THE UNIVERSITY *of* EDINBURGH

Representation learning for unsupervised speech processing

Daniel Renshaw

Master of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2016

Abstract

Automatic speech recognition for our most widely used languages has recently seen substantial improvements, driven by improved training procedures for deep artificial neural networks, cost-effective availability of computational power at large scale, and, crucially, availability of large quantities of labelled training data. This success cannot be transferred to low and zero resource languages where the requisite transcriptions are unavailable.

Unsupervised speech processing promises better methods for dealing with under-resourced languages. Here we investigate unsupervised neural network based models for learning frame- and sequence- level representations with the goal of improving zero-resource speech processing. Good representations eliminate differences in accent, gender, channel characteristics, and other factors to model subword or whole-term units for within- and across- speaker speech unit discrimination.

We present two contributions focussing on unsupervised learning of frame-level representations: (1) an improved version of the correspondence autoencoder applied to the INTERSPEECH 2015 Zero Resource Challenge, and (2) a proposed model for learning representations that explicitly optimize speech unit discrimination.

We also present two contributions focussing on efficiency and scalability of unsupervised speech processing: (1) a proposed model and pilot experiments for learning a linear-time approximation of the quadratic-time dynamic time warping algorithm, and (2) a series of model proposals for learning fixed size representations of variable length speech segments enabling efficient vector space similarity measures.

Acknowledgements

I would like to thank my supervisors Dr. Sharon Goldwater, Dr. Charles Sutton, and Prof. Mirella Lapata for their support and guidance. I especially thank Dr Goldwater for offering to supervise a project that would enable me to explore my passion for neural networks.

My interest in artificial neural networks and speech processing was sparked while interning at Google Research in New York. I would like to thank everybody I worked with there and especially Keith Hall for his advice and for introducing me to the joys of collaborative research.

I enjoyed working with Herman Kamper and Aren Jansen on our ZeroSpeech challenge entry and thank Herman especially for helping me get up to speed with unsupervised speech processing. I am also grateful to Krzysztof Jerzy Geras for helpful discussions about autoencoder methods.

The past eight years have been, at times, a challenging and distracting diversion into undergraduate education and postgraduate research and I would like to thank my family for their unflagging support throughout this whole period.

This research was supported by a Google European Doctoral Fellowship.

Declaration

I declare that this thesis has been composed by myself and that the work has not been submitted for any other degree or professional qualification. I confirm that the work contained herein is my own except where explicitly stated otherwise in the text, and except where work which has formed part of jointly-authored publications has been included. My contributions and those of the other authors to this work have been explicitly indicated below. I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others.

The work presented in Chapter 3 and portions of Chapter 2 was previously published in *The proceedings of INTERSPEECH 2015* as *A Comparison of Neural Network Methods for Unsupervised Representation Learning on the Zero Resource Speech Challenge* by *Daniel Renshaw*[*] (myself)*, Herman Kamper*[†]*, Aren Jansen*[‡]*, and Sharon Goldwater*[*] (my supervisor). This study was conceived by all of the authors. I designed and implemented the autoencoder-based models, carried out all of the *ABX* experiments, and authored the bulk of the paper. Aren Jansen trained the DNN and extracted the bottleneck features. Some of the material has been edited or rewritten for the purposes of integrating it into this thesis.

(*author*)

[*,†]ILCC and [†]CSTR, School of Informatics, University of Edinburgh, UK
[‡]HLTCOE and CLSP, Johns Hopkins University, USA

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The year is 2166 and the world is in awe. The first indisputable signs of sentient alien life have been detected. Artificial radio transmissions are being received by a probe sent to flyby a rocky planet we discovered in the habitable zone of a nearby star. The signals do not appear to be intentional efforts to communicate with unknown recipients across interstellar space. Instead, they are a mixture of frequency modulated radio transmissions much like those we created in the 20^{th} century. The content includes a lot of what sounds like natural speech in a variety of languages. The probe is rebroadcasting the signals back to Earth and you are tasked with learning the spoken languages.

This fiction emphasises the difficulty of unsupervised speech recognition. In the story, the aliens are not generating their transmissions with the intent of teaching us their languages, we cannot interact with the aliens to solicit guidance, and we do not have additional modalities with which the speech might correspond. Most notably we cannot *see*, nor *read* about, what is being referred to in what we *hear*.

Human language learning is not usually so extreme. We interact with language experts (e.g. our parents) and receive additional sensory inputs that co-occur with the speech sounds. Co-occurrence strengthens our mental models either by explicit supervised guidance (e.g. "look, a cat!") or implicit unsupervised pattern detection.

The most extreme case of unimodal fully unsupervised spoken language learning, like the alien radio signals scenario, has applications in three practical research areas:

- **Automatic speech recognition (ASR) of under-resourced languages**. There are many thousands of languages in use across the world but only a handful have adequate resources available for the development of usefully accurate supervised ASR systems. Unsupervised ASR offers an approach that does not require the costly and often infeasible collection of resources needed for supervised ASR.

- **Conventional supervised ASR**. Even well-resourced languages have more unlabelled data than labelled data. Methods yielding improved representations for unsupervised ASR can sometimes be applied usefully in supervised ASR pre-training because they can be trained on larger quantities of unlabelled data.

- **Computational models of human infant language learning**. Human infants learn language by immersion in a language speaking environment with minimal explicit supervision. Unsupervised speech processing research can contribute to the study of this process by offering cognitively plausible computational models.

Our work is motivated primarily by the first and second of these three areas. We make no claims regarding the cognitive plausibility of our models.

We assume spoken language utterances are composed of sequences of discrete abstract units (perhaps phonemes, morphemes, or whole terms) whose representative sounds run together when spoken, such that their boundaries are unclear. Following the conventional paradigm for unsupervised language learning, the first problem to solve is that of segmentation – identifying the most likely boundaries between the discrete units in the utterances we hear. Given boundaries, we can cluster the segmented units based on their similarity. We are assuming here that two units that *sound* the same are likely to *be* the same, at least when in similar contexts (the distributional hypothesis). The two tasks often go hand-in-hand – improved clusters can be found given an improved segmentation and improved segments can be found given an improved clustering.

Unit segmentation and clustering both depend on an ability to measure the similarity between arbitrary segments of speech signals. We want to find unit boundaries that maximize the similarity of units that are in fact the same, so they get clustered together, and minimize the similarity of units that are in fact different, so they do not get clustered together. The concept of clustering is meaningless without some notion of similarity.

Similarity is a function of representation. The degree of similarity between two speech signal segments is related to the degree of similarity between their representations. A speech signal representation might be analogue (e.g. the deviation of a groove in the wax covered surface of a phonograph cylinder) but we are more interested in digital representations, which are easier to analyse. The characteristics of a digital representation depend on the task for which it is designed. General purpose digital representations of sound signals are designed to retain the information needed to reproduce the original sound signal with high fidelity. We are not interested in reproduction of this kind because there are many aspects of a raw speech signal that are unimportant in the

identification of the abstract units we care about. We wish to ignore the confounding effects of, for example, ambient noise, speaker-/gender-/accent-specific variations, or the quality of the original recording.

Our ultimate goal is to find representations that (1) include all the unit-discriminative information we care about, (2) include none of the information we do not care about, (3) use an encoding enabling efficient similarity comparisons, and (4) can be learnt from (potentially large quantities of) unlabelled data. In this work we focus on applying unsupervised machine learning techniques to automatically discover representations of individual speech frames, and larger segments of speech, that better meet these requirements than previous methods.

1.1 Contributions

- *A method for learning better representations of individual speech frames using an improved version of the correspondence autoencoder (cAE).* Prior versions of the cAE used a shallow linear decoder; we present a version using a deep non-linear decoder. We perform subword unit discrimination experiments, comparing the new variant of the cAE with the previous variant and other baselines in the context of the INTERSPEECH 2015 Zero Resource Speech Challenge.

- *A novel model for learning speech frame representations by optimizing the objective of interest, dynamic time warping (DTW), explicitly.* DTW is commonly used as a measure of similarity when comparing two speech segments but representations have not previously been learnt to optimize this measure directly. We hypothesize that optimizing speech frame representations such that the DTW distance between same-class sequences is less than the DTW distance between different-class sequences will yield a better quality frame representation function.

- *A novel model that learns to approximate the DTW function enabling sequence comparisons in linear time instead of quadratic time.* The time complexity of DTW is proportional to the *product* of the lengths of the sequences being compared. We propose a model that learns to approximate DTW such that comparisons have a time complexity proportional to the *sum* of the sequence lengths. We hypothesize that the resulting approximate alignments and similarity measurements can be usefully accurate. Our method generalises to approximate linear time multi-sequence alignment, a task important in the field of bioinformatics.

- *An exploration of standard and novel models for learning fixed size representations of variable length speech segments*. Sequence similarity computations normally require an algorithm with time complexity that is a function of the sequence lengths, e.g. DTW. Fixed size representations allow sequences to be compared using a single vector space computation, e.g. cosine distance. We show how standard models can be used for this purpose, describe their weaknesses, and detail novel models that we hypothesize address those weaknesses.

Our first principal contribution is supported by experimental results and published in the proceedings of INTERSPEECH 2015, the 15th Annual Conference of the International Speech Communication Association. Our three other principal contributions comprise: (1) an analysis of a problem related to unsupervised speech processing, (2) a review of how existing methods are, or could be, applied to solving the problem and their weaknesses, (3) detailed definitions of one or more novel methods that, we hypothesize, solve the problem better than prior methods, and (4) fully worked out mathematics for non-standard elements. The aim of these final three contributions is to support future research by enabling others to quickly implement and experiment with our ideas. We present results from a trial experiment in relation to one of the final three contributions; no experimental results are presented in relation to the other two.

1.2 Thesis structure

Chapter 2 introduces the background material needed for fully understanding the material in Chapters 3 through 5.

Chapter 3 presents our enhanced version of the correspondence autoencoder and experimental results for learning representations of individual speech frames within the context of the INTERSPEECH 2015 Zero Resource Speech Challenge. Chapter 4 covers our two DTW-oriented contributions: learning representations by optimizing DTW explicitly, and approximating DTW with a linear time artificial neural network-based algorithm. Chapter 5 extends the work presented in Chapter 3 to sequence representation learning, where we attempt to avoid the bottleneck problems caused by DTW by learning fixed size representations of whole sequences.

We wrap up in Chapter 6 with a summary of our contributions and present an agenda for future research.

1.3 Notation

We use standard notational conventions throughout, summarised here for convenience.

- $\mathcal{X}$ or $\{\dots\}$: a set; $\mathbb{R}$: the conventional set of all real numbers.
- $x \in \mathcal{X}$: a scalar from domain $\mathcal{X}$ with size $|x|$ (where applicable).
- $\mathbf{x} \in \mathcal{X}^D$: a column vector of size $|\mathbf{x}| = D$ containing scalar elements $x_i \in \mathcal{X}$ at all positions $i \in [1,\dots,D]$.
- $\mathbf{X} \in \mathcal{X}^{M \times N}$: a matrix with M rows and N columns containing scalar elements $x_{i,j} \in \mathcal{X}$ at all rows $i \in [1,\dots,M]$ and all columns $j \in [1,\dots,N]$.
- $\mathbf{x}_{1:L}$: A sequence of length L containing vector elements $\mathbf{x}_t \in \mathcal{X}^D$ at all positions $t \in [1,\dots,L]$.
- $\mathbf{X}\mathbf{y}$, $\mathbf{x}\mathbf{y}$: the dot product of matrix $\mathbf{X}$, or vector $\mathbf{x}$, and vector $\mathbf{y}$.
- $\mathbf{x} \odot \mathbf{y}$: the element-wise product of vector $\mathbf{x}$ and vector $\mathbf{y}$.
- $[\mathbf{x};\mathbf{y}]$: the concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$; if the sizes of the input vectors are M and N then their concatenation is a column vector of size $M+N$.
- $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{|\mathbf{x}|} |x|_i^p\right)^{\frac{1}{p}}$: the p-norm of vector $\mathbf{x}$.
- $D^{name}(\mathbf{x},\mathbf{y})$: a distance function between vectors $\mathbf{x}$ and $\mathbf{y}$; $D^{name}(\mathbf{x}_{1:L^x},\mathbf{y}_{1:L^y})$: a distance function between sequences $\mathbf{x}_{1:L^x}$ and $\mathbf{y}_{1:L^y}$; the nature of a distance function is determined by its type, indicated by the superscript *name*.
- $\mathbf{W}^{name}$, $\mathbf{b}^{name}$: a weight matrix and bias vector; the *name* superscript indicates an artificial neural network layer to which these parameters belong.
- Δ^n: the n-simplex, i.e. $\mathbf{x} \in \Delta^n \Leftrightarrow \mathbf{x} \in [0,1]^{n+1}$ *and* $\sum_{i=1}^{n+1} x_i = 1$; that is, $\mathbf{x}$ is a valid set of parameters for a categorical probability distribution over $n+1$ outcomes.
- $\mathbf{0}$, $\mathbf{1}$: a vector or matrix of zeros or ones, respectively; the shape can be determined from the context.
- $\mathbf{I}$: the identity matrix; the shape can be determined from the context.
- $\mathbf{X}'$: transpose of matrix $\mathbf{X}$.

Chapter 2

Background

This chapter provides an overview of representation learning for unsupervised speech processing. Relevant prior work is reviewed and the key concepts and technologies on which this work is based are detailed. Section 2.1 provides a brief overview of relevant prior work and introduces the most important algorithm on which unsupervised speech processing is based: dynamic time warping (DTW). The methods by which speech signals can be encoded into representations suitable for processing is introduced in Section 2.2 along with an overview of prior work in representation learning for speech signals. Section 2.3 describes the two tasks we use for evaluation: *same-different* and *ABX* and Section 2.4 introduces the foundational models on which our contributions are principally built: autoencoders and recurrent neural networks.

2.1 Unsupervised speech processing

Automatic speech recognition (ASR) is a speech-to-text process – the goal is to automatically produce the best possible transcriptions of spoken utterances. State-of-the-art ASR is supervised – the training data labels define a mapping between spoken inputs and the corresponding, correct, textual outputs. We are interested in the unsupervised case where labelled training data is unavailable and the correspondence between utterance inputs and textual outputs must be inferred by indirect methods, e.g. frequency analysis and language modelling.

Conventional ASR can be decomposed into five major components [32]:

1. The raw speech signal is transformed into an easier to process representation.

2. An acoustic model determines the probability of the speech signal representation

given a hypothesised word sequence.

3. A lexical model specifies how each entry in a fixed dictionary of words decompose into subword units.

4. A language model determines the probability of a hypothesised word sequence.

5. A search procedure uses the other components to find the most probable word sequence given the raw speech signal.

The first three components usually involve some form of supervision or expert knowledge and, in principle, each can be replaced with an unsupervised mechanism. For example, the lexical model is typically specified by a linguist but this can be learnt from data whilst continuing to train the acoustic model on transcribed utterances. In [32], James Glass describes a spectrum of scenarios which vary in the degree to which they require, and use, supervision. State-of-the-art ASR is firmly at one end – the expert-based, fully supervised, scenario. We are interested in techniques most relevant to Glass's "major break from conventional ASR training" – the decipher-based scenario where only unannotated speech and non-parallel text are available. One might think of this scenario as learning to "listen" and "read" concurrently, from scratch, without a teacher identifying correspondences between speech and text. Our work only tackles the "listening" aspect, i.e. speech signal representation and acoustic modelling.

Semi-supervised methods, e.g. [53, 28, 65, 92], enable practical speech processing when only small quantities of labelled data are available. We do not address semi-supervised learning explicitly in our work but some of our methods may be useful in semi-supervised settings, for example, when an unsupervised pre-training stage is used.

An important fully unsupervised speech processing task, and one that is often used upstream of our methods, is unsupervised term discovery (UTD). The task, introduced by Park and Glass [67], is to automatically discover clusters of speech segments that occur frequently throughout the corpus, such that the segments within a cluster are similar to each other and dissimilar to segments in other clusters. We expect a cluster to form for each term that is commonly spoken in the corpus. The original approach of Park and Glass combines segmental-DTW (a variation of the standard dynamic time warping algorithm introduced below), to identify the frequently repeating patterns, with a graph clustering algorithm. Later work has improved both segmental-DTW, e.g. [46, 103, 47], and graph clustering, e.g. [10, 19, 71, 79], with particular emphasis on scalability, enabling processing of datasets containing 100s of hours of speech.

The study of Lyzinski et al. evaluated these graph clustering methods in the context of unsupervised spoken term discovery [58]. Other uses of unsupervised speech processing include query-by-example keyword spotting (e.g. [102, 59, 16]) and spoken document indexing and retrieval (e.g. [31, 104]).

Unsupervised speech processing development is following a similar trajectory to that of supervised speech processing [100, page 5, Figure 1.4]. Initial research is necessarily focussed on the simplest task – isolated read words with a small vocabulary in a single language. The task is made gradually harder by introducing a greater variety of speakers, a larger vocabulary, allowing continuous speech where words are no longer isolated, and moving towards conversational speech instead of directed speech. We focus on a mid-point: continuous conversational speech from multiple speakers where the recording is high quality/low noise.

The difficulty of tackling the full ASR task in an unsupervised setting is illustrated by the TIDIGITS corpus (continuous small-vocabulary directed speech) and related standard recognition task. Today's unsupervised ASR word error rates are similar to those achieved by supervised ASR 25 to 30 years ago. For example, in 2013, Walter et al.'s [97] best unsupervised TIDIGITS word error rate (WER) was 1.5% while, in 1987, Bush & Kopec's [14] (reported by Normandin et al. [64]) supervised TIDIGITS WER was also 1.5%. Modern supervised ASR can solve this task with zero errors.

We do not tackle the entire unsupervised ASR task in this work. Instead we focus on unsupervised techniques that can be applied within the scope of speech signal representation and acoustic modelling. Specifically, we wish to learn representations that enable an acoustic model to more accurately discriminate between different classes of speech units. Discrimination is a function of similarity; different speech unit classes can be discriminated if speech units within a class are similar to one another and dissimilar to speech units in other classes.

For the purposes of this work, we use speech signals encoded as sequences of continuous-valued vectors. Each vector represents a short *frame* of the signal. For example a particular utterance of the word */kæt/* might be represented by the sequence $\mathbf{x}_{1:L}$ of length L where each element $\mathbf{x}_t \in \mathbb{R}^D$ is a continuous-valued vector of dimension D. Section 2.2 describes the process of encoding analogue speech signals into frame sequences and a standard frame representation, mel frequency cepstral coefficients (MFCCs). Most unsupervised speech processing techniques, including the evaluation and downstream tasks we are interested in (see Section 2.3), rely on measuring similarity between speech signals. The basis for these similarity comparisons is the

DTW algorithm.

A sequence similarity measure is a function of the similarity of the elements in the sequences. In general the sequences being compared may be of different lengths so their elements must first be aligned. DTW, illustrated in Figure 2.1, is an algorithm for aligning two sequences such that the sum of the element alignment costs is a global minimum. To compute the cost of aligning a particular pair of frames we use a vector-space distance function, such as the cosine distance, to compare their vector representations.

The DTW algorithm can provide, as outputs, both the optimal alignment of frames and a global minimum cost of that alignment. We are usually only interested in the global minimum alignment cost which we interpret as a measure of the distance between two sequences, $D^{DTW}(\mathbf{x}_{1:L^x}, \mathbf{y}_{1:L^y})$; a small alignment cost implies a small distance which implies a high degree of similarity. We also use DTW optimal frame alignments when training correspondence autoencoders (see Chapter 3). Details of the dynamic programming algorithm used to compute DTW, and an analysis of its computational complexity can be found in Section 4.3.

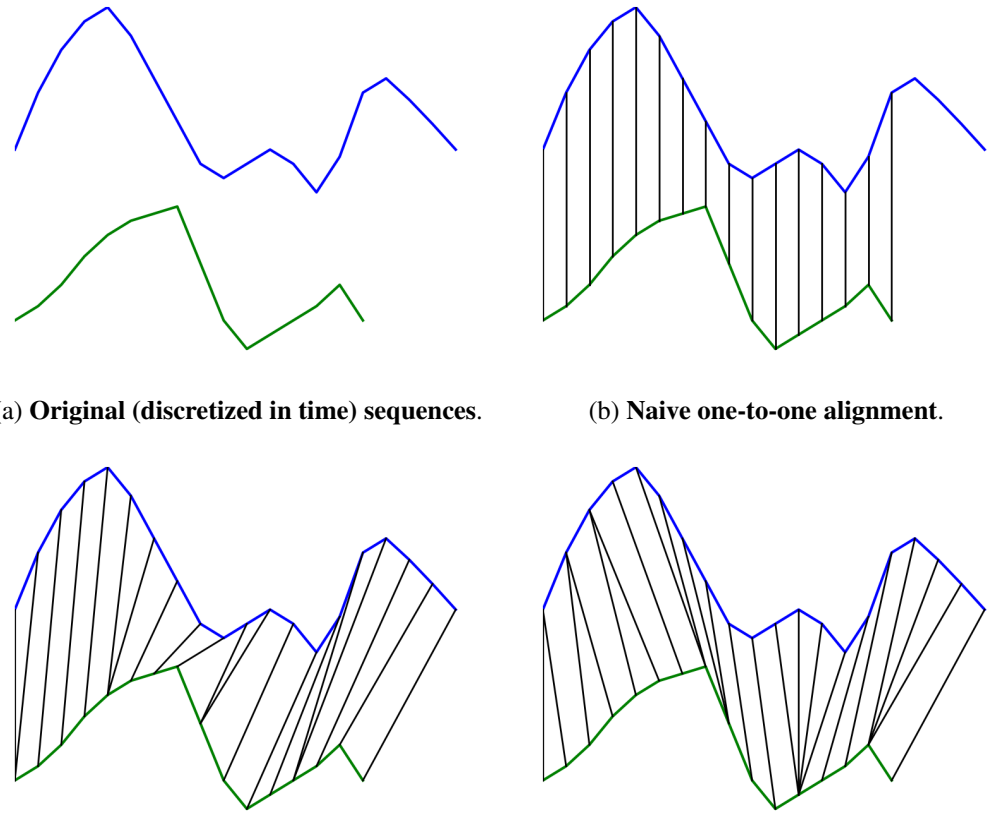
The significance of our contributions can be best understood within the context of a downstream system, one that would use our representations, such as that of Kamper et al. [49]. Their system achieves fully unsupervised small-vocabulary ASR using a segmental Bayesian model that performs both segmentation and clustering. Arbitrary length speech segments are mapped to points in a fixed-dimensional space and a Gaussian mixture model identifies clusters of points within that space. For the mapping, Kamper et al. use the fixed size embedding approach of Levin et al. [55]:

1. Speech segments are initially represented as sequences of frame representations (e.g. MFCCs).

2. A set of exemplar speech segments form a *reference set*. This set is initialized to random segments from the corpus but is refined iteratively using the Bayesian model's clusters.

3. The fixed size representation of a speech segment is computed in two steps:

   (a) Computing the DTW distance between the speech segment and every exemplar in the reference set. This yields a *reference vector* of DTW distances whose size is equal to the size of the reference set (e.g. 5,000).



(a) **Original (discretized in time) sequences**.

(b) **Naive one-to-one alignment**.

(c) **Constant rate alignment**.

(d) **Dynamic time warping alignment**.

Figure 2.1: **Dynamic time warping**. How similar are the two sequences in Figure 2.1a? Their lengths differ so a naive one-to-one alignment leaves some of the longer sequence unaligned (Figure 2.1b). A constant advancement rate assumption compares the full lengths but incorrectly aligns minima with maxima (Figure 2.1c). Dynamic time warping allows the alignment rate to vary and more accurately aligns the maxima and minima in the sequences (Figure 2.1d).

(b) Applying Laplacian eigenmaps [5] to the reference vector reduces the dimensionality (e.g. from 5,000 to 15) to form an *embedding vector*. This approach maintains local similarity properties – nearest neighbours in the reference vector space remain near neighbours in the embedding vector space.

Our speech frame representation learning methods, presented in Chapter 3 and Section 4.2, aim to deliver better frame representations than MFCCs (Step 1 above). We hypothesize that using automatically learnt frame representations can improve the quality of the embeddings generated by the procedure above compared to using MFCCs.

Although the embedding procedure described above is partially motivated by a desire to avoid costly DTW comparisons, DTW still figures strongly in the procedure's runtime. We hypothesize that our linear time DTW approximation, presented in Section 4.3, can eliminate the DTW runtime bottleneck (Step 3a above). Achieving this would allow the procedure to be applied to larger datasets enabling larger vocabulary unsupervised speech processing.

Our fixed-size representation learning methods, presented in Chapter 5, can replace the entire procedure described above (Steps 1 through 3b). We hypothesise that some variation of the techniques presented in Chapter 5 can yield fixed size representations that perform at least as well within downstream systems, such as Kamper et al.'s Bayesian clustering model, and do so with lower computational costs because DTW would no longer be required.

2.2 Speech representation learning

The communication of information by natural human speech uses an analogue physical process involving patterns of pressure or density differentials propagating through a substrate, typically air, between speaker and listener. The human speech production and reception systems are complex and we do not concern ourselves with their details in this work. However, that detail has played an important role in the engineering of speech signal representations, and is important for interpreting the patterns that are being captured in automatically learnt representations.

Natural speech signals are converted into digital form by sampling the analogue signal using a microphone at a particular rate, e.g. 16 kHz. Each digital sample comprises a measurement of the signal amplitude which is the degree to which the

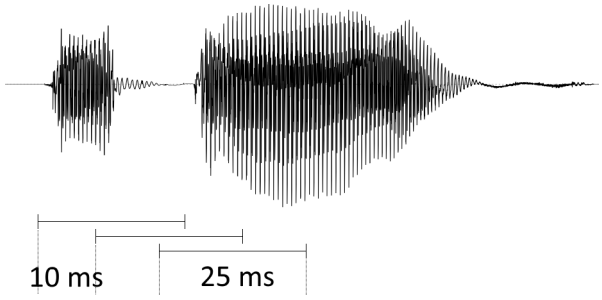


Figure 2.2: **Frame representations**. A mel frequency cepstral coefficient feature vector is computed for every frame where frames start every 10 ms and cover a 25 ms period of the signal. *Figure elements are not to scale. Modified version of image from Wikipedia (*`https://goo.gl/2U4ExM`*).*

substrate pressure is deviating from the ambient pressure. The amplitude measurement may be positive or negative and is quantized into an integer value with a particular degree of fidelity, e.g. 16 bits. These sequences of digital samples, that form the raw data our systems are working with, are typically stored in a format known as pulse-code modulation (PCM).

Until recently, speech recognition systems using the raw PCM data as input have not yielded good quality results. Advances in artificial neural network (ANN) architectures, combined with the ability to train with large quantities of data, have enabled close to state-of-the-art word error rates to be achieved when using raw inputs [81, 9, 66, 33]. However, we focus on the more traditional approach of first transforming the raw data into mel frequency cepstral coefficient (MFCC) feature vectors [20].

Full details of the MFCC computation are unimportant for this work but, in summary, MFCC features represent speech using a sequence of continuous value vectors. Each vector represents a short segment, or *frame*, of the speech signal, e.g. 25 ms in length, and frame representations are constructed at a constant rate, e.g. 100 Hz (every 10 ms), as shown in Figure 2.2. We typically compute 12 MFCCs for each frame which involves extracting spectral information using the discrete Fourier transform, converting to the mel scale, and computing the cepstrum (the spectrum of the log spectrum). The MFCC computation is designed to adjust the speech signal in ways that are similar to the transformation that occurs within the human speech perception system. The 12 MFCCs are usually combined with a measurement of the amount of energy in the frame, yielding 13 features per frame. In our work, we apply another standard technique: supplementing these 13 features with their deltas (i.e. velocity) and double-deltas (i.e. acceleration) to form a 39-dimensional feature vector per frame.

MFCC features have been shown to be an effective representation for both supervised speech recognition and unsupervised speech segment discrimination. Carlin et al. [15] and Schatz et al. [83, 84] have conducted studies to evaluate various representations, including MFCCs, in the context of unsupervised discriminability tasks – those described in Section 2.3. In both studies MFCCs compare favourably to the various alternatives. Within the context of deep neural network acoustic modelling, mel-scale log-filter bank features have been shown to outperform MFCCs, e.g. Li et al. [56], but in this work we stick with MFCCs.

State-of-the-art speech technologies have long used MFCC feature vectors which have developed from expert knowledge of signal processing and the human speech perception system, and from experimental results demonstrating their frequent superiority to alternatives. Modern machine learning techniques offer the possibility of optimizing speech signal representations beyond what has been achieved by human experts thus far. Furthermore, feature discovery for zero or low resource languages can be achieved without the need for costly and rare human experts by using unsupervised machine learning algorithms.

When applying machine learning to the task of speech representation learning, the principal goal is to obtain a function that maps inputs (typically conventional feature representations such as MFCCs) to a new, latent, representation. We use machine learning to find an optimal set of parameters for a given model structure, training objective, and training dataset. We encode our prior knowledge about the domain, and qualities desired of the resulting representations, into the model structure, its hyperparameters, and the training objective. With appropriate choices for these design decisions we can achieve better quality representations primarily because they can be task specific or emphasise a particular property that is known to be important for the types of tasks we are interested in.

In general, we are interested in unsupervised methods but this is not always true. To deal with zero-resource languages we may train a representation function on a high resource language and then apply it without any further training to the zero-resource language. This is done in Chapter 3 where a supervised ANN is trained on English but then applied to the African Bantu language Xitsonga. For low-resource languages we may apply semi-supervised techniques or adaptation.

For a good, high-level, review of representation learning, the reader is advised to start with Bengio et al. [6]. This review includes a section that answers the question "why should we care about learning representations?" by relating it to the field of speech

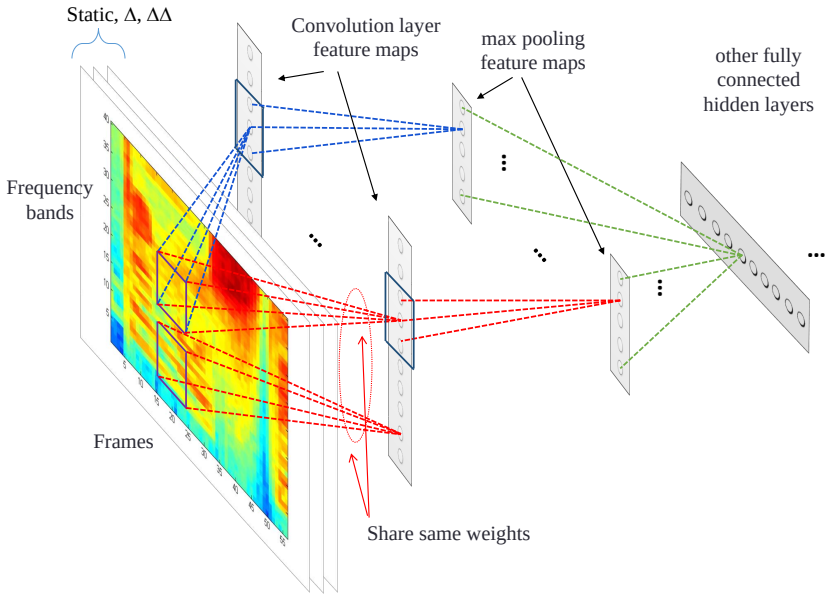


Figure 2.3: **Convolutional artificial neural network**. The input to the network in this figure is a spectrogram, but any frame representation sequence can be used (e.g. MFCCs). A set of filters scan over the input in two dimensions: time and frequency. *Reproduced from* [1] *with permission of author.*

recognition. We do not review the field in as much breadth in this section. Instead we highlight the techniques that provide foundation for our contributions or set our contributions in context. The techniques that apply most directly to our contributions are introduced in more detail in later sections. In this section we start with an overview of ANN techniques and follow with an overview of non-ANN techniques.

State-of-the-art supervised ASR typically uses a mixture of deep or convolutional ANNs (DNNs/CNNs) and hidden Markov models (HMMs). There are two common methods for combining these components: tandem and hybrid. In the tandem approach [37, 24] the ANN learns to transform the original features into better quality latent features; the latent features are then combined with the original features as input to conventional training of a GMM/HMM acoustic model. In the hybrid approach [61, 93, 27] the ANN replaces the GMM component and is trained to predict the HMM state posteriors directly. In both cases the ANN learns representations of short-duration frame sequences. Both methods can be combined as illustrated by some work related to our own: keyword spotting on low resource languages by Rath et al. [73]

DNNs are typically trained with a small, fixed-length, sequence of frames as input (e.g. 9 frames) yielding features in higher level layers of the network (often a narrow

bottleneck) that represent the whole window. CNNs typically convolve and pool over time yielding patterns spanning similar numbers of frames as DNNs. Consequently both DNNs and CNNs can learn short-duration patterns but neither can learn longer duration patterns without the HMM, or by introducing recurrence, as we do in Chapter 5.

Unsupervised ANN representation learning for ASR attempts to replicate the behaviour of the DNN/CNN components of supervised ASR systems without the use of a HMM to guide the representations towards those useful for the task of modelling the speech segments in the lexicon.

Two general and commonly used unsupervised ANN representation learning techniques are autoencoders [41] and restricted Boltzmann machines (RBMs, [38]). Autoencoders are simple ANN models, typically trained discriminatively to reconstruct an input via a hidden latent representation. Autoencoders are an important foundation technology for our work and are detailed in Section 2.4. An RBM is a generative probabilistic model with an ANN (i.e. fully connected) structure. RBMs are also trained to reconstruct their input but, in contrast to autoencoders, they are undirected graphical models and thus require training methods that involve sampling, such as contrastive divergence [39]. RBMs have been used for speech representation learning [45], but in this work we focus on autoencoders and their ilk.

ANNs often require regularization to avoid overfitting or degenerate solutions. Many forms of regularization are available including generally applicable methods such as L1 or L2 norms applied to the model parameters. Regularization is important when deploying ANNs for representation learning because we wish to avoid trivial solutions to the training objective (e.g. reconstruction via an identity function). We focus on the standard methods of denoising and representation contraction, and also develop the recently introduced correspondence pairing approach. These topics are explored in more detail in Section 3.

The field of non-ANN-based representation learning is broad, deep, and has an extensive history. We barely touch on it here as it has little direct relevance to our work but mention a few methods where they have particular relevance to representation learning for unsupervised speech processing. We do not consider supervised approaches here (e.g. linear discriminant analysis).

Perhaps the most widely used and simplest unsupervised representation learning method is principal components analysis (PCA). PCA transforms the input features into a set of features that are no larger in size than the inputs (but are often smaller) and

where the output features are linearly uncorrelated. PCA is equivalent to a single linear bottleneck layer autoencoder in that the two methods learn to project the inputs into the same subspace [12]. PCA is a linear method and linear methods cannot be usefully layered to form deep methods – a linear function of a linear function is itself a linear function. We use non-linear autoencoder layers which can be usefully stacked into deep architectures.

Independent components analysis (ICA) can be used to learn representative speech features [54]. ICA hypothesises the existence of a set of independent components and that mixtures of these components can explain the input data. There are linear and non-linear shallow variants, and deep non-linear variants, of ICA.

A variety of non-ANN methods for learning fixed-size representations of variable length speech sequences are evaluated by Levin et al. [55]. Their principal contribution is the evaluation of different dimensionality reduction techniques applied to the DTW exemplar distances method described in Section 2.1. We target the same goal in Chapter 5 where we consider ANN-based approaches for learning fixed size representations of variable length sequences.

Sparse coding has delivered some promising results in recent applications to speech representation learning [86]. We do not apply sparsity constraints within any of our models but this is an interesting area for future work – for example, the representations learnt by frame-autoencoders or frame-sequence-autoencoders can be regularized via sparsity constraints.

2.3 Evaluation tasks

Most speech representation learning aims to improve the quality of conventional supervised ASR systems. Consequently, the evaluation method typically involves using a proposed representation function within a conventional supervised ASR system and evaluating the change in word error rate. In contrast, we are interested in unsupervised speech recognition where evaluating our representation functions within a supervised ASR system could be misleading (because additional modelling assumptions confuse matters) and would be unnecessarily time consuming. Instead, we focus on tasks and evaluation methods that more directly measure the effectiveness of our representation functions in an unsupervised setting.

Throughout this work we focus on two similar evaluation tasks: *same-different* and *ABX*. Both tasks evaluate the effectiveness of a representation function to discriminate

between different classes of sequences. Representation functions that make every class clearly distinct from all other classes are preferred. In principle, both *same-different* and *ABX* could be applied to sequences representing any meaningful unit of speech but, following past work, we apply *same-different* to only whole-term pairs and *ABX* to only minimal triphone pairs.

When evaluating a frame representation function, as in Chapter 3, sequence similarity is found via DTW combined with cosine distances between the frame representations. When evaluating a fixed size representation function, as in Chapter 5, sequence similarity is measured by a single cosine distance between the two fixed size representations.

In summary, there are three independent axes of variation in the evaluation method:

- Comparison type: *same-different* or *ABX*.
- Pair type: whole-term pairs or minimal triphone pairs.
- Representation type: per-frame or whole-sequence.

2.3.1 Whole-term *same-different*

The whole-term *same-different* task [15, 55] measures the discriminability of a representation function by comparing the representations of two term instances and asking whether they belong to the same or different classes. This is a binary classification task (though tackled indirectly) and can be thought of as an instance of information retrieval where same term-pair instances are “relevant" and different term-pair instances are “irrelevant".

The test set typically includes a mixture of both same-speaker and different-speaker term pairs. The latter is clearly a more difficult condition so the proportion of different-speaker term pairs in the test set is an important statistic to bear in mind.

Same-different evaluations are achieved by firstly computing the distance between the representations of every term pair in the test data set and secondly computing the average precision, i.e. the area under the precision-recall curve. The precision-recall curve is defined by a distance threshold. A term-pair are taken to be different if the distance between their representations is greater than the threshold, otherwise they are taken to be the same. In principle, the precision-recall curve is found by computing precision and recall at every threshold value but this is equivalent to ranking all term-pairs by distance and computing average precision over the ranked list, an easier computation.

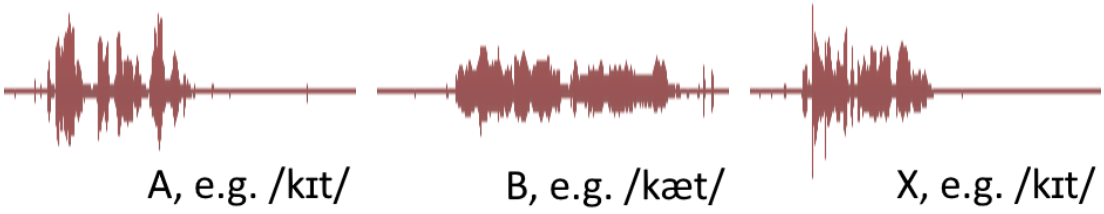


Figure 2.4: **Minimal triphone pair *ABX* evaluation task**. Is X more similar to A or B? *Waveforms do not correspond to example triphones*.

2.3.2 Minimal triphone pair *ABX*

A triphone is a sequence of three phones with the focus on the middle phone. The minimal triphone pair *ABX* task [83, 84] measures the discriminability of a representation function by asking whether triphone x is most like triphone a or triphone b, where a and x are distinct examples of the same triphone and b is a triphone differing from a and x in only the middle phone. This task evaluates a model's ability to discriminate between different phones in the same context.

For example, a and x might be two different examples of the triphone */kɪt/* while b might be an instance of the triphone */kæt/,* as depicted in Figure 2.4.

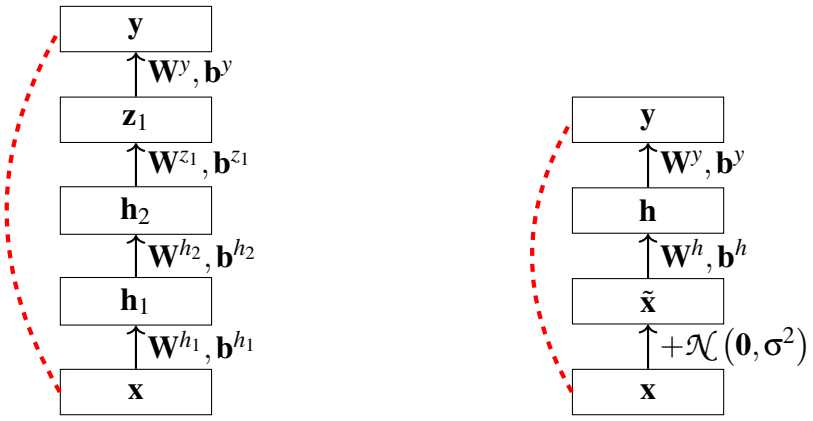
We consider two variants:

- Within-speaker: a, b, and x belong to the same speaker.
- Across-speaker: a and b belong to one speaker and x belongs to a different speaker.

If the distance between a and x is greater than that between b and x then the model has made an error. The error rate is the mean over all possible (a,b,x) triples in the test set.

2.4 Standard artificial neural network architectures

Much of this work involves the development of novel ANN architectures. In this section we describe the foundational architectures on which our contributions are based. Only the standard architectures directly relevant to our work are presented in detail here. A variant of the autoencoder architecture, the correspondence autoencoder, was introduced previously by Kamper et al. [48] and the introduction to that model can be found in Section 3.2, where we go on to develop it further.



(a) **Autoencoder**. Input $\mathbf{x}$ is encoded into latent representation $\mathbf{h}_2$ then decoded into reconstruction $\mathbf{y}$. The model is trained to minimize the difference between $\mathbf{x}$ and $\mathbf{y}$. May be shallow or deep (shown here).

(b) **Denoising autoencoder**. Artificial noise (e.g. Bernoulli masking, or additive Gaussian) is added to input $\mathbf{x}$ to form noisy input $\tilde{\mathbf{x}}$. The model is trained to minimize the difference between reconstruction $\mathbf{y}$ and original, clean, input $\mathbf{x}$. May be shallow (shown here) or deep.

Figure 2.5: **Autoencoder models**. Schematics of standard autoencoder model variants. The input is denoted by $\mathbf{x}$, the reconstruction by $\mathbf{y}$, encoding layers by $\mathbf{h}_l$, and internal decoding layers by $\mathbf{z}_l$. Dashed red lines indicate differences that are minimized during training. Model parameters are shown for each layer. When deep, encoder and decoder layers are always paired and may share weights $\left(\text{e.g. } \mathbf{W}^{z_1} = \left(\mathbf{W}^{h_2}\right)'\right)$.

Throughout this document we denote ANN layers as vectors. In practice we use minibatch stochastic gradient descent to train our networks, where a layer's activations are actually implemented as matrices, but we omit this complexity in our presentation for clarity.

2.4.1 Autoencoder

A single-layer *autoencoder* (AE) [12] is an ANN with two components. The *encoder* projects an input, e.g. an MFCC vector, $\mathbf{x} \in \mathbb{R}^{D_0}$ into hidden representation $\mathbf{h}_1 \in \mathbb{R}^{D_1}$. The *decoder* projects the hidden representation back into the original vector space $\mathbf{y} \in \mathbb{R}^{D_0}$. We treat $\mathbf{y}$ as a reconstruction of $\mathbf{x}$ and train the network to minimize the reconstruction squared error $\mathcal{L}(\mathbf{x}) = \|\mathbf{y} - \mathbf{x}\|_2^2$.

The encoder is implemented as a conventional feedforward ANN layer, $\mathbf{h}_1 = f_1\left(\mathbf{W}^{h_1}\mathbf{x} + \mathbf{b}^{h_1}\right)$, with weight parameters $\mathbf{W}^{h_1} \in \mathbb{R}^{D_1 \times D_0}$, bias parameters $\mathbf{b}^{h_1} \in \mathbb{R}^{D_1}$,

and non-linear activation function f_1 (we use the hyperbolic tangent). The decoder has a similar form as the encoder, $\mathbf{y} = \mathbf{W}^y\mathbf{h}_1 + \mathbf{b}^y$, with additional weight parameters $\mathbf{W}^y \in \mathbb{R}^{D_0 \times D_1}$ and bias parameters $\mathbf{b}^y \in \mathbb{R}^{D_0}$. To reconstruct unbounded data, such as MFCCs, the reconstruction $\mathbf{y}$ must have an unbounded range, hence the use of a linear decoder.

Deep narrow AEs typically achieve lower reconstruction error than shallow wide AEs with the same number of parameters. As shown in Figure 2.5a, each of a deep AE's L encoders project the output of the previous encoder into a new hidden representation, i.e. $\mathbf{h}_i = f_i\left(\mathbf{W}^{h_i}\mathbf{h}_{i-1} + \mathbf{b}^{h_i}\right) \in D_i$ for all $i \in [1, L]$, with $\mathbf{h}_0 = \mathbf{x}$. Each of the L decoders reconstruct their respective hidden representations in turn, finally reconstructing the input, i.e. $\mathbf{z}_i = g_i\left(\mathbf{W}^{z_i}\mathbf{z}_{i+1} + \mathbf{b}^{z_i}\right) \in D_i$ for all $i \in [1, L-1]$, with $\mathbf{z}_L = \mathbf{h}_L$ and $\mathbf{y} = \mathbf{W}^y\mathbf{z}_1 + \mathbf{b}^y$. We tie weights and use non-linear activation functions in all internal decoders, i.e. $g_i = f_{i+1} = \tanh$ and $\mathbf{W}^{z_i} = \left(\mathbf{W}^{h_{i+1}}\right)'$ for all $i \in [1, L-1]$. Our autoencoders use a consistent hidden layer size, i.e. $D_i = D_{i-1}$ for all $i \in [2, L]$.

Training all layers in a deep AE concurrently often yields poor results due to the vanishing gradient problem [42, 7] (Section 2.4.5). We use the standard mitigation of pre-training the deep AE layerwise, then fine-tuning the entire network [40].

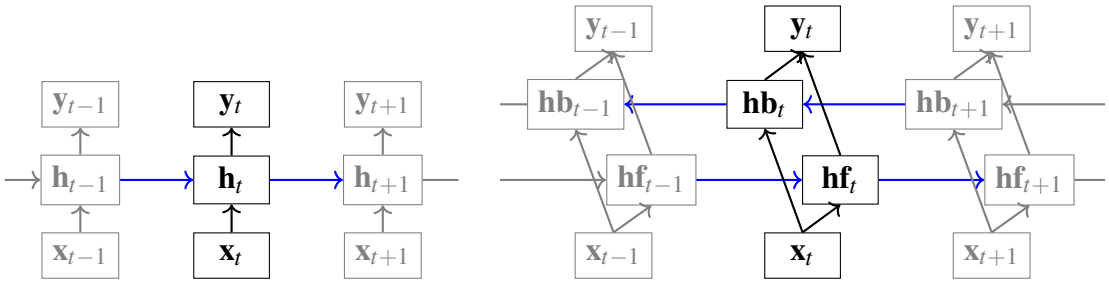
2.4.2 Denoising autoencoder

AEs are not a strong baseline for representation learning. Denoising autoencoders (dAEs, Figure 2.5b) [96] usually perform better because they implicitly regularize the parameters avoiding degenerate transformations, such as the identity function, being learned. Regularization is especially important when training overcomplete AE architectures, i.e. where $D_i \geq D_0$ for all $i \in [1, L]$.

A dAE is trained to reconstruct the clean versions of artificially noisy inputs. Different types of noise may be applied. In our case the input features, once normalized to zero mean and unit variance, are approximately Gaussian distributed so additive zero mean Gaussian noise is appropriate. A dAE is identical to a conventional AE except the input $\tilde{\mathbf{x}} = \mathbf{x} + \mathcal{N}(\mathbf{0}, \gamma\mathbf{I})$ is a noisy version of $\mathbf{x}$. γ is a hyperparameter determining the standard deviation of the noise.

2.4.3 Recurrent neural network

A RNN is an ordinary feedforward ANN containing a special layer that propagates a hidden state along the input sequence. We focus on the Elman-style RNN layer [25, 77,



(a) **Unidirectional recurrent neural network**. This architecture maintains a hidden state along the length of the input sequence that is a function of the previous hidden state and current input. The recurrent layer is denoted by $\mathbf{h}_t$.

(b) **Bidirectional recurrent neural network**. This architecture maintains a hidden state along the length of the input sequence in both directions. The forward recurrent layer is denoted by $\mathbf{hf}_t$ and backward recurrent layer by $\mathbf{hb}_t$.

Figure 2.6: **Recurrent neural network models**. In these schematics, the input is denoted by $\mathbf{x}_t$ and the output by $\mathbf{y}_t$. Black lines indicate ordinary feedforward connections and blue lines indicate recurrent connections (shown unwound).

78]. A RNN layer accepts elements from the input sequentially and, given a specific example sequence, can be *unwound* to yield a network containing a replica of the RNN layer for each element in the sequence. The number of times the RNN layer appears in the unwound network is thus equal to the input sequence length. When considered more generally (i.e. not unwound for a specific input example), the key feature of a RNN layer is the recurrent connection that feeds information forward from one position along the sequence to the next. The t'th element $\mathbf{x}_t \in \mathbb{R}^V$, where V is the dimensionality of the vectors in the input sequence $\mathbf{x}_{1:L}$, is fed into a recurrent network via

$$\mathbf{h}_t = f\left(W^h\left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}^h\right) \tag{2.1}$$

The neural network layer described in Equation 2.1 is recurrent because the hidden layer state at position t is a function of the same hidden layer's state at position $t-1$, in addition to the input element at position t. A slice through an unwound RNN is depicted in Figure 2.6a.

The dimensionality of the hidden state, D, remains constant along the sequence, i.e. $\mathbf{W}^h \in \mathbb{R}^{D\times(D+V)}$, so the hidden state at position t can be thought of as a fixed size representation of the sequence prefix at position t. Subject to an appropriate training loss function, the hidden state found at the end of the sequence, $\mathbf{h}_L$, can be thought of as a representation of the entire sequence. Typically $D \ll VL$ so the network is incapable of memorising the entire sequence verbatim – it is forced to learn a compression function.

Furthermore, the compression ratio is typically high enough, or the data is typically complex enough, that the compression function will necessarily be lossy. By using a good training loss function we can encourage the network to remember the aspects of the sequence we believe to be important for representing the sequence such that similar sequences have similar representations and dissimilar sequences have dissimilar representations.

The activation function f typically has a sigmoidal shape. Common activation functions in recurrent neural networks are the hyperbolic tangent and the logistic function; both quickly saturate as pre-activation value magnitudes increase. Alternatively, the rectified linear unit (ReLU) activation function [62], used by Hannun et al. in their end-to-end speech recognition recurrent network [36], is piece-wise linear which helps avoid the vanishing gradient problem (see Section 2.4.5).

2.4.4 Bidirectional recurrent neural network

The hidden state, of an ordinary recurrent neural network, at a particular position along the input sequence can be interpreted as a representation of the sequence prefix up to that position, i.e. a representation of the preceding context. Sometimes our training objective can benefit from a greater degree of context, i.e. the sequence suffix in addition to the sequence prefix. For example, accurate identification of a phone can be strongly influenced by the preceding and following phones, hence the use of triphones, due to the way in which spoken sounds flow into one another. A bidirectional recurrent neural network [85] maintains two hidden states at each position of a sequence; one represents the prefix (the forward direction), the other represents the suffix (the backward direction).

Figure 2.6b depicts a general slice from an unwound bidirectional neural network. The recurrent layer aspect of this same network is defined by Equations 2.2 through 2.4.

$$\mathbf{hf}_t = f\left(W^{hf}\left[\mathbf{hf}_{t-1};\mathbf{x}_t\right]+\mathbf{b}^{hf}\right) \tag{2.2}$$

$$\mathbf{hb}_t = f\left(W^{hb}\left[\mathbf{hb}_{t+1};\mathbf{x}_t\right]+\mathbf{b}^{hb}\right) \tag{2.3}$$

$$\mathbf{h}_t = \left[\mathbf{hf}_t;\mathbf{hb}_t\right] \tag{2.4}$$

The complete hidden state at position t is a concatenation of the forward and backward recurrent states at position t; this concatenated value is passed on to higher level layers. A fixed size representation of the entire sequence can be obtained by concatenating the forward hidden state at the end of the sequence and the backward

hidden state at the start of the sequence, $\mathbf{r} = [\mathbf{hf}_L; \mathbf{hb}_1]$; both components could represent the entire sequence but in different ways so their combination may be a stronger summary of the entire sequence.

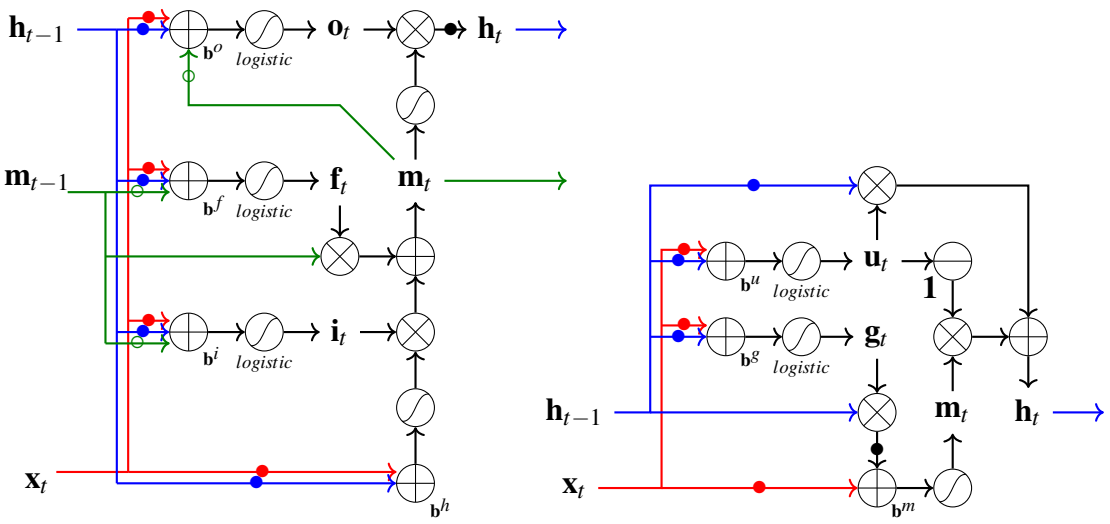
In many practical use cases, such as speech recognition, a bidirectional recurrent neural network may not be particularly cognitively plausible as it implies the entire sequence is observed and then analysed post hoc. However, this work is not concerned with the development of a computational cognitive model of unsupervised speech recognition so we adopt the use of bidirectional network architectures whenever they offer an improvement in the task of interest: representation learning for unsupervised *automatic* speech recognition.

2.4.5 Dealing with long distance dependencies

Many practical tasks involve data that exhibits long distance dependencies. That is, behaviour at position t needs to be a function of an input provided at position $t - \delta$ where δ is "large". What constitutes a long distance is model, task, and data specific. As an example, consider the task of phone identification. Current phone identity is often dependent on the identity of the preceding and following phones. Phone duration is typically 30-90 ms (3-9 frames) but can be as much as 400 ms (40 frames) [69]. The last frame of the current phone may need information from as much as 60 frames in the past. It can be difficult to train a vanilla RNN to propagate a complex state through 60 recurrent steps.

The problem occurs during training: the error signal used to update the weights is back-propagated through the unwound network, using back propagation through time [80, 98], and at each step it passes through a squashing non-linearity (e.g. the gradient of the tanh function is $\tanh'(x) = 1 - \tanh^2(x)$). Without "good weights" (which is rare) the squashing functions cause the gradients to vanish to zero or explode to infinity. This vanishing/exploding gradient problem is analysed in more detail by Hochreiter et al. [42, 43].

In Chapter 5 we use RNNs to learn fixed size representations of whole sequences. These approaches require all pertinent information to be propagated along the entire length of the input sequences, which can involve many tens or hundreds of steps. For their statistical machine translation task, Cho et al. [17] were unable to get meaningful results using a vanilla RNN; instead they used gated recurrent units to learn fixed size representations of arbitrary length word sequences.



(a) **Long short-term memory**. The schematic for a general slice from an unwound LSTM layer. The network learns to attenuate the input, hidden state, and output via three multiplicative gates which each take on values in $[0, 1]$. Information can be stored in the hidden state indefinitely thanks to the lack of a non-linearity along the $\mathbf{m}_t$ recurrent connection. An open circle on a (green peephole) line indicates a diagonal matrix multiplication. This figure depicts the LSTM variant we use, including peephole connections and an output projection.

(b) **Gated recurrent unit**. The schematic for a general slice from an unwound GRU layer. The network learns to attenuate the input and hidden state via two multiplicative gates which each take on values in $[0, 1]$. Information can be stored in the hidden state indefinitely thanks to the lack of a non-linearity along the $\mathbf{h}_t$ recurrent connection. The symbol denoted by a circle with a horizontal line running across it and with a **1** to the lower left indicates the operation of subtracting the input from a suitably sized vector of value 1.

Figure 2.7: **Alternate recurrent neural network layers**. The LSTM and GRU layers offer methods for avoiding the vanishing/exploding gradient problem by allowing information to flow along the hidden state's recurrent update connections without being modified when required. A filled circle on a line indicates a dense matrix multiplication. Circles with an s-shape inside indicate an application of a standard non-linear squashing function; the specific type, if fixed, is indicated beneath the circle.

Although, in theory, the gradients may explode or vanish, in practice we tend to be in a situation where they vanish so the term "vanishing gradient problem" is usually all that is mentioned. Both of the following mitigation techniques, however, help avoid both vanishing and exploding gradients.

A general solution to the vanishing gradient problem is to allow information to be passed onwards through the network recurrently without being changed. Of course, we do want inputs to affect the hidden state so the following two solutions utilize "gates" which the network learns to open or close depending on the state. When a gate is closed the affected stream of information is discarded and when a gate is open the stream flows on unchanged. Gate values are continuous in the range $[0, 1]$, and are multiplied with the input, so the effect is an attenuation of the signal. The degree to which a gate is open is a function of the current input and current hidden state, so the network learns when to keep a gate open and when to close it. The hidden state thus now serves two notional purposes: (1) to pass information about the sequence prefix to later points in the sequence to achieve the training objective, and (2) pass information to later recurrent layers to control their gates.

Both of the following mechanisms are different types of RNN recurrent layers and can be used as direct replacements for the vanilla RNN recurrent layer described in section 2.4.3. Both can also be used in a bidirectional RNN.

2.4.5.1 Long short-term memory

The long short-term memory (LSTM) architecture (Figure 2.7a) was originally proposed by Hochreiter and Schmidhuber in 1997 [44] but has been repeatedly enhanced in later years [29, 30, 82]. The key features of the LSTM architectures are three gates: the input, forget, and output gates attenuate the input, hidden state, and output signals respectively. The LSTM can be thought of as having two recurrent hidden states: (1) the internal state $\mathbf{m}_t$ which is only visible to, and used by, the LSTM layer itself, and (2) the external hidden state $\mathbf{h}_t$ which is the value passed on to the next layer(s) in the network. The LSTM has become particularly popular in recent years, including in acoustic modelling [82] and language modelling [74].

The original LSTM formulation [44] introduced input and output gates to selectively ignore some inputs and selectively avoid emitting information at some positions respectively. A particularly important improvement was the introduction of a forget[1]

[1]The forget gate is poorly named as when the gate is fully open, i.e. when it takes on its maximal value of 1, the layer does not forget anything! It would be better named the *remember* gate but we stick

gate [29] which allows the network to selectively forget information by stemming the information flow from the previous hidden state. The vanilla RNN is incapable of explicitly forgetting information, all that can happen is for later information to overwrite earlier information. Further improvements were made with the introduction of peephole connections [30] (an efficient view of the internal hidden state as input to the gate functions) and the introduction of an output projection layer [82] (enabling the network's internal hidden state to have a different, often larger, size compared to the external hidden state).

In this work we use an LSTM variant that includes the forget gate, peephole connections, and output projection, all defined by Equations 2.5 through 2.9 and depicted in Figure 2.7a.

$$\mathbf{i}_t = \sigma\left(\mathbf{W}^i\left[\mathbf{x}_t;\mathbf{h}_{t-1};\mathbf{m}_{t-1}\right]+\mathbf{b}^i\right) \tag{2.5}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}^f\left[\mathbf{x}_t;\mathbf{h}_{t-1};\mathbf{m}_{t-1}\right]+\mathbf{b}^f\right) \tag{2.6}$$

$$\mathbf{m}_t = \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{i}_t \odot f\left(\mathbf{W}^m\left[\mathbf{x}_t;\mathbf{h}_{t-1}\right]+\mathbf{b}^m\right) \tag{2.7}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}^o\left[\mathbf{x}_t;\mathbf{h}_{t-1};\mathbf{m}_t\right]+\mathbf{b}^o\right) \tag{2.8}$$

$$\mathbf{h}_t = \mathbf{W}^h\left(\mathbf{o}_t \odot f\left(\mathbf{m}_t\right)\right) \tag{2.9}$$

The input, output, and forget gates are denoted $\mathbf{i}_t$, $\mathbf{o}_t$, and $\mathbf{f}_t$ respectively. The internal and external hidden states are denoted $\mathbf{m}_t$ and $\mathbf{h}_t$ respectively. The activation functions used in the gates are required to be logistic functions, $\sigma(\mathbf{x}) = \frac{1}{1+exp(-\mathbf{x})}$ which has a range of $[0,1]$ as required for the gates. The activation functions used in the hidden state updates, f, can be anything; we use tanh.

The LSTM weights are all dense, as usual, except for the peephole connections, $W^{\{i,o,f\}}$, which are diagonal. Dense weights on these connections may be helpful but would increase the number of parameters quite significantly, especially if an output projection layer is also used such that $|\mathbf{h}_t| < |\mathbf{m}_t|$.

The gates all have the same dimensionality as the signals they are attenuating and can affect each dimension differently. For example, at a single position, the output gate might be concurrently fully open for the first dimension and fully closed for the second dimension.

with forget gate as that is what is used commonly in the literature.

$\mathbf{g}_t$	$\mathbf{u}_t$	$\mathbf{h}_t$ is function of
1	1	$\mathbf{h}_{t-1}$
1	0	$f(\mathbf{x}_t, \mathbf{h}_{t-1})$
0	1	$\mathbf{h}_{t-1}$
0	0	$f(\mathbf{x}_t)$

Table 2.1: **Gated recurrent unit update extremes**. The values that go into updating a GRU's hidden state depend on the state of the two gates. This table indicates what the extreme conditions look like. Gates with intermediate values provide intermediate weighted additive combination updates. Here, f indicates some non-linear function of its inputs.

2.4.5.2 Gated recurrent unit

The LSTM is an especially complicated contrivance, especially the version presented in Section 2.4.5.1, and may be more powerful than required for some tasks. The gated recurrent (GRU) architecture, introduced by Cho et al. [17] and further empirically evaluated by Chung et al. [18], has complexity and potential power somewhere between the vanilla RNN and LSTM. The GRU (Equations 2.10 through 2.13 and Figure 2.7b) uses two gates: (1) the reset gate, denoted $\mathbf{g}_t$[2], determines how much of the hidden state goes into the potential hidden state update $\mathbf{m}_t$, and (2) the update gate, denoted $\mathbf{u}_t$, determines how much of the potential hidden state update actually changes the hidden state $\mathbf{h}_t$.

$$\mathbf{g}_t = \sigma(\mathbf{W}^g[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}^g) \tag{2.10}$$

$$\mathbf{u}_t = \sigma(\mathbf{W}^u[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}^u) \tag{2.11}$$

$$\mathbf{m}_t = f(\mathbf{W}^m[\mathbf{x}_t; \mathbf{g}_t \odot \mathbf{h}_{t-1}] + \mathbf{b}^m) \tag{2.12}$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{u}_t)\mathbf{m}_t \tag{2.13}$$

The activation functions used in the gates are required to be logistic functions. The activation functions used in the hidden state updates, f, can be anything; we use tanh.

The GRU gates interact in a complex manner. Table 2.1 shows what values the hidden states become a function of at each of the gate combination extremes. If the update gate is close to 1 then the current input is mostly ignored and the hidden state

[2] We denote the reset gate with $\mathbf{g}_t$ to avoid potential confusion with representation values which are usually denoted $\mathbf{r}$.

is updated linearly. If the update gate is close to 0 then the hidden state is updated non-linearly and the reset gate determines how much that non-linear update depends on the current hidden state in addition to the current input.

Chapter 3

Learning representations of speech frames

Building on the artificial neural network models and evaluation tasks introduced in Chapter 2, we introduce the correspondence autoencoder (cAE), and our novel improvement to the cAE, in Section 3.2. Experimental results are presented in Section 3.3 where we compare our modified cAE to the original cAE and other baselines using the *ABX* subword discrimination task applied to English and the zero-resource language, Xitsonga.

The work presented in this Chapter was previously published in *The proceedings of INTERSPEECH 2015* as *A Comparison of Neural Network Methods for Unsupervised Representation Learning on the Zero Resource Speech Challenge* by *Daniel Renshaw*[*], *Herman Kamper*[†], *Aren Jansen*[‡], *and Sharon Goldwater*[*] [75].

3.1 Introduction

Automatic speech recognition systems are typically trained using tens or hundreds of hours of supervision (hand-transcribed speech data), and often still have difficulty dealing with differences in accent, gender, channel characteristics, and other factors. Yet months-old human infants begin to solve the basic problems of identifying phones and words with no comparable supervision. Recent work on zero-resource speech technology asks: how can we build artificial systems that might approach the unsupervised learning abilities of human infants? Solving this problem would provide more

[*,†]ILCC and [†]CSTR, School of Informatics, University of Edinburgh, UK
[‡]HLTCOE and CLSP, Johns Hopkins University, USA

universally available speech technology in under-resourced languages, and could lead to novel methods that also improve supervised speech recognition.

The Zero Resource Speech Challenge (ZRSC) promotes work in this area by defining two shared tasks and providing training data and common evaluation mechanisms. The challenge ran as a special session of INTERSPEECH 2015. Our entry tackles Track 1, subword modelling, by using several neural network based methods to learn frame-level representations that yield better phonetic discriminability than standard Mel Frequency Cepstral Coefficients (MFCCs) [20]. We evaluate our representations using the ZRSC's Track 1 minimal triphone pair *ABX* discrimination task.

We enhance the correspondence autoencoder (cAE) method of Kamper et al. [48], introduced in Section 3.2, which learns a non-linear mapping from MFCCs to a latent distributed feature representation. Kamper et al. trained their system on data from the Switchboard corpus and evaluated it using the *same-different* discriminability task [15], showing that the learned representations performed substantially better than the original MFCCs, and also better than representations learned by a standard autoencoder (AE) [12]. Here, we show that Kamper et al.'s model, with no additional tuning, generalizes well to other data sets, languages, and tasks by evaluating it using the *ABX* task with the two ZRSC datasets: Buckeye [68] (English, but with different channel characteristics than Switchboard) and the NCHLT Xitsonga[1] Speech corpus [21].

While the cAE is a weakly supervised model, we train it with correspondence pairs sourced from an unsupervised term discovery (UTD) system [47] making the approach unsupervised as a whole. Alternatives to the weakly supervised regularization implicit in the cAE were not considered in [48]. Here, we show that a standard unsupervised form of regularization, denoising autoencoders [95], learns better representations than AEs, but still not as good as cAEs. We also introduce an improved cAE architecture and training method that reduces the number of hyperparameters to be tuned, and show that narrow architectures work better, with reduced error rates on a zero-resource language after tuning on English. Unlike similar previous work [90, 89, 105] our methods, when considered as a single system, are fully unsupervised, train on individual frames without context, and use a loss function in the input vector space instead of the representation vector space.

Finally, we explore whether *ABX* performance can be improved on a zero-resource language by using representations trained on large amounts of supervised data in a different language. We use a deep neural network (DNN) trained on a large amount of

[1] Xitsonga is a southern African Bantu language.

English data to extract bottleneck features (BNFs) [34] for the Xitsonga test data. We find that these cross-language supervised representations perform comparably with the cAE representations trained with in-language data and that further improvements can be achieved by combining the two approaches.

Our baseline models are the vanilla autoencoder (Section 2.4.1) and the denoising autoencoder (Section 2.4.2). We compare the baseline models to two versions of the correspondence autoencoder and a deep neural network. The hyperparameters of the autoencoder models are optimized on English and then the resulting model architecture is applied without any further hyperparameter changes to Xitsonga. The DNN is trained on English and applied without any further training or adaptation to Xitsonga.

Optimizing a cAE architecture on English and applying it to Xitsonga, we obtain a relative error rate reduction of 35% compared to the original MFCCs. We also show that Xitsonga frame representations extracted from the bottleneck layer of a supervised DNN trained on English can be further enhanced by the cAE, yielding a relative error rate reduction of 39%.

3.2 Correspondence autoencoder

An autoencoder (AE, see Section 2.4.1) is trained with an unsupervised objective – reconstruction. We can learn better quality representations if we use an objective that better approximates the extrinsic task: learning features that discriminate between different subword units. To this end, Kamper et al. [48] introduced a weakly supervised AE variant: the correspondence autoencoder (cAE). Instead of reconstructing its input, a cAE is trained to predict and construct an unseen example that is known to be similar to the input.

We can view the cAE as a version of a dAE where, instead of artificial noise, the network is presented with input pairs that differ only in non-linguistic sources of variation, e.g. speaker or channel. Denoising the true sources of extrinsic variability is a more optimal method than introducing artificial sources, though we are limited to what can be discovered with the UTD system.

Four steps go into training a cAE for unsupervised speech representation learning as depicted in Figure 3.1.

1. Train stacked autoencoder: layerwise pre-training using AE objective

2. Unsupervised term discovery: identify likely same-class term pairs [47]

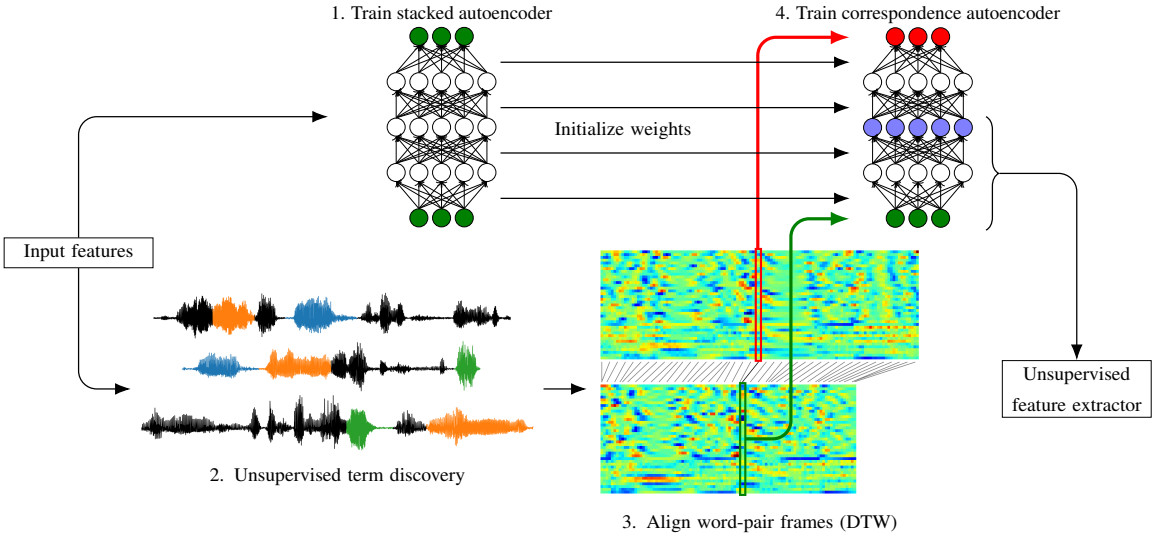


Figure 3.1: **Correspondence autoencoder training procedure**. An overview of the training procedure for our INTERSPEECH 2015 Zero Resource Speech Challenge entry. (1) The entire unlabelled speech corpus is used to pre-train a network using a standard autoencoder objective. (2) The same corpus is run through an unsupervised term discovery system. (3) Frames are aligned between every pair of UTD terms believed to be from the same class. (4) The pre-trained network is fine-tuned using the aligned frame pairs and the correspondence autoencoder objective. The final encoder portion of the network is used as our frame representation function. *Inspiration for this figure from Kamper et al. [48].*

3. Align word-pair frames: yielding frame pairs where frames in same relative positions

4. Train correspondence autoencoder: fine-tune whole network using cAE objective

We obtain a frame-level alignment for each segment pair using dynamic time warping (DTW) with cosine distance, yielding a set of frame pairs $\{(\mathbf{x}, \hat{\mathbf{x}})\}$. A deep AE is trained layerwise on the entire corpus to initialize the network's parameters prior to cAE fine-tuning. The cAE is then trained to minimize the reconstruction error of $\mathbf{y}$ relative to $\hat{\mathbf{x}}$ for each frame pair $(\mathbf{x}, \hat{\mathbf{x}})$, yielding the cAE loss $\mathcal{L}^{cAE}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{y} - \hat{\mathbf{x}}\|_2^2$. We obtain better results by using every pair twice, once where the first item is input to the network and a second time where the second item is input to the network.

In Kamper et al.'s previous work [48] the layerwise pre-trained decoders were discarded and a single, randomly initialized, decoder trained during cAE fine-tuning, as in Figure 3.2a. This approach has the advantage that the total number of layers is reduced, mitigating the vanishing gradient problem, but a single linear decoder is unable to undo the work of many non-linear encoders forcing some of the top encoder layers to be implicitly retrained as decoders during fine-tuning. The layer to use for representation must then be determined using a held-out validation set. In this work we use a deep non-linear decoder during cAE fine-tuning, as in Figure 3.2b, allowing the top encoding layer to always be used as the input's representation.

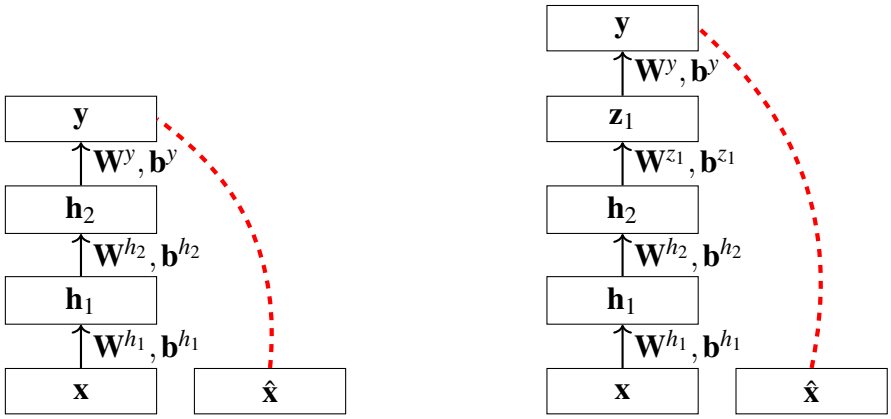
3.3 Experiments

3.3.1 Data

We use two datasets. The first is a 5 hour portion of the Buckeye corpus [68] distributed as part of the ZRSC. The second is a 2.5 hour portion of the NCHLT Xitsonga Speech corpus [21] consisting of 16 kHz, close-talking microphone, prompted speech.

As input we use HTK [99] MFCCs[2] with a 25 ms window and 10 ms step size, which are augmented with first and second order derivatives to yield 39-dimensional feature vectors. The MFCCs falling entirely within the speaker segments of interest–the ZRSC's evaluation intervals–are extracted and cepstral mean and variance normalization is applied to those segments per source file. All of the resulting frames are used during pre-training of our networks.

[2]Thanks to Herman Kamper.



(a) **Correspondence autoencoder with shallow decoder**. This variant uses a shallow decoder, i.e. a single decoder layer, irrespective of the number of encoder layers.

(b) **Correspondence autoencoder with deep decoder**. This variant uses a deep decoder, i.e. every encoder layer is paired with a decoder layer. Encoder/decoder layer pairs are nested, e.g. the bottom encoder is paired with the top decoder, and may share weights $\left(\text{e.g. } \mathbf{W}^{z_1} = \left(\mathbf{W}^{h_2}\right)'\right)$.

Figure 3.2: **Correspondence autoencoder models**. Schematics of two variants, one with a shallow decoder, the other with a deep decoder. Input $\mathbf{x}$ is encoded into latent representation $\mathbf{h}_2$ then decoded into reconstruction $\mathbf{y}$. The model is trained to minimize the difference between the corresponding, but different, input $\hat{\mathbf{x}}$ and reconstruction $\mathbf{y}$. Encoding layers are denoted by $\mathbf{h}_l$, and decoding layers by $\mathbf{z}_l$. Dashed red lines indicate differences that are minimized during training. Model parameters are shown for each layer. The networks depicted here have two encoder layers but deeper (and shallower) networks were also tested.

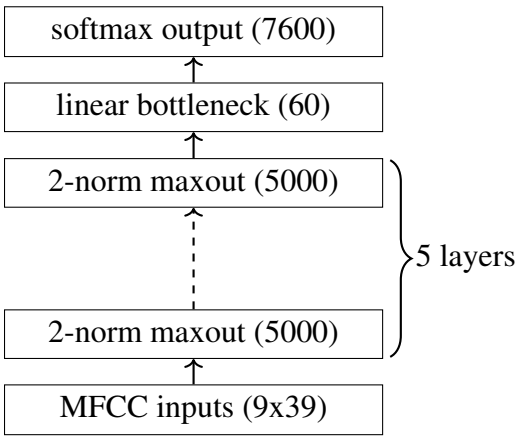


Figure 3.3: **Deep neural network**. A supervised model that learns to predict a context-dependent HMM state given 9 stacked MFCC frames. The bottleneck layer is used as our representation. Trained on English then applied without adaptation to Xitsonga. Network size depicted is that used in our experiments.

In addition to raw acoustic features as input to the various learning algorithms, we also evaluate the utility of data-driven features that exploit out-of-domain and/or out-of-language supervision. Specifically, we use BNFs extracted from a Kaldi speech recognition toolkit [70] trained DNN[3]. The DNN architecture takes a 9-frame context window of MFCCs as input to 5 hidden layers of 5,000 units (2-norm maxout non-linearity) followed by a linear bottleneck layer of 60 units (see [101] for details). The softmax output layer consists of 7600 clustered context-dependent HMM state targets. The DNN is trained using the Switchboard and Fisher English corpora, which amount to approximately 1,500 hours of English conversational telephone speech drawn from over 12,000 speakers. The resulting BNFs thus encode a detailed knowledge of the speaker-independent acoustic-phonetic structure of English, which we expect to transfer reasonably well into Buckeye despite the channel mismatch. However, as a representation for Xitsonga, any demonstrated improvement over the raw acoustic features would be derived from cross-lingual generalization of the encoded English knowledge.

Correspondence pairs for the Buckeye and Xitsonga corpora are extracted by the UTD system described in [47][3]. We use the graph clustering method of [22] to group individual discovered repetitions into term clusters from which we can derive more extensive transitive matches. In this way, we recover 11,041 token pairs for the ZRSC Buckeye portion (57% across-speaker) and 6,982 token pairs for Xitsonga (61% across-

[3]Thanks to Aren Jansen.

speaker).

3.3.2 Training

We have four sources of training data: (1) ZRSC Buckeye portion MFCCs, (2) Xitsonga MFCCs, (3) English BNFs, and (4) Xitsonga BNFs. In the spirit of using zero-resources, we demonstrate, in two ways, the effect of applying unsupervised models to domains and/or languages that differ from those used to optimize the architecture.

Our neural networks, implemented in Theano [4, 8], are trained via minibatch stochastic gradient descent backpropagation [11]. Weights and biases are random and zero initialized respectively. The training data is shuffled prior to each epoch of training. MFCC-based models include delta and double-delta features unless otherwise stated.

The "original" model architecture is identical to Kamper et al.'s 9×100-layer model [48] and is optimized for English Switchboard MFCCs on the *same-different* task and then trained with the data from one of the four sources. Nine encoders, each of size 100, are layerwise AE pre-trained over 30 epochs at a learning rate of 0.00025, and cAE fine-tuned over 120 epochs at a learning rate of 0.008, with minibatches of size 256. A single randomly initialized linear decoder is used during fine-tuning so we must select a representation layer; we report results using the Switchboard optimal layer, the 6th, and perform layerwise checks to determine the best layers for Buckeye.

The "optimal" model architecture is optimized for the ZRSC Buckeye portion MFCCs on the *ABX* task and then trained with the data from one of the remaining three sources. The optimal network structure, 5×13, was found by grid searching over number of layers (1, 3, 5, 9) and layer sizes (13, 39, 100). We also found that better results could be obtained by using a different training regime. Thus, the "optimal" architecture has 5 encoders, each of size 13, are layerwise AE pre-trained over 4 epochs at a learning rate of 0.1, and cAE fine-tuned over 320 epochs at a learning rate of 0.1, with minibatches of size 2048. Unlike the "original" architecture, the "optimal" architecture uses tied weights, a deep non-linear decoder (to avoid the layer selection problem), is trained using AdaGrad [23], and the correspondence pairs are presented in both directions. The use of AdaGrad allows the learning rate to be set to a single large value eliminating much of the advantage/cost of optimizing this hyperparameter.

We assume the optimal ratio between input size and hidden layer size is reasonably constant across datasets and models (backed up by informal experience from our collaborators' past uses of the cAE) and so use layer widths of 154 and 20 respectively

for the "original" and "optimal" models for the two BNF training data sources where the inputs are 60-dimensional.

The AE-only and dAE-only models are trained using the same training regime as the "optimal" models and their architectures are optimized via grid-search over layer widths and counts using the ZRSC Buckeye portion MFCCs. The optimal sizes are 1×13 and 1×200 for the AE and dAE respectively. The dAE was trained with $\gamma = 0.2$.

3.3.3 Evaluation

Following the ZRSC's protocol, we evaluate our frame representations using an *ABX* task [83, 84] which measures the discriminability of frame representations by asking whether triphone x is most like triphone a or triphone b, where a and x are distinct examples of the same triphone sequence and b is a triphone sequence differing from a and x in only the middle phone. More detail on this evaluation task can be found in Section 2.3.2.

3.3.4 Results

Our results are alphabetically labelled and presented in Table 3.1a (English) and Table 3.1b (Xitsonga); we focus our discussion on the more challenging across-speaker case. Comparisons are made to the ZRSC official baselines (MFCCs without delta or double delta features, `a` and `l`) and supervised toplines (Kaldi posteriorgrams and HMM-GMM, `b` and `m`) [94].

AE/dAE: As in previous work, we find that plain AEs (`c` and `n`) barely outperform MFCCs (`a` and `l`). dAEs (`d` and `o`) provide a bigger benefit, supporting previous work showing the importance of regularization in unsupervised representation learning, e.g. [96, 3]. Nevertheless, even dAEs do not match the performance of any cAEs, supporting Kamper et al.'s claim that guiding the representation learning using UTD pairs provides a major benefit over standard unsupervised methods. With further optimization (e.g. different types and levels of noise) better dAE results may be obtained which could be helpful in situations where correspondence pairs are unavailable.

"Original" cAE: Despite having its architecture optimized in a different domain, the "original" cAE (`e`) reduces Buckeye error rates compared to MFCCs by 17% relative. When we re-optimize for this domain (`f`), using the best representation layer for the Buckeye data (not a zero-resource result), the relative error rate reduction increases to 22%. These results are in line with previous work showing layer selection is important

English models	Within	Across
a Official baseline (13-dim MFCCs)	15.6	28.1
b Official topline (HMM-GMM)	12.1	16.0
c Optimal AE	16.9	28.6
d Optimal dAE	15.8	25.3
e Original cAE (Switchboard layer: 6th)	15.8	24.7
f Original cAE (Buckeye layer: 9th)	15.1	23.2
g Optimal cAE	**13.5**	**21.1**
h BNFs from English DNN	**12.8**	**18.1**
i Original cAE (Switchboard layer: 6th)	14.1	19.2
j Original cAE (Buckeye layer: 8th)	13.7	18.8
k Optimal cAE	14.0	19.3

(a) **English results**. Bold indicates per-section best results.

Xitsonga models	Within	Across
l Official baseline (13-dim MFCCs)	19.1	33.8
m Official topline (HMM-GMM)	3.5	4.5
n Buckeye optimized AE	17.4	29.5
o Buckeye optimized dAE	15.8	25.9
p Original cAE (Switchboard layer: 6th)	13.4	22.0
q Original cAE (Buckeye layer: 9th)	12.1	19.6
r Buckeye optimized cAE	**11.9**	**19.3**
s Xitsonga optimized cAE	11.6	18.5
t BNFs from English optimized DNN	14.4	19.3
u Original cAE (Switchboard layer: 6th)	14.1	19.0
v Original cAE (Buckeye layer: 8th)	13.1	17.8
w Buckeye optimized cAE	**13.0**	**18.2**

(b) **Xitsonga results**. Bold indicates best zero-resource results (architectures optimized on English data).

Table 3.1: **Minimal triphone pairs *ABX* within-/across- speaker error rates**. Top sections: official baseline (unsupervised) and topline (supervised). Middle sections: models of MFCCs including delta and double-delta features unless otherwise stated. Bottom sections: models of bottleneck features extracted from English trained DNN.

for getting the best results from a cAE that uses a single linear decoder during fine-tuning. The same architecture trained on Xitsonga has the same pattern of results but with larger relative error rate reductions of 26% and 34% (p and q) compared to the baselines (a and l). These latter cross-language results are encouraging evidence that the cAE could be applied productively to other zero-resource settings without fearing the architecture is especially sub-optimal.

"Optimal" cAE: Optimizing the cAE architecture on the Buckeye data (g) increases the error rate improvement from 17% (e) to 29% relative to the MFCC baseline (a). Clearly, the channel and task differences between Switchboard/*same-different* and Buckeye/*ABX* are significant. In the zero-resource case we find that the English improvements transfer to Xitsonga without any further optimization; the Buckeye-"optimal" cAE (r) reduces the Xitsonga error rate from the MFCC baseline by 35% relative. Optimizing the architecture on Xitsonga (s; a 9×13 architecture was best here) yields a small improvement but this is not a zero-resource result. The greater reductions achieved by the Buckeye/*ABX*-optimized architecture compared to the Switchboard/*same-different*-optimized architecture may be due to changes in architecture, to changes in training regime, or to optimizing for a different task.

DNN BNFs: Unsurprisingly, for English, the supervised BNFs (h) perform substantially better than the representations found by the unsupervised cAE (g). Furthermore, the cAE is unable to improve the English BNFs (i, j, k) suggesting the two training objectives are not complementary in the same-language setting. Pleasingly, applying the English DNN to Xitsonga (t) produces representations that perform just as well as the Buckeye-"optimal" cAE (r) representations. Our best zero-resource result is obtained by applying the "optimal" cAE to the Xitsonga BNFs (u, v, w) yielding representations of better quality than either approach achieves independently. By optimizing the Xitsonga cAE architecture a little we found that a narrower cAE network, 100 instead of 154, produced better results, reducing the error rate to 16.6% when using an "original" architecture, but this is not a zero-resource result.

More training data: Training the optimal cAE architecture on the entire Buckeye corpus, including a larger number of UTD pairs, did not yield distinctly different results from those obtained from using just the ZRSC Buckeye portion.

3.4 Summary

We have presented a selection of approaches for learning frame representations using unsupervised methods in zero-resource settings. Using a minimal triphone pair *ABX* discrimination task we showed that correspondence autoencoders (cAEs) outperform denoising autoencoders which outperform plain autoencoders. Although the cAE is a weakly supervised model, we obtain the correspondence data from a fully unsupervised term discovery system making our approach, taken as a whole, fully unsupervised. Compared to the original MFCCs, the cAE architecture optimized on Switchboard English reduces *ABX* error rates by 17% (relative) on Buckeye English and 26% on Xitsonga. Optimizing on Buckeye English instead yields a Xitsonga error rate reduction of 35%. These results demonstrate that our unsupervised system can be optimized in a high-resource setting, such as English, and then applied productively in a zero-resource setting such as Xitsonga.

We also found that applying the unsupervised cAE system to Xitsonga bottleneck features obtained from a supervised DNN trained on English yielded better results on Xitsonga than either system alone: an overall relative error rate reduction of 39% over MFCCs. This result suggests a promising future line of work combining supervised training in a high-resource language with unsupervised language adaptation to the zero-resource language.

We do not follow the high-resource thread further in this thesis. Instead, we propose a novel unsupervised model for improving the quality of frame representations further by optimizing speech unit discriminability explicitly, instead of implicitly using correspondence pairs (Section 4.2). The methods presented in this chapter used dynamic time warping (DTW), a quadratic-time algorithm, to compare sequences of frame representations; in Section 4.3 we propose a method that aims to speed up sequence comparisons by approximating DTW with a linear-time algorithm. We used the *ABX* task in this chapter to evaluate the discriminability of representations of subword units where those representations comprised sequences of frames; in Chapter 5 we propose models that can tackle the same task (or whole-term *same-different*) but where the representations are single fixed-size vectors, enabling more efficient comparison of sequences.

Chapter 4

Dynamic time warping related models

Dynamic time warping (DTW), introduced in Section 2.1, plays an important role in many unsupervised automatic speech recognition (ASR) systems where the minimal DTW alignment cost is used as a measure of sequence similarity. Unsupervised term discovery (UTD) systems are typically built on some version, or approximation, of DTW, and sequence similarity underpins speech indexing and retrieval systems.

All of the models in Chapters 4 and 5 require training data consisting of speech frame sequences; Section 4.1 introduces the methods we use to obtain suitable sequences in an unsupervised fashion. We go on to explore two DTW related topics in this chapter. In Section 4.2 we present a novel model that learns speech frame representations by optimizing DTW explicitly. In Section 4.3 we present a novel method for approximating DTW such that sequences can be compared in time proportional to the sum of their lengths instead of the product of their lengths. The triplet ranking loss introduced in Section 4.2 is used again in Section 5.4.

We ease future implementation by fully detailing each model, with particular emphasis on non-standard components. Section 4.2 does not contain any experimental results but we provide reasoned justifications for our hypotheses. We do likewise in Section 4.3 where results from a trial experiment are also presented.

4.1 Training data – sourcing sequences

We are interested in unsupervised representation learning – our training data is not pre-segmented into useful frame sequences. A raw speech corpus is typically segmented at dialogue or utterance boundaries. Ideally, we would train with segments at the level for which we wish to learn representations, e.g. phones, triphones, or whole terms. A voice

activity detection (VAD) algorithm [72] can be used to yield the most interesting portions of a corpus, removing silence and periods containing only uninteresting background noise, but the extracted segments are often still too long to be useful as they can contain many whole terms that flow into one another. A UTD system can go further and identify likely speech units, as was done for our frame-based models in Chapter 3. Another possibility is to sample random segments from the VAD output such that the distribution of lengths follows that of test-time sequences. Randomly sampled segments can be useful because:

1. The model presented in Section 4.3, and many of those in Chapter 5, do not require their training sequences to have a known or latent class (e.g. a triphone label). Instead, the models require data exhibiting sequential regularities and nothing more. Random segments will exhibit the sequential regularities of natural speech.

2. UTD systems are not perfect, often over- and under-segmenting. If our models are to be used with UTD generated sequences (e.g. after training, as part of larger unsupervised ASR system) then they must deal with extraneous material at the sequence ends. Random segments will usually contain noisy ends, [67, sec. III-C].

3. We can sample arbitrary quantities of random segments. Using a cross-entropy objective, as we do in this work, neural networks can often be trained just as effectively with larger quantities of low quality data as with smaller quantities of high quality data. The same cannot be said of sequence-discriminatively trained models [26].

4. The best of both worlds may be obtained by pre-training with large quantities of random segments and fine-tuning with UTD segments.

When using a pair loss (e.g. in a correspondence-based model) or a triplet loss (see the next section), we need a dataset consisting of paired sequences. A UTD system can be applied to identify all likely positive pairs but there is a quantity-quality trade-off. When UTD is unavailable, or undesirable, random sequences can be sampled and artificially corrupted to form pseudo-positive pairs. This method can generate arbitrarily large quantities of low quality training data which may yield a better quality representation function than using the small quantity of higher quality data typically provided by a UTD system. Artificial sequence corruption techniques are discussed in Section 5.5.

Negative pairs can be chosen randomly from all sequence pairs that are not identified as positive pairs but need to be sampled carefully. Given example $\mathbf{x}^1$, e.g. the triphone */kɪt/*, we must select another example $\mathbf{x}^3$ that does not belong to the same class, e.g. a triphone that shares the same first and third phones but has a different middle phone such as */kæt/*. Sampling uniformly from all examples that meet these conditions may not be sufficient; the speaker identity may also need to be taken into account[1]. In many cases, positive pairs will be utterances from the same speaker while the overwhelming majority of possible negative pairs will be utterances from different speakers. It may be easier for the network to learn whether the speaker is the same or different, than to learn whether the speech unit is the same or different. To prevent this undesirable learning behaviour we must take the speaker identity into account when sampling negative pairs, preferring negative pair utterances to be from the same speaker. The number of valid negative examples from the same speaker is often small; we propose sampling $\mathbf{x}^3$ from the union of valid same-speaker and different-speaker sets but disproportionately biasing towards the same-speaker examples.

4.2 Optimizing DTW explicitly

In unsupervised speech processing, standard tasks and evaluation use DTW distance as a measure of the degree of similarity between sequences of frame representations. A good representation is one that minimizes the DTW distance between sequences of frames representing different utterances of the same speech unit and maximizes the DTW distance between sequences of frames representing utterances of different speech units. The correspondence autoencoder (cAE, Section 3.2) learns representations that achieve this goal implicitly – frame representations are encouraged to be similar when they appear in the same relative position of sequences representing the same unit because the *input* frames are aligned using DTW. However, alignment of the input representations can be suboptimal due to the extraneous information contained in the source data – the noise we wish to eliminate from the representations. We do not care about the alignment of the input representations but we do care about the alignment of the output (i.e. learnt) representations since that is what is used in our evaluations and downstream tasks. We hypothesize that learning frame representations to explicitly minimize the alignment cost will yield a better quality representation function than is achieved by minimizing the alignment cost implicitly via a method such as a cAE.

[1] This observation is due to Herman Kamper in private correspondence.

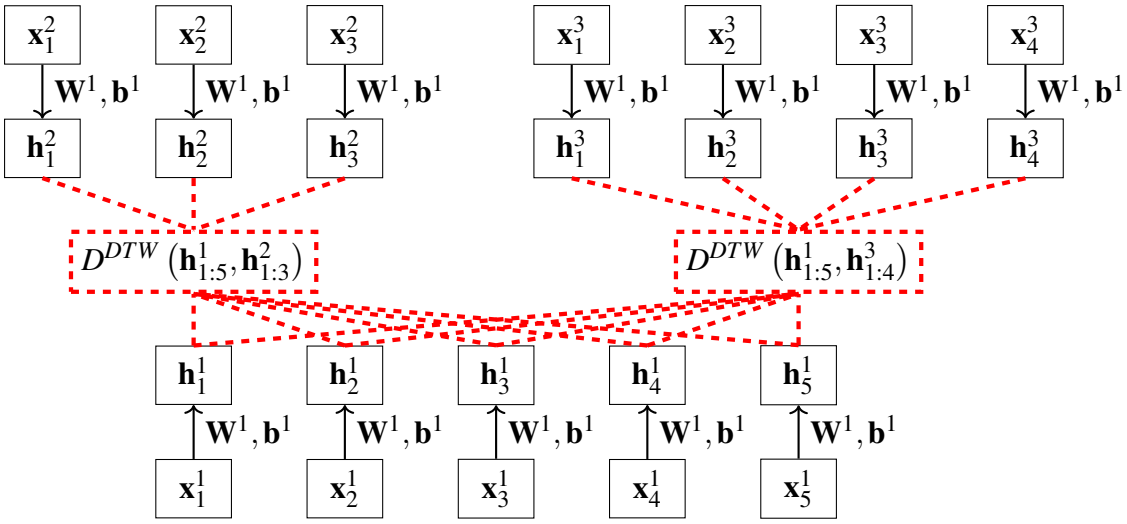


Figure 4.1: **Optimizing DTW explicitly**. The frames of sequences $\mathbf{x}_{1:5}^1$, $\mathbf{x}_{1:3}^2$, and $\mathbf{x}_{1:4}^3$ are encoded into latent representation sequences $\mathbf{h}_{1:5}^1$, $\mathbf{h}_{1:3}^2$ and $\mathbf{h}_{1:4}^3$ respectively. $\mathbf{x}_{1:5}^1$ and $\mathbf{x}_{1:3}^2$ are different utterances of the same speech unit and $\mathbf{x}_{1:4}^3$ is an utterance of a different speech unit. The training objective is to minimize the DTW distance between the positive pair whilst ensuring that the the DTW distance between the negative pair is at least a margin greater; the precise loss function is given in Equation 4.2. Every frame is encoded in the same way (the parameters $\mathbf{W}^1$ and $\mathbf{b}^1$ are used at every position of every sequence). Only one layer is shown here but many layers may be used in general.

We can minimize the alignment cost between all pairs of sequences $\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}\right)$ representing different utterances of the same speech unit, i.e all positive pairs, using the loss in Equation 4.1.

$$\mathcal{L}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}\right) = D^{DTW}\left(f_\theta\left(\mathbf{x}^1_{1:L^1}\right), f_\theta\left(\mathbf{x}^2_{1:L^2}\right)\right) \tag{4.1}$$

f_θ is a sequence-to-sequence function that transforms the input into a representation parametrized by θ. In principle the output sequences may differ in length from their corresponding inputs but we focus on the case where f_θ generates a sequence of one-to-one frame representations.

The loss in Equation 4.1 is inadequate: mapping every frame to the same point in the latent representation space (e.g. the zero vector) would trivially minimize the objective. It also fails to encourage different-class sequence comparisons to have a large DTW alignment cost. To counter these problems we propose using a *triplet loss* (Equation 4.2) as in Thiolliere et al. [91] and Hadsell et al. [35]. Here, $\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}\right)$ is a positive pair and $\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^3_{1:L^3}\right)$ is a negative pair, i.e. $\mathbf{x}^1_{1:L^1}$ and $\mathbf{x}^3_{1:L^3}$ represent utterances of different speech units. Equation 4.2 is also a *ranking loss*: θ is considered optimal if the positive pair DTW distance is always at least margin α smaller than the negative pair DTW distance.

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}, \mathbf{x}^3_{1:L^3}\right) = \\ \max\left\{0, D^{DTW}\left(f_\theta\left(\mathbf{x}^1_{1:L^1}\right), f_\theta\left(\mathbf{x}^2_{1:L^2}\right)\right) - D^{DTW}\left(f_\theta\left(\mathbf{x}^1_{1:L^1}\right), f_\theta\left(\mathbf{x}^3_{1:L^3}\right)\right) + \alpha\right\}\end{aligned} \tag{4.2}$$

Equation 4.2 differs from the loss used in the correspondence autoencoder approach of Chapter 3 because the distance measure is between whole sequences, not individual frames, and it is the output of the function that is being aligned during training, not the input.

Training the parameters θ of neural network representation function f_θ, using stochastic gradient descent with our proposed loss function, requires propagating gradients through the DTW computation. This is possible because the DTW function is composed entirely of differentiable operations. Equation 4.3 is a recursive definition of DTW distance between sequence-pair $(\mathbf{u}_{1:L^u}, \mathbf{v}_{1:L^v})$ combined with the frame-pair cosine distance. This function can be computed efficiently using the conventional

dynamic programming technique.

$$D^{DTW}\left(\mathbf{u}_{1:i},\mathbf{v}_{1:j}\right)=\begin{cases}\sum_{k=1}^{i}D^{cos}\left(\mathbf{u}_k,\mathbf{v}_1\right) & \text{if } j=1\\ \sum_{k=1}^{j}D^{cos}\left(\mathbf{u}_1,\mathbf{v}_k\right) & \text{if } i=1\\ D^{cos}\left(\mathbf{u}_i,\mathbf{v}_j\right)+\min\left\{\begin{array}{l}D^{DTW}\left(\mathbf{u}_{1:i-1},\mathbf{v}_{1:j}\right),\\ D^{DTW}\left(\mathbf{u}_{1:i},\mathbf{v}_{1:j-1}\right),\\ D^{DTW}\left(\mathbf{u}_{1:i-1},\mathbf{v}_{1:j-1}\right)\end{array}\right\} & \text{otherwise}\end{cases} \tag{4.3}$$

where

$$D^{cos}\left(\mathbf{u}_i,\mathbf{v}_j\right)=\frac{1}{2}\left(1-\frac{\mathbf{u}_i\mathbf{v}_j}{\|\mathbf{u}_i\|_2\left\|\mathbf{v}_j\right\|_2}\right)\in[0,1] \tag{4.4}$$

The gradient of the min operation in Equation 4.3 can be computed using the recursive definition $g_\theta^K=\min\left\{c_\theta^1,c_\theta^2,\ldots,c_\theta^K\right\}=\min\left\{g_\theta^{K-1},c_\theta^K\right\}$ and Equation 4.5. The gradient of g_θ^K is technically undefined when $g_\theta^{K-1}=c_\theta^K$ but we follow common practice and arbitrarily use $\frac{\partial c_\theta^K}{\partial\theta}$ in that case.

$$\frac{\partial g_\theta^K}{\partial\theta}=\begin{cases}\frac{\partial c_\theta^1}{\partial\theta} & \text{if } K=1\\ \frac{\partial c_\theta^K}{\partial\theta} & \text{if } c_\theta^K\leq g_\theta^{K-1}\\ \frac{\partial g_\theta^{K-1}}{\partial\theta} & \text{otherwise}\end{cases} \tag{4.5}$$

Our implementation of DTW, using Theano [4, 8], is available online[2]. In this implementation the gradients are derived using Theano's automatic differentiation feature. DTW itself cannot be trivially parallelized but our Theano implementation enables some degree of parallelization by enabling many DTW alignments, i.e. a minibatch, to be computed at once.

We propose to use a multi-layer perceptron frame representation function, as defined in Equations 4.6 and 4.7 with parameters $\theta=\left\{\left(\mathbf{W}^l,\mathbf{b}^l\right)\mid l\in[1,\ldots,H]\right\}$, where H is the number of layers. This approach is similar to the encoder portion of the models used in the frame-based approaches of Chapter 3 and is depicted in Figure 4.1 with a single layer.

$$f_\theta\left(\mathbf{x}_{1:L}\right) = \mathbf{h}_{1:L}^H \tag{4.6}$$

$$\mathbf{h}_{1:L}^l = \begin{cases}\mathbf{x}_{1:L} & \text{if } l=0\\ \text{relu}\left(\mathbf{W}^l\mathbf{h}_{1:L}^{l-1}+\mathbf{b}^l\right) & \text{otherwise}\end{cases} \tag{4.7}$$

[2] https://github.com/danielrenshaw/TheanoBatchDTW

We propose using the element-wise rectified linear unit (ReLU) activation function, $relu(\mathbf{a}) = \max\{\mathbf{0}, \mathbf{a}\}$ which was the approach taken by the successful frame-based Siamese networks of Thiolliere et al. [91]. Deep networks trained with ReLU activation functions can be trained efficiently without layer-wise pre-training because the ReLU gradient is piecewise constant and does not saturate for large positive inputs.

This model explicitly optimizes representations to exhibit the desired behaviours. More work is required to determine whether the representation function parameters θ can be optimized efficiently when gradients are propagated through the DTW dynamic program, and in the presence of low quality sequence pairings.

4.3 Linear time DTW approximation

The asymptotic time complexity of DTW is a function of the *product* of the sequence lengths, i.e. $O(MN)$. When aligning a large number of long sequences, the computation cost of DTW can become a bottleneck in some systems, e.g. Kamper et al. [49]. In many cases the DTW distance is of interest but not the alignment itself. Furthermore, an approximate distance may be acceptable or we may only require a ranking of sequence pair similarities where even the absolute DTW distance values are of little interest. This final scenario is applicable in many unsupervised speech processing systems.

In this section we present a model that can learn to approximate DTW such that, after training, sequences can be compared with an asymptotic time complexity that is a function of the *sum* of the sequence lengths, i.e. $O(M+N)$.

A recent result from Backurs and Indyk [2] suggests that an optimal sub-quadratic time algorithm for computing the edit distance between two strings cannot exist if $P \neq NP$. The edit distance and DTW algorithms are very similar so this result may also apply to DTW making a linear time algorithm necessarily approximate. An approximate DTW function would be acceptable for our purposes if the accuracy of sequence similarity rankings remains high.

4.3.1 Training and use

Figure 4.2 depicts the proposed training procedure and potential subsequent uses. Sequence pairs are extracted from the training dataset. The pairings may be meaningful (e.g. from sequence labels or from a UTD system) but we hypothesize that training with arbitrary sequences would be more effective. Unlike the model in Section 4.2 we

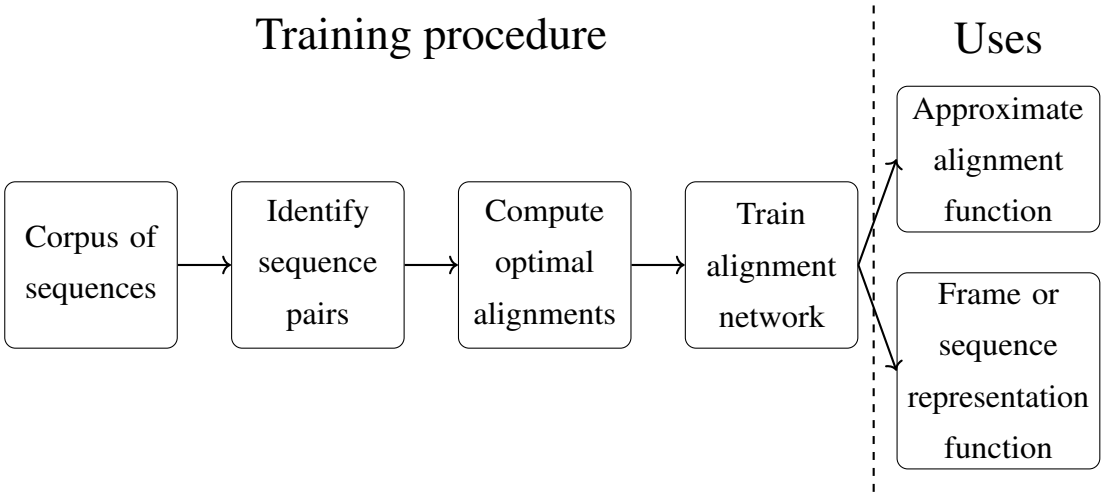


Figure 4.2: **Linear time DTW approximation – training procedure and uses**. Sequence pairs are extracted from the training corpus. The model is trained to predict the optimal DTW alignment paths. The trained model can be used to predict alignment paths for novel sequence pairings, or the lower level encoder can be used as a frame or sequence representation function.

are not training the model in this section to discriminate between different speech units. The evaluation and downstream tasks for our linear time DTW approximation model involve aligning arbitrary sequence pairs so training with arbitrary pairs will replicate the test-time use. Also, training with only positive pairs may yield a model that does not behave appropriately given negative pairs.

Alignment paths between every training sequence pair are computed using DTW. The resulting optimal paths are used as the supervision signal for training our model. Our model learns to approximate the globally optimal DTW path via a sequence of local alignment decisions. A decision is made at each position given only sequence prefix and suffix summaries.

The trained model can be used in at least two ways:

1. **As an approximate alignment function**. Given two frame sequences, the model can predict an alignment that approximates the optimal DTW alignment. The predicted alignment path or cost can be used in downstream tasks, e.g. as a measure of sequence similarity.

2. **As a frame or sequence representation function**. The frame embedding layer can be used as a frame representation function or the bidirectional recurrent neural network (BRNN) layer can be used as a whole-sequence representation function.

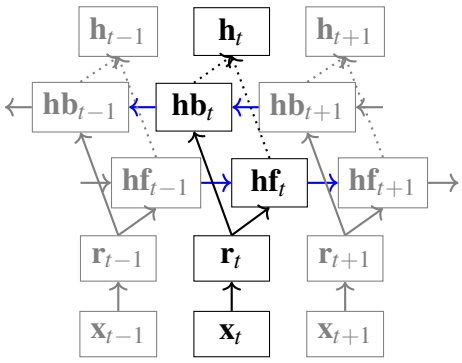


Figure 4.3: **Linear time DTW approximation – bidirectional recurrent neural network**. This figure depicts a general slice through the lower half of our linear time DTW approximation model as applied to one of the input sequences. In general, a bidirectional recurrent neural network is applied to the two input sequences, $\mathbf{x}^1_{1:L^1}$ and $\mathbf{x}^2_{1:L^2}$, independently yielding two sequences of hidden states, $\mathbf{h}^1_{1:L^1}$ and $\mathbf{h}^2_{1:L^2}$. Each hidden state $\mathbf{h}^i_t$ combines a representation of the prefix and suffix of sequence i. Solid black lines indicate normal feed-forward connections, blue lines indicate recurrent connections, and dotted black lines indicate elements that are simply concatenated together.

This model can only deliver value as an approximate alignment function if the training time is offset by sufficient use of the trained model.

4.3.2 Model

Our model is composed of two halves. A slice through the lower half, a BRNN, is depicted in Figure 4.3. A separate BRNN is applied to each of the input sequences $\mathbf{x}^i_{1:L^i}$ independently (for $i \in \{1,2\}$), yielding two sequences of hidden states $\mathbf{h}^i_{1:L^i}$. Each hidden state $\mathbf{h}^i_t$ contains information about the sequence prefix up to position t and the sequence suffix down to position t. We hypothesize that the prefix and suffix information is sufficient for the model to make local alignment decisions that yield an accurate

approximation of the globally optimal DTW path.

$$\mathbf{r}_t^i = f\left(\mathbf{W}^r \mathbf{x}_t^i + \mathbf{b}^r\right) \tag{4.8}$$

$$\mathbf{hf}_t^i = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^{hf}\left[\mathbf{hf}_{t-1}^i; \mathbf{r}_t^i\right] + \mathbf{b}^{hf}\right) & \text{otherwise} \end{cases} \tag{4.9}$$

$$\mathbf{hb}_t^i = \begin{cases} \mathbf{0} & \text{if } t = L^i + 1 \\ f\left(\mathbf{W}^{hb}\left[\mathbf{hb}_{t+1}^i; \mathbf{r}_t^i\right] + \mathbf{b}^{hb}\right) & \text{otherwise} \end{cases} \tag{4.10}$$

$$\mathbf{h}_t^i = f\left(\mathbf{W}^h\left[\mathbf{hf}_t^i; \mathbf{hb}_t^i\right] + \mathbf{b}^h\right) \tag{4.11}$$

Frame representations $\mathbf{r}_t^i$ are learnt for inputs $\mathbf{x}_t^i$. Equation 4.8 gives a non-linear frame representation function, whose parameters are learnt jointly with the rest of the model. We propose to use the tanh activation function for f.

The forward and backward passes of the BRNN, $\mathbf{hf}_t^i$ and $\mathbf{hb}_t^i$, are given in Equations 4.9 and 4.10 respectively. Here we use simple RNN layers but LSTM or GRU layers could be used instead. The forward and backward states are combined into a single state $\mathbf{h}_t^i$ (Equation 4.11) for use in the upper half of the model.

If needed, we can form fixed-size representations of the sequences by concatenating the final hidden states in each direction, $\mathbf{r}^i = \left[\mathbf{hf}_L^i; \mathbf{hb}_1^i\right]$.

An alignment between the the two hidden state sequences is predicted in the upper half of the model, depicted in Figure 4.4. This is a novel neural network structure that learns to align the inputs by making a sequence of local alignment decisions. The network parameters are optimized to minimize the difference between the model's predicted local alignment decisions and the globally optimal target alignment decisions.

The first frames in the two input sequences are force aligned and sequence position state variables p_t^1 and p_t^2 are initialized to equal 1. At each position the model decides which of the two sequences to advance; it may choose to advance either sequence alone, or advance both sequences, but may not choose to advance neither sequence. The specific hidden states used to make this decision are determined by the current position state variables p_t^1 and p_t^2.

$$\mathbf{y}_t = \text{softmax}\left(\mathbf{W}^y\left[\mathbf{h}_{p_{t-1}^1}^1; \mathbf{h}_{p_{t-1}^2}^2\right] + \mathbf{b}^y\right) \tag{4.12}$$

$$\hat{y}_t = \arg\max_j y_{t,j} \tag{4.13}$$

$$p_t^i = p_{t-1}^i + \begin{cases} 1 & \text{if } \hat{y}_t = i \text{ or } \hat{y}_t = 3 \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

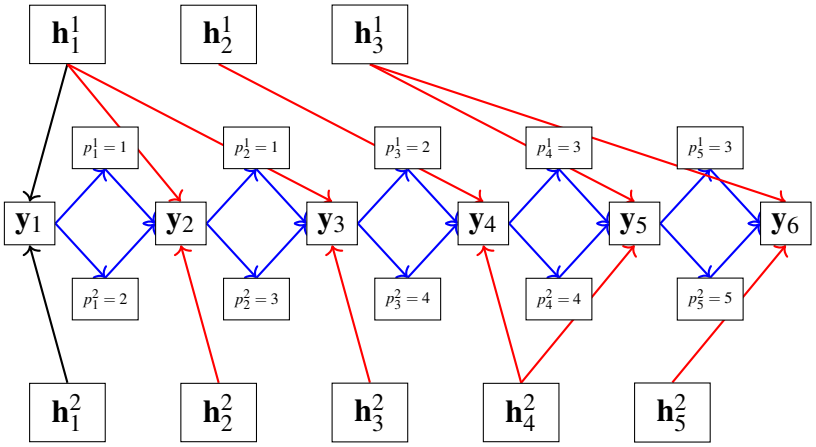


Figure 4.4: **Linear time DTW approximation – alignment path prediction**. This figure depicts a full example of the upper half of our linear time DTW approximation model as applied to sequences of lengths 3 and 5. This model component predicts an alignment between the two input sequences by making a series of local alignment decisions. Each position in the alignment path $\mathbf{y}_t$ is a distribution over three possible actions: advance sequence 1, advance sequence 2, advance both sequences. The current position in each sequence is stored in the index variables p_t^1 and p_t^2. The input sequences are generated by the lower half of the model depicted in Figure 4.3. Black lines indicate the forced alignment of the initial frames, red lines indicate the states used at each alignment decision (determined by all previous alignment decisions), and blue lines indicate recurrent links.

$\mathbf{y}_t \in \Delta^2$ (Equation 4.12) is the predicted probability distribution over the three possible actions at position t. The soft local alignment decisions are hardened into $\hat{y}_t \in \{1,2,3\}$ (Equation 4.13) to determine which position state variable(s) to increment (Equation 4.14).

By using a hard decision at each position we are not guaranteed to obtain the path that the model prefers globally. Beam search may be applied to decode many competing paths concurrently but this affects the computational complexity of the model. With beam width B, the computational complexity increases from $O(N+M)$ to $O(B(N+M))$.

During training the model is forced to predict a path that is of the same length T as the globally optimal path found by DTW. The predicted path is compared to the globally optimal path using a cross-entropy loss (Equation 4.15) where $z_t \in \{1,2,3\}$ is the correct alignment decision at position t along the alignment path.

$$\mathcal{L}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}\right) = -\frac{1}{T}\sum_{t=1}^{T} \log\left(y_{t,z_t}\right) \tag{4.15}$$

At test time we do not know the alignment path length so iterate the model until both input sequences are exhausted. To prevent the model making invalid decisions the unexhausted sequence is forced to advance once the other sequence has become exhausted.

An alternate training procedure would adopt the test time behaviour and apply a weaker supervision signal. Instead of comparing the alignment decisions at every position along the alignment path (which may now be of a different length to the globally optimal path) we instead compare only the final alignment cost with the globally optimal alignment cost. This approach more closely matches many unsupervised speech processing scenarios where we care about the alignment cost but not the alignment path. However, we hypothesize that comparing the alignment cost alone in the loss will not be a strong enough training signal to optimize the model parameters efficiently.

Our model can generalize to the task of aligning more than two sequences. This is an important task in a variety of fields including bioinformatics where we are often interested in finding the optimal alignment between multiple sequences of DNA. Optimal multiple sequence alignment has time complexity that is exponential in the number of sequences. Our model can be applied such that the time complexity is linear in the number of sequences. To achieve this, the output layers $\mathbf{y}_t$ must be altered. The number of combinations of sequences that could be advanced at each position, and thus the size of the output layers, is exponential in the number of sequences. The exponential

blow-up can be avoided by using a separate logistic output layer for each sequence. The model predicts whether each sequence should be advanced independently of all other sequences given the current hidden states. The loss function is then a sum of binary cross-entropy terms. A disadvantage of this approach is that it becomes possible for the model to make invalid predictions, i.e. "advance none of the sequences". At training time this can be ignored and penalized naturally. At test time this would stall decoding so we propose always advancing at least the single sequence whose predicted advance probability is highest, even if less than 0.5.

4.3.3 Trial experiment results

To test the feasibility of our proposed model we run a trial experiment using textual data and a spelling correction task. This scenario is simpler than using speech data with an *ABX* or *same-different* task making it more appropriate for feasibility trials. We use sequences of characters that form plain English words. These sequences are short discrete representations making them quite different from the long continuous representations we use for speech data (e.g. MFCCs).

For this setup, the edit distance algorithm provides optimal alignments. Edit distance is computed using dynamic programming, like DTW, and the optimal alignment is defined as the minimal cost edit operation sequence that turns the first sequence into the second sequence. We use different costs for each of the edit operations: insertion has cost 1.1, deletion has cost 1, and substitution has cost 1.5.

Edit distance is similar in spirit and implementation to DTW. The most important difference is that DTW operates on continuous multivariate data while the edit and Levenshtein distances require discrete data. A more subtle difference is that DTW force aligns the first pair of elements while the edit and Levenshtein distances allow initial elements in one sequence to remain unaligned.

Our training data consists of the 5,000 most frequently used plain words in Moby Dick (i.e. tokens consisting of *[a-z]+* only) which are artificially corrupted (insertion rate: 10%, deletion rate: 15%, substitution rate: 5%). 10 misspellings are generated for each original word. The resulting dataset has the following characteristics: sequence pair count: 50,000; identical pairs: 14.2%; mean word length: 6.5 characters; mean misspelled word length: 6.3 characters; mean optimal edit distance path length: 7.1 aligned characters; mean edit distance cost: 2.2. 10% of the sequence pairs are held out for evaluation. Each character is represented by a 1-hot 26-dimensional vector (a binary

vector containing a single 1 at the position corresponding to the letter of the alphabet being represented).

We use a character embedding layer of size 5 and LSTM recurrent layer of size 20. The model has 4,698 parameters. We train the model using mini-batch stochastic gradient descent with a batch size of 50 and learning rate of 0.1 over 10 epochs.

The trained model's predicted alignment paths exactly match the optimal edit distance alignment paths for 32.2% of the test set's sequence pairs. 16.8% of the predicted paths were too short, 5.6% too long, and 45.4% were the correct length but advanced the wrong sequences in at least one position. Of the 77.6% of predicted paths that had the correct length the following can be observed: correctly spelled and correctly aligned: 17.8%; correctly spelled and incorrectly aligned: 0%; incorrectly spelled and correctly aligned: 23.7%; incorrectly spelled and incorrectly aligned: 58.5%; cost identical: 53.1%.

An approximation of spelling mistake correction performance is evaluated by computing a distance between every original word and every misspelling in the test set, ranking those distances, and computing the average precision. This is a difficult task as illustrated by the optimal edit distance achieving an average precision of 36.8%. Our model achieves an average precision of 10.0%. Further work is required to determine whether substantial improvements can be obtained by optimizing the hyperparameters and improving the training regime. For example, only positive pairs were used to train the model and better results may be obtained by using arbitrary sequence pairs.

It is encouraging that an unoptimized model can predict a path of the correct length in over 77% of cases and, of those, correctly predict the cost in over 50% of cases. The model clearly learns something useful: prior to training, with random parameters, 0.7% of the predicted paths are fully correct and 1.1% have the correct length. The average precision results are mildly disappointing; more work is needed for error analysis to understand where the model is performing well, and where it fails.

4.4 Summary

Dynamic time warping (DTW) is an important technology for unsupervised automatic speech recognition (ASR). In this section we explored two aspects of DTW related to our interest in unsupervised representation learning for ASR.

We hypothesize that optimizing the frame representations to minimize/maximize DTW explicitly will yield a better representation function than can be achieved using

a correspondence autoencoder (cAE). Minimizing/maximizing DTW is the behaviour we need in our evaluation and downstream tasks when the two sequences are from the same/different classes respectively. In contrast, the cAE achieves the desired behaviour indirectly and, potentially, inefficiently. Our proposed training loss function explicitly encourages same-class sequence pairs to have a smaller DTW alignment distance then different-class sequence pairs.

In systems that rely on performing large numbers of long sequence comparisons using DTW, but where the optimal alignment path is not required, one may use our novel linear time DTW approximation model. The time complexity of our approach is $O(N+M)$, i.e. proportional to the sum of the sequence lengths, while the optimal DTW algorithm has time complexity $O(NM)$, i.e. proportional to the product of the sequence lengths. Our linear time DTW approximation model may also be used to learn frame or sequence representation functions. However, we expect these representations to be poor in comparison to those learnt by the first model of this chapter because the latter is learning to align the original (noisy) inputs while the former is learning representations that explicitly optimize the downstream alignments we care about.

It is natural to consider the possibility of combining together the two models presented in this chapter: using the DTW approximation model in place of optimal DTW in Equation 4.2. Though this may be possible, it is difficult to envisage a circumstance where the joint model could be of any more use than the two models used separately. It is not clear that a representation function optimizing DTW explicitly could only yield representations enabling a better quality approximation of DTW if they are trained jointly. If we want to optimize DTW explicitly, and also have a fast DTW approximation for use at test-time, then we can simply train the model in Section 4.2 first then train the model in Section 4.3 using the optimized representations as input.

Chapter 5

Learning representations of speech frame sequences

Speech signals are typically represented by sequences of frames where frames are produced at a constant rate, e.g. 100 Hz (i.e. every 10 ms), and each frame represents a fixed period of the signal, e.g. 25 ms. Human speech productions vary considerably in their duration, even for different utterances of the same speech unit. We are interested in the similarity of frame sequences but have no control over the lengths of those sequences. A representation function for individual frames is useful, as demonstrated in Chapter 3, but comparisons between more meaningful units, such as triphones or whole terms, requires the use of a method like dynamic time warping (DTW) to find the optimal alignments between arbitrary length sequences. DTW can become a bottleneck in systems that rely on large numbers of sequence comparisons, as discussed in Chapter 4. Linear time DTW approximations, such as the approach described in Section 4.3, offer one potential solution to this problem. An alternative solution is to avoid alignment-based methods altogether and instead learn to represent entire speech sequences by fixed size vectors. Sequences with fixed size representations can be compared efficiently using a single vector space comparison such as cosine similarity.

We hypothesize that fixed size representation functions are feasible and can usefully represent variable length speech segments because:

1. We assume the sequences of interest can be represented using finite length sequences of discrete symbols drawn from a finite vocabulary. For example, the variable length frame sequences representing every possible utterance of the triphone */kæt/* can be represented by the fixed length symbol sequence [*k*, *æ*, *t*]. As long as our fixed size representations have sufficient representational capacity

to cover every possible (valid) sequence of symbols then they could, in principle, replicate the symbolic representation.

2. We are interested only in the relative similarity of sequences; a frame alignment is unnecessary. Finding the optimal alignment path is what makes DTW slow but, we hypothesize, finding an explicit alignment path is unnecessary for determining sequence similarity.

3. Standard DTW uses a hard alignment – each frame in one sequence is associated entirely with one or more frames in the other sequence. A hard alignment may not be appropriate for our data where we might reasonably expect, for example, a frame to be associated 60% with one frame, 30% with the previous frame, and 10% with the next frame, i.e. a soft alignment. Any information lost in DTW's hard alignment may still be available to a sequence representation function that is not based on DTW.

4. Prior work has shown fixed size representations can be both meaningful and useful. For example, the recurrent neural network (RNN) encoder-decoder architecture (RNN-ED, Section 5.3) has been applied successfully in large vocabulary speech recognition [57] and machine translation [17]. Variable length sentences are represented by a fixed size vector and, from that vector alone, a translation of the original sentence is produced in a different language. The entire meaning of the arbitrary length source sentence must be compressed into the fixed size representation for the target sentence to be generated accurately.

In this chapter we present a selection of methods for learning fixed sized representations of variable length sequences using RNN architectures. Section 5.2 pairs a novel training objective for speech data (sequence reversal) with the standard bidirectional RNN (BRNN) to partially avoid the hypothesised cause of poor representations produced by the simple RNN with a predict-next training objective presented in Section 5.1. We propose applying the standard RNN-ED to our fixed-size representation learning task in Section 5.3 to overcome the problems faced by the previous models but highlight a hypothesised flaw due to a training objective that does not closely match the similarity-oriented task. We hypothesise that this final flaw can be overcome by using a triplet ranking loss with a novel Siamese RNN (SRNN), presented in Section 5.4.

All models in this chapter require training data consisting of frame sequences. The sequences may consist of meaningful segments of speech, e.g. from unsupervised term

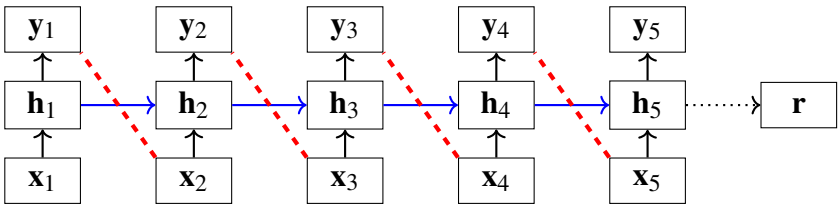


Figure 5.1: **Simple RNN with predict-next objective**. At each position the network updates its internal state $\mathbf{h}_t$ given the current input $\mathbf{x}_t$ and makes a prediction $\mathbf{y}_t$ for what the next input will be. An example is shown for a sequence of length 5. Black lines indicate normal feedforward connections, blue lines indicate recurrent connections (depicted unwound), dotted lines indicate how the sequence representation is constructed, and dashed red lines indicate differences we minimize during training.

discovery (UTD), or may be randomly sampled. Both the RNN-ED in its supervised sequence translation mode, and the SRNN, require meaningful sequences. The other models could operate with either meaningful or random sequences. Section 4.1 describes how sequences can be sourced from an unsegmented speech corpus. Related to the method for sourcing training sequences is the topic of artificially generating additional data for the purpose of regularization; this topic is covered in Section 5.5.

We ease future implementation by fully detailing each model, with particular emphasis on non-standard components. This chapter does not contain any experimental results. We provide reasoned justifications for our hypotheses, motivated in some cases by inconclusive trial experiments.

5.1 Simple RNN with predict-next objective

Perhaps the simplest method for learning unsupervised fixed size representations of variable length sequences using an RNN is to apply a predict-next objective. At each position in the sequence the model updates its internal state given the current input and predicts the next input given the updated hidden state. The model parameters are optimized to minimize the difference between the predicted next inputs and the actual next inputs. This is a common approach for neural network language models [60] where the predict-next objective closely matches the evaluation method and downstream tasks.

For input sequence $\mathbf{x}_{1:L}$, the simplest version of this model is defined in Equations 5.1

through 5.4, and an example is depicted in Figure 5.1 with $L = 5$.

$$\mathbf{h}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^h \left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}^h\right) & \text{otherwise} \end{cases} \tag{5.1}$$

$$\mathbf{y}_t = \mathbf{W}^y \mathbf{h}_t + \mathbf{b}^y \tag{5.2}$$

$$\mathbf{r} = \mathbf{h}_L \tag{5.3}$$

$$\mathcal{L}\left(\mathbf{x}_{1:L}\right) = \sum_{t=1}^{L-1} D^{sq\_euc}\left(\mathbf{y}_t, \mathbf{x}_{t+1}\right) \tag{5.4}$$

The recurrent layer's activation function f is typically the hyperbolic tangent or logistic function. A linear activation function is used in the output layer because the targets, i.e. MFCCs, have unbounded values. The squared Euclidean distance $D^{sq\_euc}\left(\mathbf{y}_t, \mathbf{x}_{t+1}\right) = \|\mathbf{y}_t - \mathbf{x}_{t+1}\|_2^2$ is used in the loss to measure the discrepancy between predicted next input $\mathbf{y}_t$ and actual next input $\mathbf{x}_{t+1}$.

The final hidden state is used as the representation of the whole sequence (Equation 5.3) but the loss function does not affect the final hidden state so an end-of-sequence pseudo-element (e.g. a zero vector) must be appended to every sequence to ensure that the final state incorporates information about every real element in the sequence.

This network cannot be made bidirectional because the objective would become trivial to solve – the next hidden state in the backward direction can precisely determine the prediction in the current position for the next position. A GRU or LSTM layer could replace the simple RNN layer used in Equation 5.1, as described in Section 2.4.5.

Our primary goal is the fixed size sequence representation $\mathbf{r}$ but the model may also yield improved element representations if the network is enriched with an additional layer by replacing Equation 5.1 with Equations 5.5 and 5.6. The frame representation function $\mathbf{r}_t$ can be used just like those presented in Chapter 3.

$$\mathbf{r}_t = f\left(\mathbf{W}^r \mathbf{x}_t + \mathbf{b}^r\right) \tag{5.5}$$

$$\mathbf{h}_t = f\begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^h \left[\mathbf{h}_{t-1}; \mathbf{r}_t\right] + \mathbf{b}^h\right) & \text{otherwise} \end{cases} \tag{5.6}$$

The main criticism of this model is that the hidden state is not encouraged to summarise the entire prefix by the training objective. Instead, the network only requires the hidden state to retain enough information about the past for it to correctly predict the next item in the sequence at the current and all following positions. In principle, the network can forget information once it believes it has become irrelevant for predicting future inputs. This is explicit when using a GRU or LSTM recurrent layer and implicit

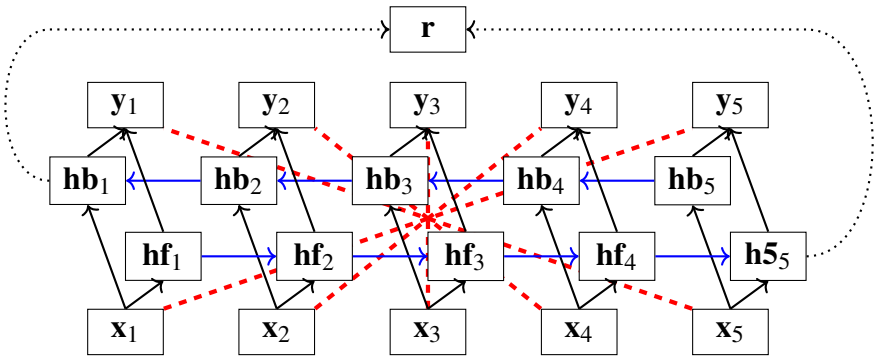


Figure 5.2: **Bidirectional RNN with sequence reversal objective**. An example of using a bidirectional RNN to learn to reverse sequences. The task can only be achieved if information about the start of the sequence is accurately retained via the forward hidden state to the end of the sequence, and similarly for the backward direction. This suggests the fixed size sequence representation will contain a useful summary of the whole sequence. Black lines indicate normal feedforward connections, blue lines indicate recurrent connections (depicted unwound), dotted lines indicate how the sequence representation is constructed, and dashed red lines indicate differences we minimize during training.

when using a simple RNN layer. For example, if our data is generated by productions $A \rightarrow B$, $B \rightarrow A \mid C$, $C \rightarrow D$, and $D \rightarrow C$ then, starting from A, the network could learn that once it sees a C there is no longer any need to remember anything about As or Bs. This potential behaviour suggests that the final hidden state may not be a good representation of the sequence as a whole.

5.2 Bidirectional RNN with sequence reversal objective

We hypothesize that pairing a BRNN with a novel training objective can offer a partial solution to the above-mentioned flaw. The simple RNN with a predict-next objective is allowed to forget information once it has become irrelevant for predicting the next elements in the sequence. To avoid this, we need information about all inputs to flow along the entire length of hidden states to enable the final hidden state to be a representation of the entire sequence.

The BRNN, introduced in Section 2.4.4, increases the representational power of the RNN architecture by allowing it to model all prefixes and suffixes independently. We

represent the entire sequence by concatenating the final hidden state in each direction. We propose to pair the BRNN with a novel training objective: at each position the model must predict the element at the corresponding position in the reverse of the input sequence. For example, if the input were *stonehenge* the network would need to predict the sequence *egnehenots*.

A BRNN trained with the proposed objective can only successfully predict a target sequence if the hidden states retain information about the ends of the input sequence to the opposite ends of the recurrence. For example, the final prediction can only be correct if information about the first input is communicated along the entire length of the recurrence. This objective ensures the final hidden states in each direction, which comprise our fixed size representation, contain long distance information not expected to be found in the simple RNN's representation.

Figure 5.2 depicts an example of learning to reverse a sequence of length 5 and shows how the fixed size representation $\mathbf{r}$ is constructed. This model is generalized in equations 5.7 through 5.11.

$$\mathbf{hf}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^{hf}\left[\mathbf{hf}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}^{hf}\right) & \text{otherwise} \end{cases} \tag{5.7}$$

$$\mathbf{hb}_t = \begin{cases} \mathbf{0} & \text{if } t = L+1 \\ f\left(\mathbf{W}^{hb}\left[\mathbf{hb}_{t+1}; \mathbf{x}_t\right] + \mathbf{b}^{hb}\right) & \text{otherwise} \end{cases} \tag{5.8}$$

$$\mathbf{y}_t = \mathbf{W}^{y}\left[\mathbf{hf}_t; \mathbf{hb}_t\right] + \mathbf{b}^{y} \tag{5.9}$$

$$\mathbf{r} = \left[\mathbf{hf}_L; \mathbf{hb}_1\right] \tag{5.10}$$

$$\mathcal{L}\left(\mathbf{x}_{1:L}\right) = \sum_{t=1}^{L} D^{sq\_euc}\left(\mathbf{y}_t, \mathbf{x}_{L-t+1}\right) \tag{5.11}$$

Details regarding the activation function f, output layer activation function, distance function D^{sq_euc}, and option to jointly learn frame representations match those for the simple RNN (Section 5.1). GRU or LSTM layers could replace the simple RNN layers used in Equations 5.7 and 5.8, as described in Section 2.4.5. Unlike the simple RNN, we need not include a pseudo-element to indicate the end-(or start-)of-sequence position(s) because the loss function is applied to the entire sequence length.

We may criticise the BRNN with sequence reversal objective in much the same way as the simple RNN with predict-next objective: the hidden states at the ends of the sequence are encouraged to contain information about the opposite ends of the sequence but need not contain information about the internal portion of the sequence. With the sequence-reversal objective, information about the middle of the sequence has become

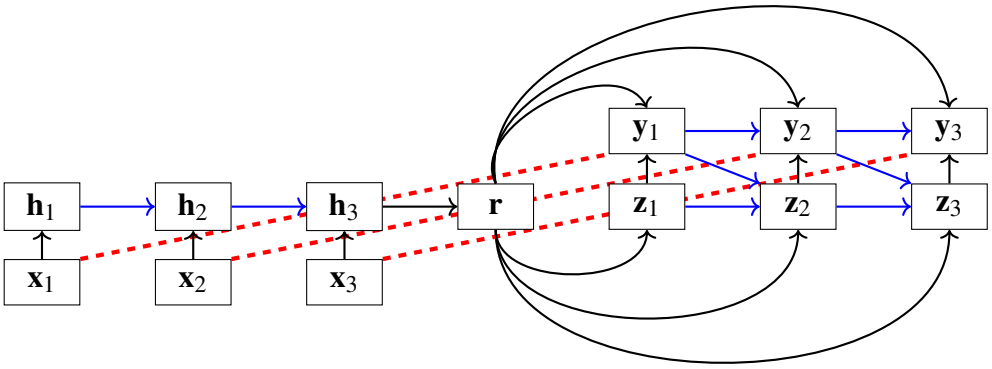


Figure 5.3: **RNN encoder-decoder**. An example of using an RNN encoder-decoder as a sequence autoencoder. A RNN, without any output at each position, is used to encode the input sequence $\mathbf{x}_{1:3}$ into the sequence representation of interest $\mathbf{r}$. From $\mathbf{r}$, the sequence is decoded using a second RNN into reconstruction $\mathbf{y}_{1:3}$. Black lines indicate normal feedforward connections, blue lines indicate recurrent connections (depicted unwound), and dashed red lines indicate differences we minimize during training.

redundant before the ends of the sequence are reached and so may be forgotten. An extreme case of this problem is illustrated in Figure 5.2 where the middle element of an odd-length sequence need not be retained in the hidden states at all. The speech sequences we intend to use with this model tend to contain uninteresting data (e.g. silence) or confusing data (e.g. due to poor segmentation) at the ends so this model may not effectively represent the sequence portion we care most about – the middle.

5.3 RNN encoder-decoder

The RNN with a predict-next training objective (Section 5.1) does not explicitly encourage the hidden state to represent the whole sequence and this is only partially remedied by the BRNN with a sequence reversal objective (Section 5.2). These problems can be addressed entirely with the use of the RNN Encoder-Decoder (RNN-ED) [17]. This model learns to compress a variable length sequence into a fixed size vector and, jointly, to decompress that vector into a target sequence.

In its most general form, the RNN-ED learns to transform from one sequence domain into a different sequence domain. Figure 5.3 depicts an example of the RNN-ED

transforming the input sequence $\mathbf{x}_{1:3}$ into output sequence $\mathbf{y}_{1:3}$ via fixed size representation $\mathbf{r}$; in general, the input and output sequences need not have the same length. The RNN-ED has been used by the machine translation community [17] where competitive translation quality has been achieved without the need for extensive feature engineering.

We can use the RNN-ED for unsupervised representation for ASR,

1. as a sequence autoencoder (unsupervised) where the output is a reconstruction of the original sequence, or

2. as a sequence translator (supervised) where the output is compared to a target, e.g. corresponding, sequence.

In both cases, but unlike machine translation, we can force the network to emit exactly the right number of elements and thus avoid the need to learn to identify the end of sequence position. The second approach requires paired data which may be sourced from an unsupervised term discovery system. As with the autoencoder approaches of Chapter 3, we discard the decoder portion of the network after training as we only care about the sequence representation $\mathbf{r}$.

The network architecture is defined by Equations 5.12 through 5.15.

$$\mathbf{h}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^h \left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}^h\right) & \text{otherwise} \end{cases} \tag{5.12}$$

$$\mathbf{r} = f\left(\mathbf{W}^r \mathbf{h}_L + \mathbf{b}^r\right) \tag{5.13}$$

$$\mathbf{z}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^z \left[\mathbf{z}_{t-1}; \mathbf{y}_{t-1}; \mathbf{r}\right] + \mathbf{b}^z\right) & \text{otherwise} \end{cases} \tag{5.14}$$

$$\mathbf{y}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ \mathbf{W}^y \left[\mathbf{z}_t; \mathbf{y}_{t-1}; \mathbf{r}\right] + \mathbf{b}^y & \text{otherwise} \end{cases} \tag{5.15}$$

When using this network as a sequence autoencoder we use the loss

$$\mathcal{L}\left(\mathbf{x}_{1:L}\right) = \sum_{t=1}^{L} D^{sq\_euc}\left(\mathbf{y}_t, \mathbf{x}_t\right) \tag{5.16}$$

and when using the network as a sequence translator, where $\mathbf{x}^2_{1:L^2}$ is the target sequence, we use the loss

$$\mathcal{L}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}\right) = \sum_{t=1}^{L^2} D^{sq\_euc}\left(\mathbf{y}_t, \mathbf{x}^2_t\right) \tag{5.17}$$

Details regarding the activation function f, output layer activation function, distance function D^{sq_euc}, and option to jointly learn frame representations match those for the

simple RNN (Section 5.1). GRU or LSTM layers could replace the simple RNN layers used in Equations 5.12 and 5.14, as described in Section 2.4.5. Indeed, Cho et al. [17] found that simple RNN layers were inadequate for their task.

Using the RNN-ED as a sequence autoencoder explicitly generates a representation of an entire input sequence – the model is only able to reconstruct the input sequence if the representation $\mathbf{r}$ contains a good summary. There is one caveat to this claim: the RNN-ED's decoder is powerful and can learn a great deal of information about regularities in the data in its own right. In principle, the decoder could learn a range of standard templates/patterns (local or global) and then the representation need only specify a standard template or pattern to follow, plus any deviations from that pattern. In a sense, the representation could be interpreted as a set of abstract instructions for the decoder. This cannot reduce the representational power of the fixed size representations but has the potential to impair the discriminability properties of the representations.

Sutskever et al. [88] found that training a similar model to emit target sequences in reverse order yielded improvements. This makes intuitive sense because it encourages the network to learn a decoder that is a more explicit inversion of the encoder; indeed, there may be scope to tie some weights. Targeting the reverse sequence may be especially useful for our application of sequence autoencoding, where we expressly want the decoder to be an inverted version of the encoder.

Unlike the simple RNN with a predict-next objective, we have an opportunity to make the network bidirectional. This could only apply to the encoder portion of the network since the decoder has no sequence input. A bidirectional encoder is specified in Equations 5.18 through 5.20.

$$\mathbf{hf}_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^{hf}\left[\mathbf{hf}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}^{hf}\right) & \text{otherwise} \end{cases} \tag{5.18}$$

$$\mathbf{hb}_t = \begin{cases} \mathbf{0} & \text{if } t = L+1 \\ f\left(\mathbf{W}^{hb}\left[\mathbf{hb}_{t+1}; \mathbf{x}_t\right] + \mathbf{b}^{hb}\right) & \text{otherwise} \end{cases} \tag{5.19}$$

$$\mathbf{r} = f\left(\mathbf{W}^{r}\left[\mathbf{hf}_L; \mathbf{hb}_1\right] + \mathbf{b}^{r}\right) \tag{5.20}$$

Although the RNN-ED does yield representations that must treat all parts of an input sequence equally (unlike the models above), there is no explicit need for the representations to be easily discriminable. Using an autoencoder-style objective will accentuate this problem because there is no requirement for the representations to be discriminable. We hypothesize that using corresponding sequence pairs, and the RNN-

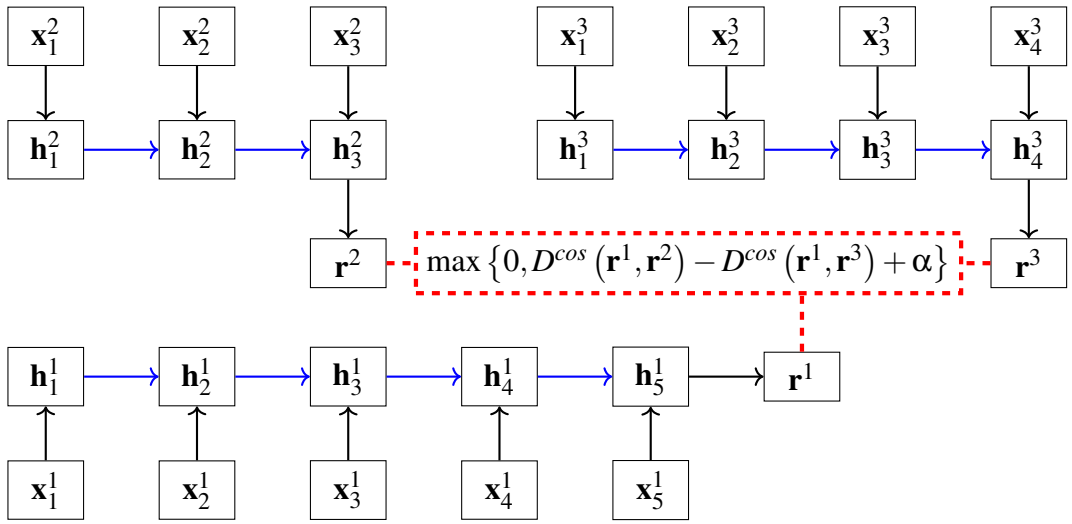


Figure 5.4: **Siamese RNN**. An example of applying the Siamese network approach to sequence modelling using a triplet ranking loss. The model parameters are trained to make the distance between same-class sequence representations $\mathbf{r}^1$ and $\mathbf{r}^2$ be at least α smaller than the distance between different-class sequence representations $\mathbf{r}^1$ and $\mathbf{r}^3$. Black lines indicate normal feedforward connections, blue lines indicate recurrent connections (depicted unwound), and dashed red lines indicate elements of the loss we minimize during training.

ED as a sequence translator, will implicitly improve representation discriminability in the same way that corresponding frame pairs improve the quality of a machine learned frame representation (Chapter 3): representations must congregate near class centroids.

5.4 Siamese RNN

We hypothesise the RNN-ED used as a sequence translator with corresponding sequence pairs will yield the best representations of all models presented in this chapter thus far, for the reasons outlined above. However, the sequence translator RNN-ED does not *explicitly* optimize the objective of real interest – learning representations that discriminate between classes. We care about class discrimination more than whole sequence representation; there is no need to represent entire sequences if just one portion is needed to discriminate the classes.

A solution to this problem is offered by Siamese neural networks (SNNs, [13]) where the training objective involves explicit comparisons between representations and

ensuring those comparisons exhibit the desired behaviours. We minimize the distance between representations of different utterances of the same speech unit and maximize the distance between representations of different utterances. By this method we explicitly match the model training objective to the evaluation metric and downstream task.

Good results have been obtained for frame-based representation learning using SNNs, e.g. Synnaeve et al. [90] and Thiolliere et al. [91]. Recent work by Kamper et al. [50] has achieved good results using Siamese convolutional neural networks over entire sequences. Here we present a novel variant on the theme – a Siamese RNN (SRNN) architecture.

The fixed size representation of sequence $\mathbf{x}^i_{1:L^i}$ is found via the architecture defined in Equations 5.21 and 5.22.

$$\mathbf{h}^i_t = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^h \left[\mathbf{h}^i_{t-1}; \mathbf{x}^i_t\right] + \mathbf{b}^h\right) & \text{otherwise} \end{cases} \tag{5.21}$$

$$\mathbf{r}^i = f\left(\mathbf{W}^r \mathbf{h}^i_{L^i} + \mathbf{b}^r\right) \tag{5.22}$$

Details regarding the activation function f and option to jointly learn frame representations match those for the simple RNN (Section 5.1). GRU or LSTM layers could replace the simple RNN layers used in Equation 5.21, as described in Section 2.4.5.

We propose using a triplet ranking loss to explicitly encode our desired representation behaviour into the training objective. We want representations of same-class sequences to be more like each other than representations of different-class sequences. Section 4.2 provides the detail and reasoning behind triplet ranking losses. Here we apply a triplet loss by comparing the distance between fixed-size representations of whole sequences instead of dynamic time warping alignment costs. Equation 5.23 is equivalent to Equation 4.2 but using the notation of this section. Figure 5.4 depicts a SRNN with this triplet loss.

$$\mathcal{L}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}, \mathbf{x}^3_{1:L^3}\right) = \max\left\{0, D^{cos}\left(\mathbf{r}^1, \mathbf{r}^2\right) - D^{cos}\left(\mathbf{r}^1, \mathbf{r}^3\right) + \alpha\right\} \tag{5.23}$$

Synnaeve et al. [90] evaluated a variety of distance functions for training frame-based Siamese networks and found the squared cosine between negative pairs worked best for their architecture, yielding a distance function called $coscos^2$. We adapt $coscos^2$ into an alternate triplet ranking loss for sequences in Equation 5.24.

$$\mathcal{L}^{alt}\left(\mathbf{x}^1_{1:L^1}, \mathbf{x}^2_{1:L^2}, \mathbf{x}^3_{1:L^3}\right) = \max\left\{0, D^{cos}\left(\mathbf{r}^1, \mathbf{r}^2\right) - D^{cos^2}\left(\mathbf{r}^1, \mathbf{r}^3\right) + \alpha\right\} \tag{5.24}$$

where

$$D^{cos^2}\left(\mathbf{r}^1,\mathbf{r}^3\right) = cos^2\left(\mathbf{r}^1,\mathbf{r}^3\right) = \left(\frac{\mathbf{r}^1\mathbf{r}^3}{\|\mathbf{r}^1\|_2\,\|\mathbf{r}^3\|_2}\right)^2 \in [0,1] \tag{5.25}$$

The SRNN can be made bidirectional by replacing Equations 5.21 and 5.22 with Equations 5.26 through 5.28.

$$\mathbf{hf}_t^i = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ f\left(\mathbf{W}^{hf}\left[\mathbf{hf}_{t-1}^i;\mathbf{x}_t^i\right] + \mathbf{b}^{hf}\right) & \text{otherwise} \end{cases} \tag{5.26}$$

$$\mathbf{hb}_t^i = \begin{cases} \mathbf{0} & \text{if } t = L+1 \\ f\left(\mathbf{W}^{hb}\left[\mathbf{hb}_{t+1}^i;\mathbf{x}_t^i\right] + \mathbf{b}^{hb}\right) & \text{otherwise} \end{cases} \tag{5.27}$$

$$\mathbf{r}^i = f\left(\mathbf{W}^r\left[\mathbf{hf}_{L^s}^i;\mathbf{hb}_1^i\right] + \mathbf{b}^r\right) \tag{5.28}$$

Section 4.1 details the methods by which a paired-sequence training dataset can be sourced in an unsupervised fashion.

The SRNN instils the behaviours we desire of our representations into the training objective but it has lost the notion of explicit sequence representation. The triplet ranking loss ensures that different speech unit classes are distinct in the representation space but there is no requirement for representations to summarise the frame sequences from which they are computed. While we care most about class discrimination we may still get some value from sequence summarization, particularly in combination with sequence regularization methods such as those described in Section 5.5. As a result we may consider combining the SRNN with the RNN-ED: the training objective would mix both triplet ranking loss and reconstruction loss via a mixing weight hyperparameter λ, e.g. Equation 5.29.

$$\begin{aligned} \mathcal{L}\left(\mathbf{x}_{1:L^1}^1,\mathbf{x}_{1:L^2}^2,\mathbf{x}_{1:L^3}^3\right) &= \max\left\{0, D^{cos}\left(\mathbf{r}^1,\mathbf{r}^2\right) - D^{cos}\left(\mathbf{r}^1,\mathbf{r}^3\right) + \alpha\right\} \\ &\quad + \sum_{i=1}^{3}\frac{\lambda}{L^i}\sum_{t=1}^{L^i} D^{sq\_euc}\left(\mathbf{y}_t^i,\mathbf{x}_t^i\right) \end{aligned} \tag{5.29}$$

5.5 Regularization of sequence models

Standard techniques for neural network regularization can be applied to any of the sequence models presented in this chapter, including applying an L1 or L2 norm to the model parameters [63], using dropout [87], or artificially expanding the training data. In this section we explore (1) a variant of dropout (denoising), (2) application of a weak supervision signal (sequence correspondence), and (3) applying the idea of contractive autoencoders to sequence models.

Denoising autoencoders regularize their training by artificially corrupting the inputs. A standard version of this is random zero-masking – in each epoch we apply a random zero-mask to each training datum so only some proportion of the data is actually visible to the network at any one time. This has the effect of expanding the size of the training data by making it look like we have many more instances, and those instances are also more noisy. The extra noise helps prevent the network from overfitting the data.

Adding noise to a sequence can be more involved than adding noise to non-sequential inputs because in addition to modifying the frames, we may also wish to adjust the length of the sequence.

5.5.1 Artificial noise methods that do not change the sequence length

- Zero-masking frame components: Each frame is independently corrupted by element-wise multiplying it by a binary mask sampled from a multivariate Bernoulli distribution; $\hat{\mathbf{x}}_t = \mathbf{x}_t \odot (\mathbf{1} - \mathbf{m}_t)$ where $\mathbf{m}_t \sim Bernoulli(p)$ and p is the probability of a dimension being zeroed out.

- Adding Gaussian noise to frame components: Our speech input representations are normalized to be approximately multivariate Gaussian distributed with zero mean and identity variance so additive zero-mean Gaussian noise is appropriate; $\hat{\mathbf{x}}_t = \frac{1}{2}(\mathbf{x}_t + \mathbf{n}_t)$ where $\mathbf{n}_t \sim Gaussian\left(\mathbf{0}, \mathbf{I}\sigma^2\right)$ and σ^2 is the variance of the noise applied to each dimension.

- Zero-masking entire frames: As for frame components, but applied to entire frames instead; $\hat{\mathbf{x}}_t = m_t \mathbf{x}_t + (1 - m_t)\mathbf{x}_t$ where $m_t \sim Bernoulli(p)$ and p is the probability of an entire frame being zeroed out.

Given the usual overlap in duration of speech frames (sampled every 10 ms but each covers a period of 25 ms, thus overlapping the next frame fully and the following frame partially) the above noise application methods may need to be applied consistently to non-overlapping sub-sequences of length 2 or 3 frames.

Each of the three techniques above could be used to artificially expand the size of the training data, or used in the context of denoising training. For the former, the four sequence based models described in Sections 5.1 through 5.4 would be unaware of the original uncorrupted inputs, they are simply trained with more inputs, some of which are corruptions of the originals. For denoising, the input and target sequences differ

only in that the input sequence is the noisy version of the target sequence; all four of the sequence models in this chapter can be used this way.

5.5.2 Artificial noise methods that can change the sequence length

- Up- or down-sample the speech representation sequence: MFCCs are conventionally sampled at a rate of 100Hz but we may produce (somewhat) semantically identical sequences by sampling at different rates. Computing sequences of MFCCs for the same speech signal at different rates (e.g. 100 Hz and 200 Hz) yields different sequences which we want to have identical representations.

- Use over- or under-segmented sequences: We want to use these representations within systems that attempt to segment a speech signal into meaningful units. The segmentation decisions will be error-prone but we would still like the representations of segments to accurately identify their principal contents. Our networks can be trained with sequences that are deliberately over- or under-segmented such that the segment contains some of the preceding or following segment, or does not contain the complete intended segment. This method only applies when training with meaningful, i.e. non-random, speech segments.

- Drop or insert random frames: Sequences can be made explicitly shorter or longer by dropping or inserting random frames; if done carefully, this can help the network generalize to sequences of varying length. When dropping frames the neighbouring frames can be adjusted to include some of the dropped frame's information, i.e. if the frame at position t is dropped then $\hat{\mathbf{x}}_{t-1} = \lambda \mathbf{x}_{t-1} + (1-\lambda)\,\mathbf{x}_t$ and $\hat{\mathbf{x}}_{t+1} = \lambda \mathbf{x}_{t+1} + (1-\lambda)\,\mathbf{x}_t$. When inserting frames, the inserted frame can be a noisy mixture of the neighbouring frames, i.e. $\hat{\mathbf{x}}_t = \frac{\lambda}{2}\,(\mathbf{x}_{t-1} + \mathbf{x}_{t+1}) + (1-\lambda)\,\mathbf{n}_t$. In both cases $\lambda \in [0,1]$ is a hyperparameter that determines how much mixing occurs.

5.5.3 Using a correspondence supervision signal

As with frame-based models, we may apply the idea of correspondence training to sequence-based models. Given a supervision signal that identifies pairs of same-class sequences (e.g. unsupervised term discovery), we can use one sequence in a pair as the input sequence and the other sequence in the pair as the target sequence. For the simple RNN with predict-next objective, and the BRNN with sequence reversal objective, the

input and target sequences need to be frame-aligned, as for the frame-based models, because the input and target sequences must have the same length. For the RNN-ED and SRNN the input and target sequences need not be frame aligned as these two models support input and target sequences of different lengths.

We hypothesize that a correspondence training signal can improve the quality of sequence representations in the same way it improves the quality of frame representations – it is a more powerful form of regularization: we use natural variance in the data instead of adding artificial noise.

5.5.4 Contractive objective

A final form of regularization that could have particular application to sequence representation learning comes from contractive autoencoders [76]. The basic idea is to add a term to the training objective so as to minimize the gradient of the representation with respect to the input. Achieving this objective ensures a small change in the input yields only a small change in the representation – a behaviour we would like our representations to exhibit. Given a loss $\mathcal{L}$, we add the contractive objective to form the composite loss given in Equation 5.30 where γ is a hyperparameter determining the relative strength of the contractive objective compared to the model's standard training objective.

$$\mathcal{L}^{cont}(\mathbf{x}_{1:L}) = \mathcal{L}(\mathbf{x}_{1:L}) + \gamma \|J_{\mathbf{r}}(\mathbf{x}_{1:L})\|_F^2 \tag{5.30}$$

The contractive objective uses the squared Frobius norm of the Jacobian of the representation function, given in Equation 5.31.

$$\|J_{\mathbf{r}}(\mathbf{x}_{1:L})\|_F^2 = \sum_{t=1}^{L} \sum_{i=1}^{|\mathbf{x}_t|} \sum_{j=1}^{|\mathbf{r}|} \left(\frac{\partial r_j}{\partial x_{t,i}} \right)^2 \tag{5.31}$$

For sequence inputs the Jacobian becomes large, with LD^xD^r entries, and each gradient involves $O(L)$ factors. Trial experiments using the contractive objective in a sequence model suggest it may not be practical – computing the Frobius norm of the Jacobian takes too much time with real sized speech data – unless a more efficient implementation can be found.

5.6 Summary

Many possibilities exist for finding fixed size representations of variable length speech frame sequences. No experimental results have been presented here so no conclusions

can be drawn about the degree to which these methods work as intended. We have instead presented justified hypotheses about the strengths and weaknesses of each proposed method. Further work is needed to verify or refute the assumptions and assertions presented here.

We hypothesize that a simple recurrent neural network (RNN) with a predict-next style objective will fail to yield useful fixed size representations because the objective does not encourage the model to retain pertinent information within its hidden state, which we ultimately use as the representation, along the entire length of the sequence. We hypothesize that the bidirectional RNN with a sequence reversal objective only partially solves this problem because this alternate objective only encourages the model to retain information about the sequence ends (about which we tend to care little) and, potentially, discards information about the sequence interior (about which we care most).

The RNN Encoder Decoder is structured such that the fixed size representation must contain a summary of the entire sequence but we hypothesize that this summary will not be as distinctive as it could be because the training objective does not compare representations in any way. We hypothesize that the Siamese RNN solves this final problem by using a training objective that directly matches the way in which the model is evaluated and used in downstream tasks – maximizing representation similarity when sequences are drawn from the same class and minimizing representation similarity when they are drawn from different classes.

We hypothesize that the sequence models presented here will work at their best when paired with a supervision signal providing correspondence information. This signal may come from an unsupervised source, e.g. an unsupervised term discovery system, which would make the system as a whole unsupervised, the scenario of most interest to us. When a correspondence supervision signal is unavailable, or undesirable, the models presented in this chapter can be trained with alternate methods of regularization, and we have presented a variety of methods here. We hypothesize that artificially adding noise to the training data will not yield models that perform as well as a correspondence-based model, but will still outperform models that are not regularized at all.

Chapter 6

Conclusions and future work

Unsupervised automatic speech recognition (ASR) is a difficult task: there is a 25 to 30 year gap between comparable state-of-the-art supervised and unsupervised word error rates. Representation learning is a critical component of unsupervised speech processing which is typically built on similarity-based methods and it is the representations of the sequences being compared that determine their similarity. Without labelled data we must use unsupervised representation learning methods, such as autoencoders, or combine supervised representation learning with a training signal from an unsupervised source, e.g. unsupervised term discovery (UTD). This thesis has contributed four methods that are either shown, or hypothesised, to improve the quality of representations or the efficiency of their use for unsuperivsed ASR.

In Chapter 3 we showed that an improved version of the correspondence autoencoder (cAE) can learn better quality frame representations than some baseline methods and the previous cAE version. The general approach used in our entry to the INTERSPEECH 2015 Zero Resource Speech Challenge suggests that the combination of powerful *supervised* representation learning with a noisy training signal from an unsupervised source (e.g. UTD) can be more powerful than standard unsupervised representation learning methods alone.

Our evaluation methods, and downstream tasks, use DTW to compare speech signal representations but DTW has not previously played an explicit role within the optimization of the representations. In Section 4.2 we proposed a model that explicitly optimizes the representations for DTW. Training with a triplet ranking loss, the representations are optimized to make DTW distances between same-class sequence pairs smaller than the distances between different-class sequence pairs.

Optimal sequence similarity algorithms such as dynamic time warping (DTW) can

limit the scalability of unsupervised speech processing methods because they have a time complexity that is quadratic in the lengths of the sequences being compared. In Section 4.3 we proposed a model that learns to approximate the DTW alignment algorithm using only local information. The trained model can approximately align two sequences with a time complexity that is linear in the sequence lengths. We went further in Chapter 5 and proposed a series of models that learn fixed size representations of variable length speech signals enabling even faster comparison of sequences using a single vector space distance computation.

The most urgent need for future work is clearly experimentation of the models proposed in Chapters 4 and 5. At present we cannot be certain that the hypotheses expounded in those chapters would be supported or weakened by empirical results. Beyond that we have considered:

- **Combining model components and training objectives**. Good representations of individual frames (Chapter 3) may improve the representations of whole sequences (Chapter 5). One could construct a model that incorporates both frame-level and sequence-level training objectives. For example, a loss that combines frame-reconstruction with sequence-reconstruction, or a loss that combines Siamese frame comparisons and Siamese sequence comparisons. This could be potentially developed further by also incorporating an explicit DTW optimization component (Section 4.2).

- **Convolution vs. recursion vs. recurrence**. Speech signals include both short and long temporal patterns. Convolutional neural networks (CNNs) have proved powerful solutions for both supervised ASR and unsupervised representation learning [50]. CNNs can only detect short temporal patterns but RNNs in comparison can capture long temporal patterns. Combining a CNN with a RNN has proved successful in language modelling [51] and may be valuable here for the same reasons: we can capture patterns at a wider variety of time scales.

- **Representation-segmentation-clustering-labelling**. Our work has largely focussed on representation learning as a stand-alone activity, independent of representation use. But representation learning is only one subtask necessary for unsupervised ASR. Considering the larger task of unsupervised automatic speech recognition, we may find it important to develop models that learn to jointly segment, represent, cluster, and label the meaningful units in the speech signal.

This might be achieved by coarsely combining multiple models together and iteratively applying gradual optimization until convergence. Alternatively it might be achieved by developing an integrated model, perhaps a neural network, that performs all of these tasks in a "soft", i.e. differentiable, manner combined with a decoding method, e.g. beam search, to predict the "hard" output sequence. Some recent work is moving in this direction, for example Kamper et al [49] jointly segment and cluster while Kong et al. [52] jointly represent and segment.

We are still a long way from creating an ASR system capable of contributing significantly to an alien language understanding scenario like that portrayed in Chapter 1 but maybe in 150 years that scenario will not seem so far fetched. Current trends suggest unsupervised ASR technology will have advanced to a point where it could be used productively to understand our interstellar neighbours.

Bibliography

[1] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 3366–3370, 2013. 15

[2] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015. 49

[3] Leonardo Badino, Claudia Canevari, Luciano Fadiga, and Giorgio Metta. An auto-encoder based approach to unsupervised learning of subword units. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 7634–7638, 2014. 39

[4] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: New features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 38, 48

[5] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 12

[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. 14

[7] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. 21

[8] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation. 38, 48

[9] Mayank Bhargava and Richard Rose. Architectures for deep neural network based acoustic models defined over windowed speech waveforms. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 6–10, 2015. 13

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. 8

[11] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991. 38

[12] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988. 17, 20, 32

[13] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993. 68

[14] Marcia A. Bush and Gary E. Kopec. Network-based connected digit recognition. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 35(10):1401–1413, 1987. 9

[15] Michael A. Carlin, Samuel Thomas, Aren Jansen, and Hynek Hermansky. Rapid evaluation of speech representations for spoken term discovery. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 821–824, 2011. 14, 18, 32

[16] Guoguo Chen, Carolina Parada, and Tara N. Sainath. Query-by-example keyword spotting using long short-term memory networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5236–5240, 2015. 9

[17] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014. 24, 28, 60, 65, 66, 67

[18] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. 28

[19] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004. 8

[20] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980. 13, 32

[21] Nic J. de Vries, Marelie H. Davel, Jaco Badenhorst, Willem D. Basson, Febe de Wet, Etienne Barnard, and Alta de Waal. A smartphone-based ASR data collection tool for under-resourced languages. *Speech Communication*, 56:119–131, 2014. 32, 35

[22] Mark Dredze, Aren Jansen, Glen Coppersmith, and Ken Ward Church. NLP on spoken documents without ASR. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 460–470, 2010. 37

[23] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. 38

[24] Daniel PW Ellis and Manuel Reyes-Gomez. Investigations into tandem acoustic modeling for the Aurora task. In *Eurospeech 2001: Scandinavia: 7th European Conference on Speech Communication and Technology: September 3-7, 2001, Aalborg Congress and Culture Centre, Aalborg-Denmark: proceedings*, pages 189–192. ISCA-Secretariat, 2001. 15

[25] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. 22

[26] Gunnar Evermann, Ho Yin Chan, Mark J. F. Gales, Bin Jia, David Mrva, Philip C. Woodland, and Kai Yu. Training LVCSR systems on thousands of hours of data. In *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '05, Philadelphia, Pennsylvania, USA, March 18-23, 2005*, pages 209–212, 2005. 44

[27] Michael Franzini, Kai-Fu Lee, and Alex Waibel. Connectionist Viterbi training: A new hybrid method for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 425–428. IEEE, 1990. 15

[28] Alvin Garcia and Herbert Gish. Keyword spotting of arbitrary words using minimal speech resources. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19, 2006*, pages 949–952, 2006. 8

[29] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000. 26, 27

[30] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002. 26, 27

[31] Herbert Gish, Man-hung Siu, Arthur Chan, and Bill Belfield. Unsupervised training of an HMM-based speech recognizer for topic classification. In *Tenth Annual Conference of the International Speech Communication Association*, 2009. 9

[32] James R. Glass. Towards unsupervised speech processing. In *11th International Conference on Information Science, Signal Processing and their Applications, ISSPA 2012, Montreal, QC, Canada, July 2-5, 2012*, pages 1–4, 2012. 7, 8

[33] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Convolutional neural networks for acoustic modeling of raw time signal in LVCSR. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 26–30, 2015. 13

[34] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky. Probabilistic and bottle-neck features for LVCSR of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–757–IV–760, April 2007. 33

[35] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 1735–1742, 2006. 47

[36] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. 23

[37] Hynek Hermansky, Daniel W Ellis, and Shantanu Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE, 2000. 15

[38] G. E. Hinton. Boltzmann machine. *Scholarpedia*, 2(5):1668, 2007. revision number 91075. 16

[39] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. 16

[40] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 21

[41] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 3–10, 1993. 16

[42] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 1991. 21, 24

[43] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Networks*, pages 237–243. Wiley-IEEE Press, 2001. 24

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 26

[45] Navdeep Jaitly and Geoffrey E. Hinton. Learning a better representation of speech soundwaves using restricted Boltzmann machines. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5884–5887, 2011. 16

[46] Aren Jansen, Kenneth Church, and Hynek Hermansky. Towards spoken term discovery at scale with zero resources. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1676–1679, 2010. 8

[47] Aren Jansen and Benjamin Van Durme. Efficient spoken term discovery using randomized algorithms. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*, pages 401–406, 2011. 8, 32, 33, 37

[48] Herman Kamper, Micha Elsner, Aren Jansen, and Sharon Goldwater. Unsupervised neural network based feature extraction using weak top-down constraints. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5818–5822, 2015. 19, 32, 33, 34, 35, 38

[49] Herman Kamper, Aren Jansen, and Sharon Goldwater. Fully unsupervised small-vocabulary speech recognition using a segmental Bayesian model. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 678–682, 2015. 10, 49, 77

[50] Herman Kamper, Weiran Wang, and Karen Livescu. Deep convolutional acoustic word embeddings using word-pair side information. *CoRR*, abs/1510.01032, 2015. 69, 76

[51] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015. 76

[52] Lingpeng Kong, Chris Dyer, and Noah A. Smith. Segmental recurrent neural networks. *CoRR*, abs/1511.06018, 2015. 77

[53] Lori Lamel, Jean-Luc Gauvain, and Gilles Adda. Lightly supervised and unsupervised acoustic model training. *Computer Speech & Language*, 16(1):115–129, 2002. 8

[54] Jong-Hwan Lee, Ho-Young Jung, Te-Won Lee, and Soo-Young Lee. Speech feature extraction using independent component analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*, pages 1631–1634, 2000. 17

[55] Keith Levin, Katharine Henry, Aren Jansen, and Karen Livescu. Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 410–415, 2013. 10, 17, 18

[56] Jinyu Li, Dong Yu, Jui-Ting Huang, and Yu Gong. Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 131–136. IEEE, 2012. 14

[57] Liang Lu, Xingxing Zhang, Kyunghyun Cho, and Steve Renals. A study of the recurrent neural network encoder-decoder for large vocabulary speech recogni-

tion. In *Sixteenth Annual Conference of the International Speech Communication Association*, pages 3249–3253, 2015. 60

[58] Vince Lyzinski, Gregory Sell, and Aren Jansen. An evaluation of graph clustering methods for unsupervised term discovery. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3209–3213, 2015. 9

[59] Florian Metze, Xavier Anguera, Etienne Barnard, Marelie Davel, and Guillaume Gravier. The spoken web search task at MediaEval 2012. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8121–8125. IEEE, 2013. 9

[60] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010. 61

[61] Nelson Morgan and Herve Bourlard. An introduction to the hybrid HMM/connectionist approach. *IEEE Signal Processing Magazine*, pages 25–42, 1995. 15

[62] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010. 23

[63] Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM. 70

[64] Yves Normandin, Régis Cardin, and Renato de Mori. High-performance connected digit recognition using maximum mutual information estimation. *IEEE Transactions on Speech and Audio Processing*, 2(2):299–311, 1994. 9

[65] Scott Novotney, Richard M. Schwartz, and Jeff Z. Ma. Unsupervised acoustic and language model training with small amounts of labelled data. In *Proceedings of*

the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009, 19-24 April 2009, Taipei, Taiwan, pages 4297–4300, 2009. 8

[66] Dimitri Palaz and Mathew Magimai-Doss Ronan Collobert. Analysis of CNN-based speech recognition system using raw speech as input. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 11–15, 2015. 13

[67] A. S. Park and James R. Glass. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech & Language Processing*, 16(1):186–197, 2008. 8, 44

[68] Mark A. Pitt, Keith Johnson, Elizabeth Hume, Scott Kiesling, and William D. Raymond. The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability. *Speech Communication*, 45(1):89–95, 2005. 32, 35

[69] Louis C. W. Pols, Xue Wang, and Louis ten Bosch. Modelling of phone duration (using the TIMIT database) and its potential benefit for ASR. *Speech Communication*, 19(2):161–176, 1996. 24

[70] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB. 37

[71] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007. 8

[72] J. Ramirez, J. M. Gorriz, and J. C. Segura. Voice activity detection. fundamentals and speech recognition system robustness. In *Robust Speech Recognition and Understanding*, pages 1–22. I-Tech Education and Publishing, 2007. 44

[73] Shakti P. Rath, Kate M. Knill, Anton Ragni, and Mark J. F. Gales. Combining tandem and hybrid systems for improved speech recognition and keyword spotting on low resource languages. In *INTERSPEECH 2014, 15th Annual Conference*

of the International Speech Communication Association, Singapore, September 14-18, 2014, pages 835–839, 2014. 15

[74] Daniel Renshaw and Keith B. Hall. Long short-term memory language models with additive morphological features for automatic speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5246–5250, 2015. 26

[75] Daniel Renshaw, Herman Kamper, Aren Jansen, and Sharon Goldwater. A comparison of neural network methods for unsupervised representation learning on the Zero Resource Speech Challenge. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3199–3203, 2015. 31

[76] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 833–840, 2011. 73

[77] A. J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, Mar 1994. 22

[78] Tony Robinson and Frank Fallside. A recurrent error propagation network speech recognition system. *Computer Speech & Language*, 5(3):259–274, 1991. 22

[79] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. 8

[80] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. 24

[81] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform CLDNNs. In *INTERSPEECH*

2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015, pages 1–5, 2015. 13

[82] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 338–342, 2014. 26, 27

[83] Thomas Schatz, Vijayaditya Peddinti, Francis R. Bach, Aren Jansen, Hynek Hermansky, and Emmanuel Dupoux. Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 1781–1785, 2013. 14, 19, 39

[84] Thomas Schatz, Vijayaditya Peddinti, Xuan-Nga Cao, Francis R. Bach, Hynek Hermansky, and Emmanuel Dupoux. Evaluating speech features with the minimal-pair ABX task (II): resistance to noise. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 915–919, 2014. 14, 19, 39

[85] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 23

[86] Garimella S. V. S. Sivaram, Sridhar Krishna Nemala, Mounya Elhilali, Trac D. Tran, and Hynek Hermansky. Sparse coding for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010, 14-19 March 2010, Sheraton Dallas Hotel, Dallas, Texas, USA*, pages 4346–4349, 2010. 17

[87] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 70

[88] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014. 67

[89] Gabriel Synnaeve and Emmanuel Dupoux. Weakly supervised multi-embeddings learning of acoustic models. *CoRR*, abs/1412.6645, 2014. 32

[90] Gabriel Synnaeve, Thomas Schatz, and Emmanuel Dupoux. Phonetics embedding learning with side information. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 106–111, 2014. 32, 69

[91] Roland Thiolliere, Ewan Dunbar, Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3179–3183, 2015. 47, 49, 69

[92] Samuel Thomas, Michael L. Seltzer, Kenneth Church, and Hynek Hermansky. Deep neural network features and semi-supervised training for low resource speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6704–6708, 2013. 8

[93] Edmondo Trentin and Marco Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37(1):91–126, 2001. 15

[94] Maarten Versteegh, Roland Thiolliere, Thomas Schatz, Xuan-Nga Cao, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux. The Zero Resource Speech Challenge 2015. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3169–3173, 2015. 39

[95] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 1096–1103, 2008. 32

[96] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010. 21, 39

[97] Oliver Walter, Timo Korthals, Reinhold Haeb-Umbach, and Bhiksha Raj. A hierarchical system for word discovery exploiting DTW-based initialization. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 386–391, 2013. 9

[98] P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990. 24

[99] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK book (for HTK version 3.4)*. Cambridge University Engineering Department, 2006. 35

[100] Dong Yu and Li Deng. *Automatic Speech Recognition – A Deep Learning Approach*. Springer-Verlag London, first edition, 2015. 9

[101] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 215–219, 2014. 37

[102] Yaodong Zhang and James R Glass. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 398–403. IEEE, 2009. 9

[103] Yaodong Zhang and James R. Glass. An inner-product lower-bound estimate for dynamic time warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5660–5663, 2011. 8

[104] Yaodong Zhang, Ruslan Salakhutdinov, Hung-An Chang, and James Glass. Resource configurable spoken query detection using deep Boltzmann machines. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5161–5164. IEEE, 2012. 9

[105] Xin Zheng, Zhiyong Wu, Helen Meng, and Lianhong Cai. Contrastive auto-encoder for phoneme recognition. In *IEEE International Conference on Acoustics,*

Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014, pages 2529–2533, 2014. 32