw conclusions based on the outcomes.

4.2.1. Purpose of experiment

The purpose of this experiment is to perform a fair comparison of the CAE and Triamese networks. Both are promising networks in the field of speaker-independent feature extraction, yet these networks have never been compared by training them on the same datasets and evaluating them using the same evaluation metric. We aim to evaluate each network's capability of training a generic feature extractor in an unsupervised environment. Each network will be trained on words produced by an unsupervised term discovery system. The features should disregard speaker-specific information, such as gender and tone of voice, while maintaining linguistically meaningful contrasts that contribute to word identity. We would also like to know how each network is affected by the fact that it is operating in an unsupervised environment. To establish this, we also evaluate each network on *ground truth* terms. These are terms that correspond to actual words, not words discovered by the UTD system. This should give a fair indication as to how each network is affected by the use of UTD terms.

4.2.2. Datasets

We use the datasets as outlined in Chapter 3, which we briefly summarise here.

The Buckeye corpus of conversational English

We use the English dataset, the Buckeye corpus of conversational English [20], as our development dataset. The dataset is split into three subsets: a training set, a validation set, and a test set, with no speaker identity overlap in any of the sets. Each model is trained using the training set and feature performance as training progresses is tracked using the validation set. If performance on the validation set is not satisfactory, hyperparameters

Stellenbosch University https://scholar.sun.ac.za

and network variables are tweaked and the model is trained again[1]. Once performance on the validation set is satisfactory, the model is tested a single time on the test set.

The NCHLT speech corpus of South African languages

We use the Xitsonga dataset from the NCHLT speech corpus of the South African languages as our zero-resource environment. The dataset is split into two subsets: a training set, and a test set. This dataset represents a zero resource environment. Because of this, there is no validation set, and no fine-tuning will be performed on this dataset, as it would not be possible in a truly zero resource environment. A model will be trained on this dataset for a fixed number of epochs. The feature performance will be evaluated only once.

4.2.3. Unsupervised term discovery

For the unsupervised tests, word pairs are produced from each dataset using the UTD system proposed by [26]. From the UTD system we produce word pairs which we align using DTW alignment. However, the Triamese network requires triplets as input. An aligned word pair satisfies two of the three inputs in the Triamese input triplet, so only a third (negative) example is necessary to form a complete triplet. For each word pair produced by the UTD system, we randomly sample a third word from the same dataset the satisfies the following conditions:

- It must not share a predicted word identity with the word pair.
- It must share a speaker identity with the first word in the word pair.

This third word most likely contains a different number of frames than the DTW-aligned word pair. Where the word pair could be DTW-aligned because they share a word identity, this third word cannot. Instead, a nearest neighbour scaling is performed on the frame indices to have the start and end frames of the third word align with that of the original word pair.

4.2.4. Evaluation metric

To evaluate the performance of the features produced by each network we use the same-different task [2] as described in Section 2.5.1. The same-different task is a word discrimination task that also measures speaker independence and was specifically developed to evaluate speech features in a zero resource setting.

[1]A holdout approach is used instead of cross validation, simply because the current state of the code runs the evaluation task as three separate background processes, which makes training and evaluation automation difficult.

Stellenbosch University https://scholar.sun.ac.za

4.2.5. Experimental setup

In this section we describe the specifics of each network architecture. We also describe the training and evaluation procedure for each network.

Correspondence autoencoder

For the CAE, we use the network setup that [6] found to be optimal. The network uses a 39-unit input layer. Following this are six 100-unit densely connected hidden layers, each using a ReLU activation. Concluding the encoder half of the network is a 39-unit densely connected layer, also using a ReLU activation. This is the layer from which the embeddings are extracted. We maintain the same dimensionality as the input so that we can compare the performance of the extracted features with that of the MFCCs that we use to train the network. The decoder half of the network mirrors the encoder half. Following the 39-unit embedding layer are another six 100-unit densely connected hidden layers, each using a ReLU activation. Finally, the output layer is a 39-unit densely connected layer with linear activation. The network is trained using an Adadelta [33] optimiser with a learning rate of 0.001 and a mean squared error loss function. A code implementation of the network can be found in Appendix A.1.

Triamese network

All three subnetworks in the Triamese network are identical and share weights. Each subnetwork starts with a 39-unit input layer, followed by six 100-unit densely connected hidden layers with ReLU activations. The final embedding layer is a 39-unit densely connected layer with a ReLU activation. Zeghidour et al. [15] used RReLU activation functions, but noted that it performed the same as the regular ReLU activation function. We use the loss function proposed by Zeghidour et al. [15] which we showed in Equation 2.20. We list it here for convenience:

$$\ell_\gamma(\boldsymbol{e}_a, \boldsymbol{e}_b, \boldsymbol{e}') = \max(0, \gamma - \cos(\boldsymbol{e}_a, \boldsymbol{e}_b) + \cos(\boldsymbol{e}_a, \boldsymbol{e}')), \tag{4.1}$$

For the margin parameter γ we use a value of 0.15, which Zeghidour et al. [15] found to be optimal. Finally, the network is trained using stochastic gradient descent with a learning rate of 0.01 and a decay of 10^{-6}. We also train the network using an Adam optimiser with a learning rate of 0.001. A code implementation of the network and the loss function can be found in Appendix A.2.

Training

We train each network using the training set from the Buckeye corpus. Each network was trained frame by frame using pairs discovered via an unsupervised term discovery system.

Stellenbosch University https://scholar.sun.ac.za

Table 4.1: Results from the evaluation task show that, when trained on UTD terms, features produced by the CAE outperforms those produced by the Triamese network. The Triamese network also failed to outperform the baseline MFCCs on the English test set, possibly due to being sensitive to the selection of training pairs.

	English validation		**English test**		**Xitsonga**	
Model	AP	PRB	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400	0.281	0.344
CAE	**0.474**	**0.477**	**0.460**	**0.485**	**0.574**	**0.561**
Triamese	0.382	0.414	0.348	0.387	0.459	0.483

Each network was trained 20 epochs at a time, and we evaluated each network using the validation set between each training session. We also saved the model after each training session, so that as soon as performance began to decline, we could use the previous model as our feature extractor. The CAE was trained for 180 epochs in total, while the Triamese network was trained for 140 epochs in total.

For our unsupervised test, no fine-tuning of a network was allowed. We noted the final number of epochs used to produce the best model for each network on the Buckeye corpus and trained each network on the Xitsonga dataset for the same number of epochs.

Evaluation

Once training was complete, we used a feature extractor from each network. For the CAE it would be the encoding half of the network. For the Triamese network it would be a single branch of the network. We used each feature extractor to produce features from the test sets of the respective datasets. The average precision of each feature set was then calculated.

4.2.6. Results

Table 4.1 shows the average precision of each model on each dataset when trained on terms discoverd by the UTD system. We include the performance of the MFCCs from each dataset as a baseline.

The CAE outperformed the baseline MFCCs and the Triamese network on each dataset. The Triamese network, on the other hand, failed to outperform the baseline MFCCs on the English test set. Riad et al. [40] found that Siamese networks could be rather sensitive to the input pairs they are trained on. It stands to reason that the Triamese network, as a derivative of the Siamese network, also inherits this property.

Not shown in the table is the fact that the Triamese network tended to be prone to overfitting. The average precision of features produced by the network once it was trained

Stellenbosch University https://scholar.sun.ac.za

Table 4.2: Results from the evaluation task show that, when trained on ground truth terms, features produced by the CAE outperforms those produced by the Triamese network. Using an Adam optimiser instead of stochastic gradient descent worsened the performance of the Triamese network.

	English validation		**English test**	
Model	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400
CAE	**0.519**	**0.513**	**0.455**	**0.474**
Triamese (SGD)	0.463	0.471	0.388	0.419
Triamese (Adam)	0.431	0.452	N/A	N/A

past its optimal operating point declined sharply, whereas the CAE was very resistant to overfitting.

Table 4.2 shows the average precision of each model on each dataset when trained on the ground truth terms. This table does not contain a column for the Xitsonga dataset, since that dataset was only used to evaluate the performance of features produced in a completely unsupervised scenario. MFCCs are once again included as a baseline. The Triamese network was also trained and evaluated using the Adam optimiser instead of stochastic gradient descent, just to see how the difference in optimiser affected the outcome.

The CAE still outperformed the Triamese network on the ground truth terms. However, we see a large increase in performance for the Triamese network when trained on ground truth terms instead of the UTD terms. This leads us to believe that the Triamese network is somewhat more sensitive to noisy data. The use of the Adam optimiser instead of stochastic gradient descent on the Triamese network degraded performance slightly on the evaluation set, so we decided to not run it on the test set.

4.3. Chapter summary and conclusions

We compared the performance of speech features produced by the CAE and Triamese networks using an evaluation task that measures word discrimination and speaker independence. Features produced by the CAE outperformed features produced by the Triamese network when trained on discovered terms across both the English and Xitsonga datasets. The Triamese network performed slightly worse than standard MFCCs on the English test set, but, as noted by [40], Siamese networks (and possibly their derivitives) can be very sensitive to the pairs on which they are trained. This notion is further confirmed by the fact that the Triamese network performed significantly better when trained on terms that correspond to actual words. The UTD system introduces some degree of noise to the data, since the discovered terms are not guaranteed to be perfect. We can also conclude that the CAE tends to be somewhat more robust than the Triamese network, as it was

Stellenbosch University https://scholar.sun.ac.za

less prone to overfit during training and did not see as large a jump in performance when trained on ground truth terms.

Stellenbosch University https://scholar.sun.ac.za

Chapter 5

Introducing speaker information into the CAE

In this chapter we investigate whether speaker-conditioning can improve the features produced by the CAE. Firstly, we describe our hypothesis. Next, we explain the method by which we are going to introduce speaker-conditioning to the CAE. After the model description, we describe the experimental environment and discuss the results of the experiment.

5.1. Hypothesis

As mentioned before, the CAE consists of two halves: an encoder and a decoder. However, once trained, only the encoder is used as a generic feature extractor. What this means is that we are free to modify the decoder in whatever way during training, as it will not affect how we use the encoder once training is complete. This raises a question: what can we do to the decoder to improve the quality of the features produced by the encoder? The CAE is trained using weak top-down supervision provided by discovered word pairs. We know the following about the word pairs: that they correspond to the same word identity (at least according to the word discovery task), and we know the speaker identity of each word pair. During training, the network tries to transform the speaker of a word from the input speaker to the output speaker. If we can help the network during this decoding process, the encoder might have more freedom to focus on maintaining the word identity, rather than maintaining speaker features that are common to both speakers. Thus, if we can augment the decoder's knowledge about the speaker it is trying to replicate, we may improve the speaker-independence of the features produced by the encoder.

5.1.1. Adding speaker information

Since we only have a speaker identity to work with, we are going to investigate the possibility of using this identity to learn a speaker representation of each speaker during the training of the network. This is similar to what Zeghidour et al. did in [15]. However, the focus of their experiment was to see if speaker and phonetic information could be

Stellenbosch University https://scholar.sun.ac.za

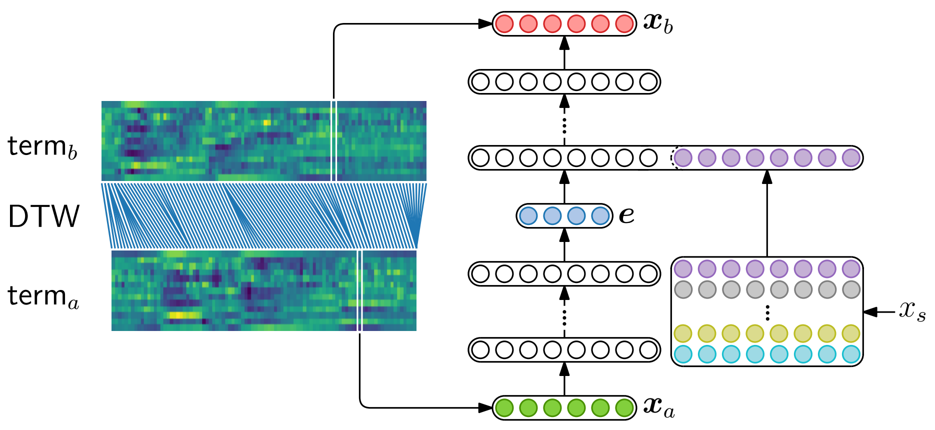

Figure 5.1: Speaker information is introduced to the CAE by manner of an $n \times m$ matrix that is trained alongside with the network.

disentangled using a single network. Their network also produced a generic speaker representation extractor, rather than learning individual speaker representations for each speaker in the dataset. The purpose of our experiment is to learn a speaker representation that can augment the decoder task during training. To represent our speaker information, we add an $n \times m$ matrix to the network, where n is equal to the number of speakers in our dataset and m is the dimensionality of our speaker representation. During training, we pair the identity of the output speaker together with the input-output pair. The network selects a representation from the matrix based on the speaker identity and concatenates it to the first hidden layer following the embedding layer. This provides the decoder with m extra units which it can use to approximate the output target more closely. Initially, these matrix values would be randomised. However, as the network encounters the same speaker time and again, these values should approximate a different set of values for each speaker. The resulting network is depicted in Figure 5.1.

5.2. Experiments

In this section, we describe the experimental setup used to determine the effectiveness of the proposed method.

5.2.1. Experimental setup

As before, the network is developed using the Buckeye corpus. A final unsupervised test is performed using the Xitsonga language from NCHLT dataset. The training set of the Buckeye corpus contains 13 speakers, making $n = 13$. For m, we chose a value of 100. The rest of the CAE uses the same setup as the experiment in Section 4, which we restate here for convenience. The network has an input layer of 39-units, followed by six 100-unit densely connected hidden layers using ReLU activation. Following this is

Stellenbosch University https://scholar.sun.ac.za

Table 5.1: Each network is trained using discovered terms. The network is then used as a feature extractor, and the features are evaluated using the same-different task.

	English validation		English test		Xitsonga	
Model	AP	PRB	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400	0.281	0.344
CAE	**0.474**	**0.477**	0.460	**0.485**	0.574	0.561
CAE w/ speaker	0.472	0.475	**0.464**	0.464	**0.610**	**0.595**

Table 5.2: Each network is trained using ground truth terms. The network is then used as a feature extractor, and the features are evaluated using the same-different task.

	English validation		English test	
Model	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400
CAE	**0.519**	**0.513**	**0.455**	**0.474**
CAE w/ speaker	0.462	0.478	0.445	0.469

a 39-unit embedding layer, also with ReLU activation. The decoder section starts with a 100-unit densely connected layer to which we concatenate the speaker representation. Following the concatenated layer is five 100-unit densely connected layers, each with ReLU activation and a final 39-unit output layer with linear activation. The network is trained using a mean squared error loss using an Adadelta [33] optimiser with a learning rate of 0.001. The features produced by the network is evaluated using the same-different task to calculate an average precision. A code implementation of the network can be found in Appendix A.3.

5.2.2. Results

The results of the experiment are shown in Table 5.1.

Adding speaker conditioning to the CAE did not yield significant results when the network is trained on discovered words. Features from the speaker-conditioned CAE performed slightly better than the regular CAE on the English test set, but this improvement is negligible and is possibly due to the random weight initialisation of the network. However, in our unsupervised environment, features from the speaker-conditioned CAE did manage to achieve a higher AP than their non-speaker-conditioned counterparts.

The results for the evaluations using ground truth terms are shown in Table 5.2. The additional speaker conditioning worsened the performance of the CAE on both the validation and the test sets. Since the Xitsonga dataset is only used for unsupervised

Stellenbosch University https://scholar.sun.ac.za

testing, we could not train the network on that dataset in a supervised capacity.

5.3. Chapter summary and conclusions

In this chapter, we conducted an experiment to see whether or not speaker conditioning could be used to improve the features extracted by the CAE. Speaker representations were trained along with the network and fed back into the decoder half of the CAE. In most cases, speaker conditioning did not improve the features extracted by the CAE. Average precision remained roughly the same on all datasets. However, on the unsupervised Xitsonga set, the speaker-conditioned features did manage to outperform their non-speaker-conditioned counterparts. We conclude that there is still potential for speaker-conditioning to improve the features extracted by a network, but further experiments would have to be conducted. Future work should explore concatenating the speaker representations at different layers.

Stellenbosch University https://scholar.sun.ac.za

Chapter 6

The Correspondence Triamese Network

In this chapter we introduce the *correspondence Triamese* (CTriamese) network. First we give a brief overview of the CAE and Triamese networks, after which we discuss how these networks can be complementary to one another. Next we describe our hypothesis, followed by a detailed network architecture. After describing the model architecture, we describe the implementation details and give an overview of the experimental environment. Finally, we show that features produced by the CTriamese network outperform features from both the CAE and the Triamese networks in the same-different task. We also show that, unlike the CAE, the CTriamese network consistently benefits from the inclusion of speaker conditioning.

6.1. CAE and Triamese

We did a thorough comparison of the CAE and Triamese networks in Chapter 4 using the same-different evaluation task. Both networks are trained using word pairs, where the words share a word identity but do not necessarily share a speaker identity. Both networks also extract an embedding from a given input, but where the Triamese network compares that extracted embedding with another embedding, the CAE attempts to decode the embedding to the output word of the input-output pair. The networks are also fundamentally different in their loss functions. The Triamese network's contrastive loss function prevents the network from simply zeroing all the feature embeddings. Zeroing the embeddings would minimise the distance between the two similar embeddings. However, it would also minimise the distance between the differing embeddings, which needs to be maximised instead. The CAE's loss function only compares the feature embedding with the intended output once it has been decoded.

When we compare a single branch of the Triamese network with the CAE, we can see that these networks are identical up until the embedding layer. We illustrate this in Figure 6.1.

Stellenbosch University https://scholar.sun.ac.za

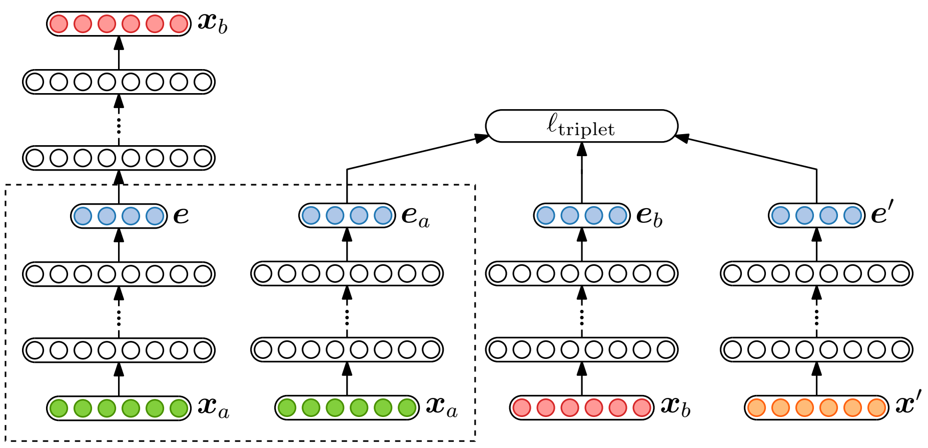

Figure 6.1: A CAE next to a Triamese network. As shown by the dotted box, a single branch of a Triamese network is identical to the encoding half of a CAE.

6.2. Hypothesis

Since a branch of the Triamese network is already half of a CAE, we hypothesise that each branch of the Triamese network can be expanded to a full CAE. We call this new hybrid network the *correspondence Triamese* (CTriamese) network. Doing so would cause two loss functions to constrain the feature embedding simultaneously: the contrastive loss from the Triamese network, and the loss function from the CAE (which is applied after it decodes the feature embedding again). The combined constraints could theoretically lead to an improved feature extractor.

6.3. The correspondence Triamese network (CTriamese)

In this section we describe the network architecture of the CTriamese network. As with the Triamese network, the CTriamese network consists of three branches. As with the CAE, each branch consists of two halves: an encoder and a decoder. In the middle of each branch, between the encoder and decoder, is our embedding layer. As with the Triamese network, this is the layer where we apply the triplet loss of the Triamese network. The normal Triamese network is trained on triplets of the form $(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}')$, where $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ represent words of the same identity (with possibly different speakers), and $\boldsymbol{x}'$ represents a word of a different identity, but it shares a speaker identity with $\boldsymbol{x}_a$. Since our input structure is the same, we can use the same input structure as the Triamese network. However, each branch is also a complete CAE, so each branch will need an output target that is the same word identity as its corresponding input, spoken by any speaker. For the first two branches of the CTriamese network, we already have a word in the Triamese input that satisfies these conditions: input $\boldsymbol{x}_a$ can use $\boldsymbol{x}_b$ as an output and $\boldsymbol{x}_b$ can use $\boldsymbol{x}_a$ as an output. However, this leaves $\boldsymbol{x}'$, which we will now refer to as $\boldsymbol{x}'_a$, without a

Stellenbosch University https://scholar.sun.ac.za

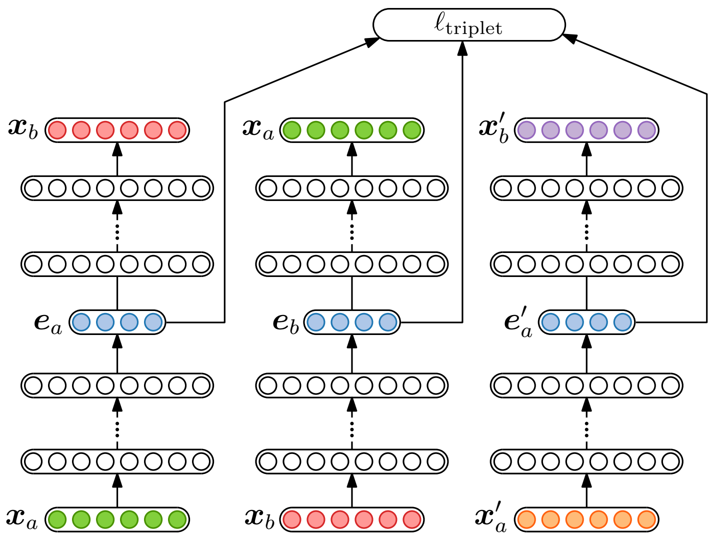

Figure 6.2: The final network architecture of the CTriamese network. Each branch of the Triamese network can be extended to a full CAE network, thus employing the constraints of both networks on the embedding layer.

corresponding output. For our fourth word, $\boldsymbol{x}'_b$ we randomly sample a word from our dataset that matches the word identity of $\boldsymbol{x}'_a$. We then DTW align it with $\boldsymbol{x}'_a$ so that the input and output frames are comparable. The final network architecture is depicted in Figure 6.2.

The loss functions of each network stay the same. The entire network uses a summation of these loss functions, making the final loss function:

$$\begin{aligned} \ell(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}'_a, \boldsymbol{x}'_b) = \ell_{\text{cae}}(\boldsymbol{x}_a, \boldsymbol{x}_b) + \ell_{\text{cae}}(\boldsymbol{x}_b, \boldsymbol{x}_a) + \ell_{\text{cae}}(\boldsymbol{x}'_a, \boldsymbol{x}'_b) \\ + \ell_{\text{triplet}}(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}'_a) \end{aligned} \tag{6.1}$$

where ℓ_{cae} and ℓ_{triplet} represent the losses of the CAE and Triamese networks respectively.

6.4. Speaker information

While the addition of speaker information did not yield significant improvements for the CAE in Chapter 5, it did show promise by improving the results on the Xitsonga dataset. Therefore, in this section, we implement speaker conditioning for the CTriamese network. Each branch of the CTriamese network is a fully functional CAE network. We can introduce speaker conditioning to each branch, just as we did for the CAE in Chapter 5, in the form of an $n \times m$ matrix. Since there are a fixed number of speakers that each CAE subnetwork will see, each CAE does not need its own speaker matrix. Instead, we will use a single speaker matrix that is shared amongst all three CAE subnetworks. We

Stellenbosch University https://scholar.sun.ac.za

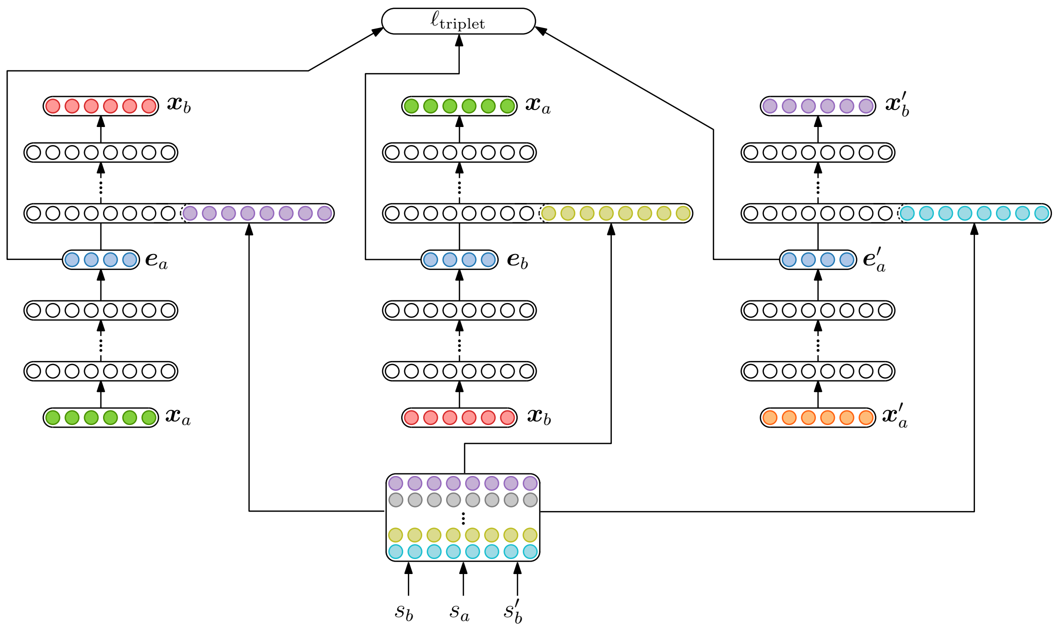

Figure 6.3: The final network architecture of the CTriamese network. Each branch of the Triamese network can be extended to a full CAE network, thus employing the constraints of both networks on the embedding layer.

will have to adjust the inputs we pass to the network during training so that the network can select a speaker embedding for each CAE subnetwork. For each output, we provide its corresponding speaker identity, so for the three outputs $(\boldsymbol{x}_b, \boldsymbol{x}_a, \boldsymbol{x}'_b)$, we provide speaker identities $(\boldsymbol{s}_b, \boldsymbol{s}_a, \boldsymbol{s}'_b)$. The speaker identities are used to select an embedding from the speaker matrix. The speaker embedding is then concatenated to the first hidden layer after the embedding layer of its corresponding CAE, giving the decoder m extra units per speaker to manipulate in order to reconstruct its target output. These units will be randomised at first but should converge to a specific speaker representation as training progresses. The resulting network with speaker information is presented in Figure 6.3.

6.5. Experiments

In this section, we explain our experimental setup, including the datasets used, the specific hidden layer setup of the model we are training, and the hyperparameters we choose. We explain our testing procedure, and we present the results of our experiment.

6.5.1. Experimental setup

The CAE branches

The three CAE branches of the CTriamese network are identical and they all share the same set of weights. Each branch starts with a 39-unit input layer, followed by six 100-unit

Stellenbosch University https://scholar.sun.ac.za

densely connected layers with ReLU activations. In the middle of the branch is the embedding layer, which is a 39-unit densely connected layer, also with ReLU activation. We keep the dimensionality 39 so that the features produced are comparable with the MFCCs we are training on. This concludes the encoder half of the CAE branch. The decoder half consists of a further six 100-unit densely connected hidden layers, each with ReLU activation. Finally, the output layer is a 39-unit densely connected layer with linear activation. Each CAE branch is trained using a mean squared error loss.

The Triamese triplet loss

We apply the contrastive triplet loss from the Triamese network to the embedding layers of the CAE branches. Our loss function remains the one proposed by Zeghidour et al. [15], which we showed in Equation 2.20. Here it is again for convenience:

$$\ell_\gamma(\boldsymbol{e}_a, \boldsymbol{e}_b, \boldsymbol{e}'_a) = \max(0, \gamma - \cos(\boldsymbol{e}_a, \boldsymbol{e}_b) + \cos(\boldsymbol{e}_a, \boldsymbol{e}'_a)). \tag{6.2}$$

Our margin parameter γ remains 0.15. The Triamese part is trained using an Adadelta [33] optimiser with a learning rate of 0.001.

Code implementation

A code implementation of the entire CTriamese network, with speaker conditioning, can be found in Appendix A.4.

6.5.2. Training procedure

The network is trained using the training set from the Buckeye corpus. We train the network 20 epochs at a time so that we can evaluate the features it produces after every step. The current model is saved after every training step. As soon as feature performance starts to decline, the previous model is reloaded and used as our feature extractor. We extract features from our test set and evaluate them using the same-different task. For our unsupervised test, no fine-tuning on epochs is allowed. We train the network in one shot for the same number of epochs that produced the best features on the English dataset. We do this for discovered and ground truth terms.

6.5.3. Results

The results for the CTriamese trained on discovered terms are shown in Tabel 6.1.

The features produced by the CTriamese network managed to score a higher average precision than the features produced by both the CAE and the Triamese networks on the same datasets. While the improvements over the standard CAE are marginal, they are consistent. Since the CTriamese network shares characteristics with the Triamese network,

Stellenbosch University https://scholar.sun.ac.za

Table 6.1: Results from the evaluation task with each network trained on UTD terms. For both AP and PRB a higher value is better.

	English validation		**English test**		**Xitsonga**	
Model	AP	PRB	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400	0.281	0.344
CAE	0.474	0.477	0.460	0.485	0.574	0.561
Triamese	0.382	0.414	0.348	0.387	0.459	0.483
CTriamese	0.493	0.490	0.464	0.482	0.575	0.564
CTriamese w/ speaker	**0.504**	**0.502**	**0.472**	**0.492**	**0.607**	**0.589**

Table 6.2: Results from the evaluation task with each network trained on ground truth terms. For both AP and PRB a higher value is better.

	English validation		**English test**	
Model	AP	PRB	AP	PRB
MFCC	0.368	0.392	0.359	0.400
CAE	0.519	0.513	0.455	0.474
Triamese	0.463	0.471	0.388	0.419
CTriamese	0.519	0.514	0.491	0.498
CTriamese /w speaker	**0.530**	**0.523**	**0.508**	**0.509**

the performance of the network can likely be further improved by using a proper sampling method for its training pairs, as proposed by [40]. Additional speaker conditioning further improved the performance of the features produced by the CTriamese network. The gains are once again small but consistent, showcasing that speaker conditioning may still be able to have an impact on the system. These results prove our hypothesis that the CAE and Triamese networks are indeed complementary and that applying a triplet loss function on intermediary representations can improve the performance of the CAE.

The results for the CTriamese trained on ground truth terms are shown in Table 6.2. The CTriamese features also perform slightly better when trained on ground truth terms, much the same as the CAE, but no major improvements as was seen with the Triamese network. The CTriamese still manages to produces features that score higher on the evaluation task than the CAE when trained on ground truth terms. The additional speaker conditioning also still provides a small but consistent boost to the AP of the extracted features.

Stellenbosch University https://scholar.sun.ac.za

6.6. Chapter summary and conclusions

In this chapter, we showed how the CAE and Triamese networks could be combined to produce a hybrid network that can extract speaker-independent features from audio data that performs better than those produced by the other networks on the same-different task. We also showed that additional speaker conditioning slightly improves the performance of the features produced by the CTriamese network. Much like the CAE, the CTriamese-produced features also perform better when trained on ground truth terms, but it does not see improvements like those produced by the Triamese network did. This could be due to the dominance of the CAE in the loss function. Future work should include an experiment where weights are added to the loss functions to measure how it impacts the performance of the network.

Stellenbosch University https://scholar.sun.ac.za

Chapter 7

Summary and conclusion

In this thesis, we compared the CAE and Triamese networks. We investigated the possibility of using speaker conditioning as a means to improve the speaker-independence of features produced by the CAE. Finally, we showed how the CAE and Triamese networks could be combined to produce a new network, which we called the CTriamese network. In this chapter, we summarise our findings, give an overview of our main contributions, and discuss what future work could entail.

7.1. Comparison of the CAE and Triamese networks

We trained both the CAE and the Triamese network using the Buckey corpus of conversational English and the Xitsonga language from the NCHLT dataset. Both networks were trained frame by frame on terms produced by an unsupervised term discovery system proposed by [26]. Then, using each network as a feature extractor, we evaluated the feature set each network produced using the same-different task: a word discrimination task that also measures speaker-independence. We showed that, under these conditions, the CAE produced features that scored a higher average precision than the features produced by the Triamese network on both the English and the Xitsonga datasets (0.460 vs 0.348 and 0.574 vs 0.459 respectively). The Xitsonga dataset represented a completely unsupervised scenario on which no fine-tuning was performed.

We also evaluated each network on ground truth terms — terms that correspond to actual words - rather than discovered terms. We found the Triamese network produced significantly better features when trained on ground truth terms, leading us to believe that the Triamese network could be sensitive to the data on which it is trained. This corresponds with the work done by Riad et al. [40], in which they found that the performance of Siamese networks can be affected by the selection of training pairs. We believe this holds for derivatives of the Siamese network, like the Triamese network, but a comparison of Triamese networks trained on various selected training pairs would have to be performed.

Stellenbosch University https://scholar.sun.ac.za

7.2. Speaker conditioning in the CAE

The CAE network was modified to learn representations of each speaker in the training data during training. These representations were then fed back into the decoder half of the CAE. The hypothesis is that the extra information would allow the decoder to reconstruct the output word better, and in turn, allow the encoder to focus less on features that represent speaker information and focus more on features that represent word identity. The speaker conditioning did not affect the CAE in a consistent capacity. In the evaluation on the English and Xitsonga test sets, speaker conditioning did provide a minor increase in average precision (0.464 vs 0.460 and 0.610 vs 0.574 respectively), but when trained on ground truth terms the CAE performed better without the speaker information (0.455 vs 0.445). This leads us to believe that speaker conditioning can still be used to better the features produced by a network.

7.3. The CTriamese network

We developed a novel CAE-Triamese hybrid network called the CTriamese network. The network applies the triplet loss of the Triamese network to the intermediary representations of the CAE. We evaluated the network on the same English and Xitsonga datasets as the CAE and Triamese networks using the same same-different evaluation task. Using this hybrid network, we showed that the combined constraints of the CAE and Triamese networks lead to better features than the individual CAE and Triamese networks could produce. We also applied speaker conditioning to this network and found that it produced small but consistent gains in the average precision of the features extracted by the network.

7.4. Contributions

Our main contributions in this thesis were as follows:

- To our knowledge, this is the first time that the CAE and Triamese networks were compared when trained on the same data using the same evaluation task.
- We investigated the possibility of using speaker conditioning in the CAE as a means to produce features that are less speaker-dependent.
- We proposed a novel CAE-Triamese hybrid network by applying a triplet loss to the intermediary representations of the CAE. To our knowledge, this is the first time that network performance has been increased in this manner.

Stellenbosch University https://scholar.sun.ac.za

7.5. Future work

Future work involving comparisons between the CAE and Triamese networks should consider using one of the sampling techniques proposed by [40], as training pair selection can affect the performance of the Triamese network. While the two networks were measured on equal ground here, the performance of the Triamese network might change once the pairs are sampled by one of the proposed techniques. It might also be worth exploring the impact that a different triplet loss could have on the performance of the Triamese and CTriamese networks. One loss to consider is the semihard online triplet loss [41]. The CTriamese network showed that speaker conditioning can affect the performance of features produced by the network. It is worth investigating if the decoder of the CAE could be restructured to make better use of the learned speaker representations. Work extending the CTriamese network should consider adding weights to the combined CTriamese loss function. Equation 6.1 can be adjusted to weigh each part of the CTriamese network's loss function differently:

$$\begin{aligned}\ell(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}'_a, \boldsymbol{x}'_b) = \alpha_1 \ell_{\text{cae}}(\boldsymbol{x}_a, \boldsymbol{x}_b) + \alpha_2 \ell_{\text{cae}}(\boldsymbol{x}_b, \boldsymbol{x}_a) + \alpha_3 \ell_{\text{cae}}(\boldsymbol{x}'_a, \boldsymbol{x}'_b) \\ + \alpha_4 \ell_{\text{triplet}}(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}'_a), \qquad (7.1)\end{aligned}$$

where α is a weight parameter. By weighting the loss function in favour of the CAE or the triplet loss, the performance of the network could be adjusted.

Another consideration could be to train the networks on features other than MFCCs. Perceptual linear prediction [42] (PLP) is a feature extraction approach that can be considered to produce baseline features other than MFCCs.

Each network should also be evaluated using a variety of different optimisers, as the choice of optimiser can greatly affect the performance of each network.

An alternative to the CTriamese network would be to train the Triamese and CAE networks separately, and then produce features by having the MFCCs be processed first by one network and then the other. A comparison between this approach and the CTriamese network should be made.

Another experiment that can be considered is to train a network with a reduced speaker set and gradually increase the number of speakers in the training set as training progresses, mimicking the training environment of infants.

Stellenbosch University https://scholar.sun.ac.za

Bibliography

[1] R. Menon, H. Kamper, E. Van Der Westhuizen, J. Quinn, and T. Niesler, "Feature exploration for almost zero-resource ASR-free keyword spotting using a multilingual bottleneck extractor and correspondence autoencoders," in *Proc. Interspeech*, 2019.

[2] M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky, "Rapid evaluation of speech representations for spoken term discovery," in *Proc. Interspeech*, 2011.

[3] P. K. Austin and J. Sallabank, *The Cambridge Handbook of Endangered Languages*, ser. Cambridge Handbooks in Language and Linguistics. Cambridge: Cambridge University Press, 2011.

[4] D. M. Eberhard, G. F. Simons, and C. D. Fennig, *Ethnologue: Languages of the World*, 22nd ed. Dallas: SIL International, 2019.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *Proc. ICASSP*, 2015.

[7] E. Dupoux, "Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner," *Cognition*, vol. 173, pp. 43–59, 2018.

[8] G. Adda, S. Stüker, M. Adda-Decker, O. Ambouroue, L. Besacier, D. Blachon, H. Bonneau-Maynard, P. Godard, F. Hamlaoui, D. Idiatov, G. N. Kouarata, L. Lamel, E.-M. Makasso, A. Rialland, M. Velde, F. Yvon, and S. Zerbian, "Breaking the unwritten language barrier: The BULB project," *Procedia Computer Science*, vol. 81, pp. 8–14, 2016.

[9] M. Versteegh, R. Thiollière, T. Schatz, X.-N. Cao Kam, X. Anguera, A. Jansen, and E. Dupoux, "The Zero Resource Speech Challenge 2015," in *Proc. Interspeech*, 2015.

[10] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The Zero Resource Speech Challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67 – 72, 2016.

[11] K. L. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *Proc. ICASSP*, 2015.

Stellenbosch University https://scholar.sun.ac.za

[12] Y. Zhang and J. R. Glass, “Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams,” in *Proc. ASRU*, 2009.

[13] S. Settle, K. Levin, H. Kamper, and K. Livescu, “Query-by-example search with discriminative neural acoustic word embeddings,” in *Proc. Interspeech*, 2017.

[14] Y.-H. Wang, H.-y. Lee, and L.-s. Lee, “Segmental audio word2vec: Representing utterances as sequences of vectors with applications in spoken term detection,” in *Proc. ICASSP*, 2018.

[15] N. Zeghidour, G. Synnaeve, N. Usunier, and E. Dupoux, “Joint learning of speaker and phonetic similarities with siamese networks,” in *Proc. Interspeech*, 2016.

[16] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” *arXiv preprint arXiv:1703.00395*, 2017.

[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[18] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a ‘Siamese’ time delay neural network,” in *Proc. NIPS*, 1993.

[19] T. Schatz, V. Peddinti, F. Bach, A. Jansen, H. Hermansky, and E. Dupoux, “Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline,” in *Proc. Interspeech*, 2013.

[20] M. A. Pitt, K. Johnson, E. Hume, S. Kiesling, and W. Raymond, “The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability,” *Speech Communication*, vol. 54, no. 1, pp. 89–95, 2005.

[21] N. J. De Vries, M. H. Davel, J. Badenhorst, W. D. Basson, F. De Wet, E. Barnard, and A. De Waal, “A smartphone-based ASR data collection tool for under-resourced languages,” *Speech Communication*, vol. 56, pp. 119–131, 2014.

[22] J. Glass, “Towards unsupervised speech processing,” in *Proc. ISSPA*, 2012.

[23] D. Crystal, *Language Death*, ser. Canto Refresh Your Series. Cambridge University Press, 2002.

[24] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

Stellenbosch University https://scholar.sun.ac.za

[25] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.

[26] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. ASRU*, 2011.

[27] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Computer Speech & Language*, pp. 154–174, 2017.

[28] M. Sun and H. Van hamme, "Joint training of non-negative tucker decomposition and discrete density hidden markov models," *Computer Speech & Language*, vol. 27, pp. 969–988, 2013.

[29] K. M. Ting, *Precision and Recall*. Boston, MA: Springer US, 2010, pp. 781–781.

[30] W. A. Munson and M. B. Gardner, "Standardizing auditory tests," *The Journal of the Acoustical Society of America*, vol. 22, no. 5, pp. 675–675, 1950.

[31] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[32] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[33] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.

[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010.

[36] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013.

[37] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[38] R. Thiollière, E. Dunbar, G. Synnaeve, M. Versteegh, and E. Dupoux, "A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling," in *Proc. Interspeech*, 2015.

Stellenbosch University https://scholar.sun.ac.za

[39] D. Renshaw, H. Kamper, A. Jansen, and S. J. Goldwater, “A comparison of neural network methods for unsupervised representation learning on the Zero Resource Speech Challenge,” in *Proc. Interspeech*, 2015.

[40] R. Riad, C. Dancette, J. Karadayi, N. Zeghidour, T. Schatz, and E. Dupoux, “Sampling strategies in siamese networks for unsupervised speech representation learning,” in *Proc. Interspeech*, 2018.

[41] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. CVPR*, 2015.

[42] H. Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *The Journal of the Acoustical Society of America*, vol. 87, 1998.

Stellenbosch University https://scholar.sun.ac.za

Appendix A

Network implementations in Python

This appendix illustrates how each network was implemented in code using Python.

A.1. CAE

```python
from tensorflow import keras
from tensorflow.keras.models import Model, Sequential

def build_cae(n_visible, n_hiddens, n_embedding):
    """
    Builds a correspondence autoencoder.
    Args:
        n_visible: The input and output dimensions of the network.
        n_hiddens: List of hidden layer dimensions.
        n_embedding: Dimensionality of the feature embedding.

    Returns:
        cae: Keras model ready to be compiled.
        seq_encode: Encoder half of the network, to be used as
                    feature extractor after training.
    """
    input_layer = keras.layers.Input(shape=(n_visible,))

    seq_encode = Sequential()
    for i, n_hidden in enumerate(n_hiddens):
        seq_encode.add(keras.layers.Dense(n_hidden,
            activation=keras.activations.relu))
    seq_encode.add(keras.layers.Dense(n_embedding,
        activation=keras.activations.relu))

    embedding = seq_encode(input_layer)
```

Stellenbosch University https://scholar.sun.ac.za

```

    seq_decode = Sequential()
    for i, n_hidden in enumerate(reversed(n_hiddens)):
        seq_decode.add(keras.layers.Dense(n_hidden,
            activation=keras.activations.relu))
    seq_decode.add(keras.layers.Dense(n_visible,
        activation=keras.activations.linear))

    output_layer = seq_decode(embedding)

    cae = keras.models.Model(inputs=[input_layer],
                             outputs=[output_layer])

    return cae, seq_encode
```

A.2. Triamese

```
from tensorflow import keras
from tensorflow.keras.models import Model, Sequential

def build_triamese(n_visible, n_hiddens, n_embedding):
    """
    Builds a Triamese network.
    Args:
        n_visible: The input and output dimensions of the network.
        n_hiddens: List of hidden layer dimensions.
        n_embedding: Dimensionality of the feature embedding.

    Returns:
        triamese: Keras model ready to be compiled.
        seq_encode: One branch of the network, to be used as
                    feature extractor after training.
    """
    input_1 = keras.layers.Input(shape=(n_visible,))
    input_2 = keras.layers.Input(shape=(n_visible,))
    input_3 = keras.layers.Input(shape=(n_visible,))

```

Stellenbosch University https://scholar.sun.ac.za

```
    seq_encode = Sequential()
    for i, n_hidden in enumerate(n_hiddens):
        seq_encode.add(keras.layers.Dense(n_hidden,
            activation=keras.activations.relu))
    seq_encode.add(keras.layers.Dense(39,
        activation=keras.activations.relu))

    embedding_1 = seq_encode(input_1)
    embedding_2 = seq_encode(input_2)
    embedding_3 = seq_encode(input_3)

    d1 = keras.layers.dot([embedding_1, embedding_2],
                          axes=-1, normalize=True)
    d2 = keras.layers.dot([embedding_1, embedding_3],
                          axes=-1, normalize=True)

    output_layer = keras.layers.Concatenate()([d1, d2])

    triamese = keras.models.Model(inputs=[input_1, input_2, input_3],
                                  outputs=[output_layer])

    return triamese, seq_encode

def margin_triplet_loss(margin):
    def loss_func(y_true, y_pred):
        d1 = y_pred[0][0]
        d2 = y_pred[0][1]

        loss = keras.backend.maximum(margin - d1 + d2,
                                     0.0)
        return loss
    return loss_func
```

A.3. CAE with speaker conditioning

```
from tensorflow import keras
from tensorflow.keras.models import Model, Sequential
```

Stellenbosch University https://scholar.sun.ac.za

```python

def build_cae_speaker(n_visible, n_hiddens, n_embedding, n_speakers):
    """
    Builds a correspondence autoencoder with speaker conditioning.
    Args:
        n_visible: The input and output dimensions of the network.
        n_hiddens: List of hidden layer dimensions.
        n_embedding: Dimensionality of the feature embedding.
        n_speakers: Number of speakers in the training set.

    Returns:
        cae_with_speaker: Keras model ready to be compiled.
        seq_encode: Encoder half of the network, to be used as
                    feature extractor after training.
    """
    input_layer = keras.layers.Input(shape=(n_visible,))
    speaker_input_layer = keras.layers.Input(shape=(1,),
                                             dtype='int32')
    speaker_weights = np.random.rand(n_speakers, 100)

    speaker_embeddings = keras.layers.Embedding(
        input_length=1, input_dim=n_speakers, output_dim=100,
        trainable=True, weights=[embedding_weights])
    speaker_embedding = keras.layers.Flatten()(
        speaker_embeddings(speaker_input_layer))

    seq_encode = Sequential()
    for i, n_hidden in enumerate(n_hiddens):
        seq_encode.add(keras.layers.Dense(n_hidden,
            activation=keras.activations.relu))
    seq_encode.add(keras.layers.Dense(n_embedding,
        activation=keras.activations.relu))

    embedding = seq_encode(input_layer)

    seq_speaker = Sequential()
    seq_speaker.add(keras.layers.Dense(reversed(n_hiddens)[0],
        activation=keras.activations.relu))
    first_layer = seq_speaker(embedding)
```

Stellenbosch University https://scholar.sun.ac.za

```
    embedding_with_speaker = keras.layers.concatenate(
        [embedding, speaker_embedding])

    seq_decode = Sequential()
    for i, n_hidden in enumerate(reversed(n_hiddens)):
        if i != 0:
            seq_decode.add(keras.layers.Dense(n_hidden,
                activation=keras.activations.relu))
    seq_decode.add(keras.layers.Dense(n_visible,
        activation=keras.activations.linear))

    output_layer = seq_decode(embedding_with_speaker)

    cae_with_speaker = keras.models.Model(inputs=[input_layer],
                                          outputs=[output_layer])

    return cae_with_speaker, seq_encode
```

A.4. CTriamese with speaker conditioning

```
from tensorflow import keras
from tensorflow.keras.models import Model, Sequential

def build_ctriamese_with_speaker(n_visible, n_hiddens, n_embedding, n_speakers):
    """
    Builds a correspondence autoencoder with speaker conditioning.
    Args:
        n_visible: The input and output dimensions of the network.
        n_hiddens: List of hidden layer dimensions.
        n_embedding: Dimensionality of the feature embedding.
        n_speakers: Number of speakers in the training set.

    Returns:
        ctriamese_with_speaker: Keras model ready to be compiled.
        seq_encode: Encoder half of the network, to be used as
                    feature extractor after training.
    """
```

Stellenbosch University https://scholar.sun.ac.za

```
    input_1 = keras.layers.Input(shape=(n_visible,))
    input_2 = keras.layers.Input(shape=(n_visible,))
    input_3 = keras.layers.Input(shape=(n_visible,))
    speaker_input_1 = keras.layers.Input(shape=(1,), dtype='int32')
    speaker_input_2 = keras.layers.Input(shape=(1,), dtype='int32')
    speaker_input_3 = keras.layers.Input(shape=(1,), dtype='int32')
    speaker_weights = np.random.rand(n_speakers, 100)

    speaker_embeddings = keras.layers.Embedding(
        input_length=1, input_dim=n_speakers, output_dim=100,
        trainable=True, weights=[embedding_weights])

    speaker_embedding_1 = keras.layers.Flatten()(
        speaker_embeddings(speaker_input_1))
    speaker_embedding_2 = keras.layers.Flatten()(
        speaker_embeddings(speaker_input_2))
    speaker_embedding_3 = keras.layers.Flatten()(
        speaker_embeddings(speaker_input_3))

    seq_encode = Sequential()
    for i, n_hidden in enumerate(n_hiddens):
        seq_encode.add(keras.layers.Dense(n_hidden,
            activation=keras.activations.relu))
    seq_encode.add(keras.layers.Dense(n_embedding,
        activation=keras.activations.relu))

    embedding_1 = seq_encode(input_1)
    embedding_2 = seq_encode(input_2)
    embedding_3 = seq_encode(input_3)

    seq_speaker = Sequential()
    seq_speaker.add(keras.layers.Dense(reversed(n_hiddens)[0],
        activation=keras.activations.relu))
    first_layer_1 = seq_speaker(embedding_1)
    first_layer_2 = seq_speaker(embedding_2)
    first_layer_3 = seq_speaker(embedding_3)
    embedding_with_speaker_1 = keras.layers.concatenate(
        [first_layer_1, speaker_embedding_1])
    embedding_with_speaker_2 = keras.layers.concatenate(
```

Stellenbosch University https://scholar.sun.ac.za

```
        [first_layer_2, speaker_embedding_2])
    embedding_with_speaker_3 = keras.layers.concatenate(
        [first_layer_3, speaker_embedding_3])

    seq_decode = Sequential()
    for i, n_hidden in enumerate(reversed(n_hiddens)):
        if i != 0:
            seq_decode.add(keras.layers.Dense(n_hidden,
                activation=keras.activations.relu))
    seq_decode.add(keras.layers.Dense(n_visible,
        activation=keras.activations.linear))

    output_1 = seq_decode(embedding_with_speaker_1)
    output_2 = seq_decode(embedding_with_speaker_2)
    output_3 = seq_decode(embedding_with_speaker_3)

    d1 = keras.layers.dot([embedding_1, embedding_2],
                          axes=-1, normalize=True)
    d2 = keras.layers.dot([embedding_1, embedding_3],
                          axes=-1, normalize=True)

    output_layer = keras.layers.Concatenate()([d1, d2])

    ctriamese_with_speaker = keras.models.Model(
        inputs=[input_1, input_2, input_3,
                speaker_input_1, speaker_input_2, speaker_input_3],
        outputs=[output_layer, output_1, output_2, output_3])

    return ctriamese_with_speaker, seq_encode
```