Unsupervised Phonetic Speech Segmentation with Neural Networks

Zixin (Yasmin) Yang

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2022

Abstract

Phonemes are the basic units of speech, and speech processing tasks, either recognition or synthesis, are made possible under the idea that speech can be decomposed into phonetic segments. The development of systems which we use to automatically segment speech into phonetic units is thus crucial for improving the performances of speech technology. In this project, we look into the workings of the first representation learning based convolutional neural network model on the task of unsupervised phonetic speech segmentation, and carry out in-depth analysis on the distribution of segmentation errors over each generic phonetic class. The effectiveness of different aspects of the model architecture is examined through designed experimentation, from which the mechanisms behind this best-performing approach on this particular task are explored and justified, with suggestions on the future design choices for improving representation learning based unsupervised phonetic speech segmentation model.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Zixin (Yasmin) Yang*)

Acknowledgements

I would like to thank my project supervisor, Dr. Peter Bell, for introducing this topic to me, and for his invaluable advice and motivation, which has helped me in successfully completing this project.

I would also like to thank Zeyu Zhao, a tutor in automatic speech recognition, for his time and efforts on helping me solve technical problems encountered at the experimentation stage of the project.

Thank you to all who had supported and encouraged me for my work on this project.

Table of Contents

Chapter 1

Introduction

1.1 Motivations

Speech is the single most important means of communication, that is the most natural and effective for humans to express opinions, deliver information, and convey emotions [9]. Speech processing has been an active field of research for centuries [19]. The idea that spoken speech can be decomposed, or segmented, into smaller units is crucial for both speech recognition and speech synthesis, which are the two fundamental areas in speech processing — the former is the task of transcribing speech signals into the textual form of word sequences, and the latter task is the reverse of this procedure [20]. As these tasks are made feasible on the fact that speech can be segmented into word, syllabic, or phonetic units, the development of speech segmentation systems is essential to improving the technology used for speech recognition and synthesis.

The main focus of this project is on the segmentation of speech at the phonetic level. Although there are in general more works on segmenting speech at the word level as word sequences are end results of most automatic speech recognition (ASR) systems, phonetic segments are as significant as their word counterparts given their various applications. Examples include the generation of segment libraries used in speech synthesis [38], the computation of speech rate, the production of annotations for phonetic analysis [25], and the masking strategy based on phone units in ASR systems [21].

For decades, phonetic segmentation has been studied as an automated task for various reasons. Firstly, manual annotation is tedious and expensive, particularly given the increasing sizes of corpora nowadays; however, the annotation process requires substantial work by skilled experts. Secondly, reliable automated systems may prevent human mistakes and the potential differences in annotation due to subjectivity [24], and allow reproduction of results when necessary, while it takes only a fraction of the original time [38]. Therefore, improving the performance of an automated phonetic segmentation model is extremely desirable.

This task is further automated through the application of unsupervised methods, also known as “Blind” segmentation, which is the location of boundaries of discrete speech units when no transcription is available in advance [32]. With recent advances in computing and machine learning, models for phonetic speech segmentation have also started to employ neural networks [25, 40, 22], which are trained on large amounts of data to achieve adequate performance. Unlike supervised learning that requires transcribed data, unsupervised learning allows the network model to be trained on data without any labeling. The fact that unsupervised methods do not need manual transcriptions on speech is especially useful for under-resourced, or endangered, languages, and languages without consistent orthographies (English, for example, has conflicting rules for pronunciation that don't apply in general) [25]. Besides, the amount of data significantly increases for training deeper neural networks, which are starting to become more commonly used today in most prediction and classification tasks. These are what make unsupervised learning more important, in addition to reasons mentioned prior.

Yet, previous work on the unsupervised neural network approach to blind segmentation is very limited. Recent success in machine learning methods on speech representation learning [33, 3, 4] has demonstrated the power of learned representations on various downstream tasks, such as phoneme recognition (PR), ASR and speaker diarization (SD) [42]. The work by Kreuk *et al.* on phonetic segmentation in 2020 ([22]) was the first that proposed a convolutional neural network (CNN) model on the basis of self-supervised representation learning, and has outperformed all previous unsupervised methods on blind segmentation, and is still the cornerstone of the current state-of-the-art model in 2021 [5]. Due to the remarkable improvement in performance brought by this model [22], it is meaningful to investigate its strengths and weaknesses, and to explore the effectiveness of different components of its architecture, which is the main goal of this project.

1.2 Goal and Research Question

The goal of this project is to investigate aspects of the model architecture in Kreuk's work ([22]) and their effectiveness on the phonetic speech segmentation task. The model's application of a CNN trained via Contrastive Predictive Coding is a novel approach to this task compared to the various methods proposed previously. Though it is significantly different from previous unsupervised models, it has outperformed all of them on this task, which makes it meaningful and important to examine closely the particular architecture used in this model. Therefore, the primary research question is:

> Why does the particular representation learning approach using a Convolutional Neural Network work best among unsupervised methods for the task of automatic phonetic speech segmentation?

We would further break this down into minor objectives in order to explore the different aspects and design decisions of this model.

1.2.1 Minor Objectives and Rationales

- Reproduce results and analyse the predicted segments of the original model;

    To make meaningful investigations on the model later on, it's important to gain better understanding on the strengths and weaknesses of the baseline model in terms of segmentation performance on different phonetic classes.

- Investigate intermediate representations by visualising feature maps from each layer of the CNN;

    It is hard to understand the workings in the hidden layers of a CNN, and visualization is a straightforward way to show which features are learned at each convolution block; and by also evaluating segmentation performance on these intermediate representations, how effectively the neural network learns can be illustrated.

- Investigate whether longer contextual information is beneficial to the segmentation results or not, by adding a Gated Recurrent Unit layer to the network;

    The transitional behaviors at phonetic boundaries may be affected by a longer context than the window of adjacent phones. By analysing the change in the distribution of errors compared to the baseline when more contextual information is incorporated into the learned representations, we may gain insights into the specific effects of context on the phonetic segmentation task in order to propose meaningful improvements.

- Investigate the impacts on performance when Mel-spectrogram feature engineering is combined with neural-network based representation learning;

    Conventionally, raw waveforms are used as inputs to speech representation learning networks, and no work were found to have applied engineered feature to speech representation learning. Yet, spectrogram-based engineered features contain valuable information on phonetic variations. Thus we would like to investigate the potential impact of the combined usage of both learned and engineered features on the segmentation performance, by replacing a convolutional block with Mel-spectrogram feature extraction.

- Compare model performances when trained and/or tested on different datasets;

    Automatic phonetic speech segmentation models are mostly trained and evaluated on TIMIT (a read speech corpus designed for phonetic analysis), and sometimes with Buckeye (a conversational speech corpus), which are the only two commonly used English corpora that provide transcriptions at the phone level. Yet, thorough evaluation of a model should consider performance variations due to differences in the dataset.

1.3 Contributions

1. The first speech representation learning based CNN model on the unsupervised phonetic speech segmentation task, proposed by Kreuk *et al.* [22], was closely investigated:

   (a) Model performances on different generic phonetic classes were analyzed in detail, which offered implications on future improvements;

   (b) The learning mechanisms of the CNN was better understood via the visualization of hidden feature maps, and were shown to be effective with segmentation performance evaluated on intermediate representations given by each hidden layer.

2. The effects of incorporating global contextual information into the learned representations were investigated via the addition of a GRU layer into the model:

   (a) Considering a longer context at each output frame was shown to slightly reduce phonetic segmentation performance, unlike in ASR;

   (b) A longer context can effectively reduce over segmentation rate on all phonetic classes, and the overall reduction in performance with longer context per frame was attributed to the decrease in the total number of predicted boundaries.

3. The power of CNN to capture underlying speech features was validated in comparison to conventional feature engineering:

   (a) The conventional combination of raw waveform inputs with CNN was testified to yield the best performance;

   (b) When the learning of speech representations through the CNN was based on Mel-spectrogram feature inputs, segmentation performance slightly reduced, though still better than previous unsupervised segmentation methods;

   (c) The significance of good representations in ML tasks was illustrated by the large gap in performances between direct segmentation on Mel-spectrogram features and segmentation on CNN-learned representations.

4. Different combination of training, validation, and testing datasets were evaluated:

   (a) The quality of additional training data (LS clean and other) contributes positively to the model's performance on TIMIT;

   (b) The model was shown to generalize well across datasets, giving (1) similar performances on TIMIT when training was done on LS or WSJ, and (2) adequate performance when model trained on TIMIT was tested on WSJ;

   (c) The model was trained and tested completely on WSJ with forced-aligned transcriptions for validation and testing, which gave lower performance compared to TIMIT;

In this report, **Chapter 2** introduces the essential background of the task and the context of related works. Major methodologies are explained in **Chapter 3**. In **Chapter 4**, experimental results are presented and discussed to fulfill the above mentioned objectives, and we reach final conclusions in **Chapter 5** with suggestions on next steps to be taken in the future.

Chapter 2

Background

2.1 Technical Background

2.1.1 Phonetic Speech Segmentation

Automatic phonetic speech segmentation is the task to partition speech into its constituent phoneme units [32]. These phoneme units are discrete and non-overlapping, each defined by two boundaries, one on each side of the phoneme. To conduct phonetic segmentation on a given speech utterance, the segmentation model outputs the positions in the utterance where it predicts such boundaries would occur.

2.1.1.1 Phoneme

In phonology, phonemes are the basic units of analysis; they are abstract categories of phonemic contrasts, or of sounds that belong together in the mind of the speaker [20]. These phonemes can be distinguished by humans and speech recognition systems because a single sound change in a "minimal pair" of words is represented by two different phonemes in the language. For example, English words *pirate* and *pilot* is a minimal pair with only one difference in pronunciation, where the two different sounds represent two phonemes /r/ and /l/. This definition of phonemes by the differences in sounds lay the bedrock for the task of determining boundaries between them – acoustic change is a typically good measure to segment speech, such that significant change signifies a change in phonemes [32]. Meanwhile, to automatically determine phonetic boundaries, it is common and useful to assume that each phonetic segment is spectrally stable [38], meaning that the acoustic change within the phoneme is assumed to be less significant relative to that at the boundaries.

2.1.1.2 Waveform

Raw waveforms are used as inputs to the neural network model we are investigating in this project [22]. Waveforms are the digital representations of the speech sound that we hear. When speech is recorded to be stored in a computer, the signal is quantized into discrete samples at a specific sampling frequency – a fixed time interval at which points of amplitude are taken. The resulting sequence of amplitude samples in the time domain is the waveform of the input speech signal. A sampling frequency of 16kHz is commonly used for ASR corpora [11, 26, 27]. The number of frames (samples) of a waveform can be computed as the product of its duration in seconds and its sampling frequency in hertz.

2.1.1.3 Engineered Acoustic Features

Because of the common usage of engineered acoustic features in previous work on this task, understanding the extraction procedure of these features is important to better understand the effects of features under the context of related works. Here we give a typical workflow for extracting the most frequently used feature – Mel-frequency Cepstral Coefficients (MFCCs) – as inputs to previous phonetic segmentation algorithms. Firstly, we apply tapered windows with a fixed length and stride (such as a 25ms Hann window with a 10ms stride) to a given waveform, and conduct fast Fourier Transform on each window (frame) to obtain the magnitude spectrum in the frequency domain, which is where undesired phase information is removed. Then, we apply filter banks and warp the logged output energies of these filters onto a Mel scale. On the resulting log magnitude spectrum on a Mel frequency scale, we further conduct Cepstral analysis (i.e. Discrete Cosine Transformation), so that correlations between adjacent filter outputs are removed. The outcome is the MFCC feature vector for each frame, which contains uncorrelated information about the shape of the spectrum.

2.1.1.4 Supervision

The various approaches on the task of phonetic speech segmentation can be classified into supervised or unsupervised methods. In supervised approaches, we are given in advance the transcriptions of the audio data, which are the corresponding phonetic or word labels of the utterances. Most supervised approaches are based on forced alignment [32], where a speech signal is optimally time-aligned with the corresponding transcribed phonetic sequence. However, in the unsupervised case, we do not have any knowledge on the phoneme set of the target language or any phonetic transcriptions for the original speech signals. In this sense, phonetic transitions are typically inferred by the amount of acoustic or spectral changes [38]. Thus, two things are implied with unsupervised segmentation: the number of phonetic segments in the signal, and the position of each boundary which marks the transition from one phoneme to another [34]. The unsupervised setting is the focus of this project.

2.1.1.5 Challenges

Phonetic speech segmentation is a challenging task due to the high variability of speech. The analysis conducted by [32] shows the major errors found in the segmentation results that suggest three main categories of the fundamental problems in phoneme segmentation. First, phonemes vary significantly in duration (for example, a vowel is typically much longer than a plosive), so if an inflexible window of frames is considered, then a segment that's much longer or shorter than the window is likely to result in the prediction of additional or missed boundaries. Second, adjacent phonemes can be very similar in pronunciation (for instance, consecutive fricatives that are similar in terms of the manner of articulation), so the boundaries between these phonemes are less likely to be detected due to less significant acoustic change. Third, there are phonemes that are inherently dynamic, which involve transitions during pronunciation (for example, the transition from one vowel to another in diphthongs such as /ɔi/ in "boy", or from closure to burst in stops such as /t/) in "test", and therefore, they are prone to being overly segmented with extra boundaries. These fundamental problems give an overview of the challenges that may hinder the performance of automatic phonetic segmentation models.

2.1.1.6 Evaluation Metrics

Using meaningful and universal evaluation metrics are important to analyse segmentation results and compare with different models. In a comprehensive analysis on phonetic segmentation results conducted by [32], two metrics combined are claimed to be the most important quantities for evaluating hypothesized phoneme boundaries: correct detection rate (CDR) and over-segmentation (OS) rate. The former is the percentage of correctly predicted boundaries among the total number of boundaries in the ground truth (for example, manually labeled boundaries); and the latter is the total number of predicted boundaries over the true number, minus one, indicating the amount of over or under segmentation. OS is essential to show flaws of the model because a high CDR can come from a large amount of predicted boundaries, and that a high Recall with low Precision can still lead to an adequate F1-score. For this reason, the commonly reported F1-score in previous works on this task should also be amended. [29] proposed a single metric combining CDR and OS, called the *R-value*, which is used by almost all models afterwards. Equations below present formulae for the above mentioned metrics.

$$\begin{aligned}
Recall(R) &= \frac{TP}{TP+FN} \quad (= CDR) \\
Precision(P) &= \frac{TP}{TP+FP} \\
F1 &= 2 * \frac{P * R}{P+R} \\
OS &= R/P - 1 \\
R - value &= 1 - \frac{|r1| + |r1|}{2} \\
r1 = \sqrt{(1-R)^2 + OS^2}, \quad r2 &= \frac{-OS + R - 1}{\sqrt{2}}
\end{aligned} \tag{2.1}$$

where TP, TN, FP, FN represent true positives, true negatives, false positives, and false negatives, respectively. In this project, we evaluate model performance primarily based on the *R-value* metric, reported together with precision, recall, and the F1-score. Larger R-values are desired, indicating improvement in CDR (or recall) without increasing OS.

2.1.2 Deep Learning with Neural Networks

In this project, we are interested in the unsupervised neural network approach to segment speech. The application of machine learning (ML) methods has become much more common in the past decade, where people no longer use hard-coded knowledge, but instead ustilise data to train ML algorithms to solve problems. These ML models (such as classifiers using naive Bayes or logistic regressions) are capable to learn patterns from the data and carry out tasks that involve seemingly subjective decisions [12].

Deep learning is a specific field under ML that employs artificial neural networks with multiple layers to learn a hierarchy of features from large amounts of data, that can then be applied to make intelligent outputs or predictions. Deep learning networks (such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)) are good at tasks that are originally simple for humans but hard for machines, for instance, speech recognition [12].

2.1.2.1 Representation Learning

The particular network model under investigation for this project in [22] is an example of representation learning. Past experience demonstrated the importance of representations, or input features, on the performance of ML systems, which led to the development of representation learning [12]. In terms of neural network models, it is the learning of not only features that map representations to model outputs, but also the learning of the representation itself to be used for the particular task.

Specifically, in the case of phonetic speech segmentation, we consider representations to be the features used at the time of the actual segmentation (i.e. outputs from the convolutional network in [22]). Apparently, segmenting the waveforms from recorded speech directly would yield poor results, because waveforms contain unwanted variability such as phase information that hinders generic pattern recognition. Hence, we want more informative descriptions of speech that have any unnecessary variability removed, while retaining all important information for distinguishing segments of phonemes.

Before Kreuk's work [22], past phonetic segmentation algorithms tend to use manually engineered spectrogram-based features as inputs, such as MFCCs or band-energies from mel-scaled filter banks [7, 36, 25, 15]. Yet, hand-designed representations in general tend to perform worse than representations learned automatically through deep learning models [12]. Besides, most learned representations are also capable of adapting to new tasks with least manual effort. Particular to this project, [22] has proven that the learned representation through a 5-layered CNN is more powerful than engineered features, by achieving a better performance on the unsupervised phonetic segmentation task.

2.1.2.2 Self-supervised Learning

Self-supervised learning (SSL) is a type of unsupervised learning. In terms of labeling information provided with the data, they are the same that no label or transcription is available for training at all. However, SSL is characterized by the provision of pseudo-labels (such as the definition of positive and negative samples in Contrastive Predictive Coding (CPC), as described later this chapter), from which the network can learn to do tasks that are commonly more suitable for supervised learning, such as image classification and speech recognition. SSL has been especially successful in speech representation learning in recent years ([33, 3, 4, 42]) with CPC [37].

In this project, the term *unsupervised* refers to the task of phonetic speech segmentation, which is the general research area known as *blind segmentation*, with or without the application of neural networks. Under this area, self-supervision specifically refers to the learning framework of the investigated neural network model that is applied to solve unsupervised segmentation.

2.2 Related Work

2.2.1 Previous Approaches

The task of automatic phonetic speech segmentation can be categorized into supervised and unsupervised approaches. In general, the state-of-the-art model in the supervised setting performs better than that in the unsupervised setting. This gap between performances was approximately 10% in the R-value in 2020, given by the unsupervised model in [40] and the supervised model in [23], which both utilised recurrent neural network. After the proposal of the investigated CNN model ([22]) in the unsupervised setting, this performance gap has decreased significantly to 5% in the R-value.

Forced alignment is the most commonly used method in the supervised case [23]. However, compared to unsupervised approaches, it has several limitations: the alignment requires the phoneme transcriptions not only for training, but also at inference time, and the trained model is dependent on the particular phoneme set, which would limit the capability of the model to transfer well to different datasets or languages [23]. Another kind of supervision to mention involves using phoneme transcriptions to aid the segmentation algorithm indirectly for performance improvements [38, 23]. Still, these supervised methods rely heavily on the transcriptions for good performance. These limitations of supervised approaches on phonetic speech segmentation correspond to the various motivations behind unsupervised methods as mentioned in the introduction, and illustrate the need for unsupervised models.

Under the unsupervised category, techniques vary significantly, and we explain representative previous works in the following paragraphs.

The first work to mention belongs to the conventional methods that aim to detect significant points of acoustic change by operating directly on extracted MFCC features. The method in [7] employs a maximum margin clustering (MMC) algorithm that splits frames in each sliding window of 18 MFCC vectors into two clusters, and boundaries are placed based on the combination of two metrics computed for all windows over the signal: the Euclidean distance between each two clusters, and whether the shift of the 18 cluster labels over consecutive windows results in a diagonal structure. Thus, peaks of Euclidean distances and middle vectors of the diagonal structures are signals for notable acoustic change where boundaries should be placed. This method proposed in 2007 achieved an R-value of 80% on TIMIT (reported by [36]). It was analysed thoroughly by [32], which illustrated the problem of over-segmentation due to the high sensitivity of MMC to very subtle changes within a phoneme segment, and showed that a fixed window length of 90ms would likely result in under-segmentation as only one boundary can be found in a window using two clusters. Improvements on the MMC algorithm were clearly desired.

[36] achieved the same R-value as the above method in [7], but with a novel approach where the spectrogram is processed into both image-based and acoustic-based features. The combination of the color variations in the processed spectrogram and the magnitude variations in the Mel-cepstral coefficients together indicates the spectral, or acoustic change for segmenting speech. It reports performance separately that an R-value of only 48% is achieved when only acoustic features are used, but image-based features alone achieves 74%. This implies that Mel-cepstral coefficients may lack information significant to distinguishing phoneme segments, and more robust and informative acoustic features are required for better performance.

A third variation of the segmentation method, achieving an R-value of 81%, is the one proposed in [15] based on Legendre polynomial approximation. It uses the normalised band-energies from 8 Mel-scaled triangular filter banks, which are obtained by applying the filters to the spectrogram. The energy of each band is modelled with Legendre polynomial coefficients to a maximum degree of 3, and the variations of which are indicators of spectral change, for locating phoneme boundaries. This is a highly mathematical based algorithm, which is again, much different from the previous two methods, while achieving a similar performance.

After the proposal of the above conventional methods that work with algorithms which directly detect significant acoustic change based on the engineered features, unsupervised models that use a Recurrent Neural Network (RNN) approach were introduced.

The work by Michel *et al.* tries to identify phonetic boundaries at places where the error returned by a sequence prediction RNN model is maximal, taking MFCC features as inputs [25]. However, the R-value evaluated on TIMIT given by this method is lowered to 78.8%. It can be said that the much longer contextual information given by outputs of the RNN might be adding distracting information that are less helpful for determining transitions at the phonetic level. Meanwhile, because predictions given by the RNN are based on preceding timesteps, the starting 70ms of the error outputs are claimed to be meaningless and set to 0, which should be considered as a weakness of such recurrent models.

On the other hand, Wang *et al.* proposed a novel RNN-based approach that enhanced performance to an R-value of 83.16%, when Gated Activation Signals (GAS) within each Gated RNN unit are utilised in combination with error outputs from the recurrent predictor model [40]. It is interesting to observe the correlation that exists between the GAS over time and the phoneme boundaries, where a sudden change in GAS is shown to imply changes in the acoustic signal.

Both of these RNN-based models also follow the same principle as conventional methods, that significant acoustic change implies a change in phonemes. Yet, though RNNs are powerful for many tasks performed on sequential data [1, 33, 12], its robustness and suitability for the unsupervised phonetic segmentation task should be further examined due to the limited amount of previous work and varied results on this specific topic.

We can see that performances of the above methods overall showed an improvement of only approximately 3% in the R-value over a decade from 2007 to 2017, despite attempts on using various different segmentation algorithms or engineered features. Hence, there is significant room for improvement for the unsupervised setting of phonetic speech segmentation, and, we may consider the model investigated in this project and its use of representation learning as a "milestone" for this particular research topic.

2.2.2 Origin and Rationale for Current Approach

We mainly focus on the model proposed in Kreuk *et al.*, which is a novel approach under this topic that achieved the state-of-the-art result at the time of publication with an R-value of 86% [22], which increased performance on unsupervised phonetic speech segmentation by the same amount of increase (3%) that other previous works took a decade to achieve.

There are several techniques involved in understanding the effectiveness of this model ([22]), and the following paragraphs would explain the rationale behind each of these techniques, in the order of their approximate development timeline.

The work on Contrastive Predictive Coding (CPC) [5] would be one of the major breakthroughs that led to the widespread of self-supervised representation learning. CPC is especially good at learning more robust representations with neural networks, through the minimization of a *contrastive loss* based on the Noise Contrastive Estimation principle ([14]) during training. There are many variants of the formulation of this loss for different networks, and the key idea is: given a frame of output o_i, a positive sample p is defined and forms a positive pair with the current output frame, and 'noise' is constructed by forming negative pairs with (randomly) selected samples n from other output frames, and thus the 'contrast' of the positive pair against the 'noise' can be maximized via minimizing a negative log loss. Equation 2.2 shows a general formula typically used, for the loss of outputs from one piece of data input; where *sim* is a similarity function such as cosine similarity, and *exp* is the exponential function.

$$L = -\sum_i log \frac{exp(sim(o_i, p))}{exp(sim(o_i, p)) + \sum_n exp(sim(o_i, n))} \tag{2.2}$$

For example, in [22], the positive sample is the output frame from the CNN that's adjacent to the current frame, while distractors are K randomly chosen non-adjacent frames (details will be presented in Chapter 3). Through the concept of maximizating 'contrast' in CPC, network outputs are then capable of removing distractive information, while retaining important information that has positive contributions to the task. The design of learned representations using the principle of CPC in [22] exploits and reinforces the characteristic of phoneme segments that is of particular importance to our goal of detecting phonetic boundaries — the resulting representations tend to remain stable within the phoneme and change when transitioning, and hence they are considered robust acoustic features for the task [25].

Unsupervised speech representation learning, with techniques including CPC, has become widely popular and has achieved much success only within the past three years [37, 33, 3, 22, 2]. These works have demonstrated that learning robust and generic representations of speech can significantly improve the performance of different downstream tasks [42], and can be effectively and easily transferred well across different languages [30], therefore also increasing the efficiency of resource utilization.

Among the many neural network architectures for speech representation learning, the Convolutional Neural Network (CNN) is an essential component of recent state-of-the-art models [42]. We discuss the commonly studied wav2vec models as examples in this case [33, 3, 4], which have offered inspirations for the design of Kreuk's model [22].

wav2vec, VQ-wav2vec, and wav2vec 2.0 are a series of models originally developed by Facebook for the ASR task. They all take raw waveforms as inputs, encode them with a CNN, pass the encoded features through a context network that further extracts information from a wider context, and output speech representations; they are trained via optimizing a contrastive loss. The fact that they are highly successful on various downstream tasks [2, 42] validates the effectiveness of using CNN as a feature extractor. Kreuk *et al.* exploits this idea and applies CNN directly on raw waveforms to learn representations, also through a contrastive loss, that are optimal for phonetic segmentation [22]. This combination of raw waveform inputs and CNN may be justified as CNNs learn more robust features that are temporally invariant due to parameter sharing [18].

Both VQ-wav2vec and wav2vec 2.0 applied a quantization module to discretize features extracted from the CNN network. Evaluation on similar self-supervised pre-training models by Baevski *et al.* has shown that the quantization of latent representations into discrete units can improve the recognition accuracy for ASR. However, on the phonetic segmentation task, the additional quantization of representations in [21] on the same model architecture as in Kreuk *et al.*, lowered segmentation performance instead. We may reason that differently from the recognition task, we need to capture subtle changes at the frame level in order to better discriminate between adjacent phonemes, whereas quantization potentially gets rid of many of these subtle differences.

On the other hand, the context network in wav2vec models has been experimentally proven to be unhelpful for our task either in [22]. Since each phoneme is affected by a much smaller span of adjacent time frames compared to words, it would be less ideal to extract utterance-level contextual features which we normally do for ASR.

The application of CNN on raw waveforms and the technique of CPC together form the central architecture of the current model by Kreuk *et al.*, and contribute to the rise in performance on the unsupervised phonetic segmentation task compared to previous conventional models. Nevertheless, the segmentation algorithm performed on the output representations from the network at inference time is rather simple: the negative cosine similarities for each pair of adjacent frames are computed, and boundaries are predicted at the peaks of these scores, which are where the dissimilarity exceeds a certain threshold. Similarly to conventional methods, acoustic change is still the indicator for inferring phoneme boundaries, and are implied by the dissimilarity scores here computed by the peak detection algorithm. The simplicity of this algorithm suggests that the improvement in segmentation performance may be attributed to the learned representations, that they are better for phoneme boundary detection than the previously used spectrogram-based engineered features. We will focus on this idea and investigate from different aspects the potential reasons behind the effectiveness of the learned representations on phonetic speech segmentation.

Chapter 3

Methodology

3.1 Neural Networks

3.1.1 Convolutional Neural Network

A Convolutional Neural Network, or CNN, is a special type of deep learning neural networks for processing data with a grid-like topology [12]. It is characterized by the use of a Convolution operation in a convolutional layer, which is a linear operation on two functions: x that gives the input matrix based on the receptive field, and w that gives the weight matrix, also known as the *kernel*. The convolution output at point t in the discrete case is given by: $(x * w)(t) = \sum_a x(a)w(t-a)$, where $*$ denotes the convolution operation, and a implies the correspondence between values in x and w [12]. In practice, deep learning libraries such as PyTorch use cross-correlation (denoted by $\star$) for more efficient implementation, with which kernel weights are simply reflected horizontally and vertically. Also, an optional bias term b can be added so that outputs of the convolutional layer are allowed to be shifted. The formula we actually use for a convolutional layer now becomes:

$$(x \star w)(t) = \sum_a x(a)w(t+a) + b \tag{3.1}$$

In a convolutional layer, a kernel takes regular strides through layer inputs and performs the convolution operation, and the output is passed through an activation function (as explained later), to produce a *feature map*. Each layer can have multiple input and output channels, which means multiple kernels are applied so that different feature maps can be generated. The stacking of kernels of all the channels forms the *filter* of that convolutional layer. Visualisation of feature maps and filters may provide insights into the internal learning mechanisms of CNNs and has been shown to be helpful for understanding the feature extraction process for CNNs that process images [43].

The dimensionality of feature maps and filters depends on the size of the kernel (K), the stride (S), size of optional padding on inputs (P), and the number of output channels (C_{out}). The general equation for computing the output length of convolution in a certain dimension is given by:

$$\frac{(L+2\times P-K)}{S}+1 \tag{3.2}$$

where L is the input length in that dimension. In practice, the floor function is applied to ensure we have integer output dimension.

Speech in the form of either waveforms or spectrogram-based features is time-series data that's suitable for 1D convolution, where the kernel takes strides at regular intervals in the time domain. For example, given a raw waveform of one channel with length L, passing the waveform through a 1D convolutional layer with output channels $C_{out} = 256$, a kernel size $K = 10$, and a stride $S = 5$ indicate a filter of dimension $(256, 1, 10)$ and would yield 256 feature maps each of length $\frac{(L-10)}{5} + 1$.

3.1.1.1 Parameter Sharing

In the operation of a convolution layer, parameters, or weights of a kernel, are shared by applying the same kernel over the entire length of the input, so that weights are no longer specific to each input unit as in tranditional neural networks like Multi-layer Perceptrons [18]. This concept enables CNNs to be equivariant to translations, meaning that output feature maps vary in the same way the inputs vary [12]. This property also grants CNN the power to operate well on raw inputs like waveforms without any pre-processing.

3.1.2 Recurrent Neural Network

A Recurrent Neural Network (RNN) has the power to represent information over time, which is most suitable with sequential data input. In understanding speech, which is naturally a sequence of phonemes or words, contextual information is often important. In RNN, input vectors are processed one by one in sequential order, with hidden states h representing the 'memory' of information at each time step. Hence, information in the past is passed on and combined with the current input in each cell of an RNN layer, which then further contributes to the learning of weights at future time steps. Weights are then updated via back-propagation through time. Figure 3.1 shows a simplified flow of information in an RNN, where $x^{(t)}$ and $h^{(t)}$ represents the input and hidden state at time t, respectively.

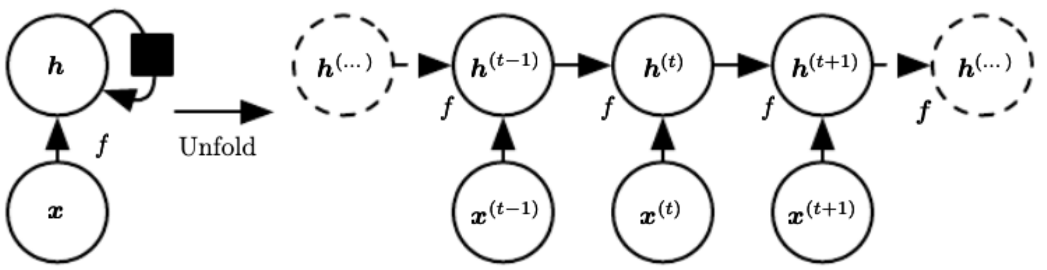

Figure 3.1: Simplified architecture of an Recurrent Neural Network [13].

However, due to the multiplication of gradients in the backpropagation through time, an RNN is prone to the vanishing/exploding gradient problem. Therefore, gated RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are proposed as common variations of RNN to solve this problem. Particularly, GRU is used as a modification to the model in this project.

3.1.2.1 Gated Recurrent Unit

A Gated Recurrent Unit (GRU) network is a type of RNN that is similar but simpler than an LSTM network. It differs from RNN only in the internal architecture of the recurrent cells, so it also has the power to represent information over time just like RNN. The key to GRUs is the use of gates within each cell, which regulates the interpolation of previous hidden state and the current input to produce the new hidden state [6]. Thus, gates are trained to keep the most relevant information in the 'memory' of a hidden state. Each GRU cell contains 2 gates — the reset gate and the update gate. Figure 3.2 illustrates the architecture within one hidden GRU cell.

The **Reset Gate** r_j of the j^{th} hidden unit controls the amount of information to forget from the previous hidden state ($h_j^{<t-1>}$). It outputs a value between 0 and 1 with the logistic sigmoid function:

$$r_j = \sigma([\mathbf{W}_r x]_j + [\mathbf{U}_r h^{<t-1>}]_j) \tag{3.3}$$

where $\mathbf{W}_r$ and $\mathbf{U}_r$ are weights to be learned, and x is the corresponding input to the j^{th} unit. Thus, the closer r_j is to 0, the more it forgets from the past.

The **Update Gate** z_j then controls the interpolation of new input and past information, enabling the network to have long-term memory:

$$z_j = \sigma([\mathbf{W}_z x]_j + [\mathbf{U}_z h^{<t-1>}]_j) \tag{3.4}$$

where $\mathbf{W}_z$ and $\mathbf{U}_z$ are also weights to be learned. The hidden state $h_j^{<t>}$ is then:

$$h_j^{<t>} = z_j \odot h_j^{<t-1>} + (1 - z_j) \odot \tilde{h}_j^{<t>} \tag{3.5}$$

$$\tilde{h}_j^{<t>} = \phi([\mathbf{W}x]_j + [\mathbf{U}(r_j \odot h^{<t-1>})]_j) \tag{3.6}$$

where ϕ represents the *tanh* activation, and $\odot$ represents the Hadamard (element-wise) product. The actual realisation of the GRU layer is implemented with the PyTorch `torch.nn.GRU` class, with which bias may be added as a learned parameter.

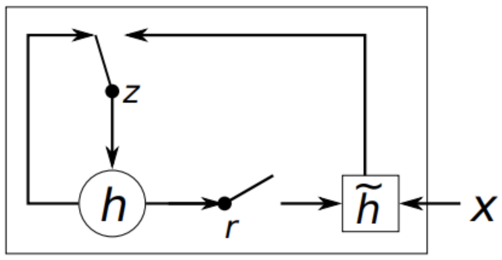

Figure 3.2: The Reset and Update gates in a hidden unit (cell) of a GRU network [6].

3.1.3 ReLU Activation

Activation functions are essential for deep learning networks. They introduce non-linearity to the network, which is important because operations including convolution and affine transformations through weight matrices are linear; so if we want the network to learn more complex patterns (with non-linear decision boundaries for instance), non-linearity must be added onto outputs of hidden units via activation functions.

There are many different activation functions available; *sigmoid*, *tanh*, and *Rectified Linear Unit (ReLU)* are the most common ones. Among these, ReLU is shown to have achieved better empirical results than *sigmoid* or *tanh* [12].

In CNN, each convolution layer is followed by activation. *Leaky ReLU*, a variant of ReLU, is applied in each activation layer, in the CNN used in this project. Leaky ReLU solves the saturation problem found in *sigmoid* and *tanh*. Compared to ReLU, it gives non-zero outputs on negative inputs, so that the "dying ReLU" problem due to the zero gradient on negative inputs in ReLU activation is corrected, which allows strong learning signals to be given also on negative values. Equation 3.7 specifies the Leaky ReLU function and its gradients, where α is a pre-defined small number that defaults to 0.01 in the PyTorch `torch.nn.LeakyReLU` layer.

$$LeakyReLU(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \qquad \frac{d}{dx}LeakyReLU(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{3.7}$$

3.1.4 Batch Normalization

Normalization techniques bring better performance and significant decrease in the training time for a neural network model [35]. Batch Normalization (BN) is a type of normalization applied as layers throughout the neural network, normalizing each training mini-batch with learnable parameters. Deep learning networks are prone to internal covariance shift and the exploding/vanishing gradient problem, which both make training extremely slow. The prior is the varied distribution of activations when weights are updated during training due to inputs from previous layers that are not standardized to a certain scale [17]. The latter is caused by multiplying weights during back-propagation that might result in extremely large/small numbers. BN may effectively alleviate these issues by first normalizing outputs of the respective layer based on the mean μ and standard deviation σ across a mini-batch, and then shift and scale the normalized outputs with learned parameters β and γ [17]. The formulae below illustrate the workings of a BN layer, where u_i is the input from previous the hidden layer on the i^{th} feature dimension.

$$batchNorm(u_i) = \gamma_i \hat{u}_i + \beta_i \tag{3.8}$$

$$where \quad \hat{u}_i = \frac{u_i - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} \tag{3.9}$$

In our experiments, outputs from each convolution layer is first passed through a batch normalization layer before applying the Leaky ReLU activation.

3.1.5 Fully Connected Layer

A Fully Connected (FC) layer in a deep neural network is a linear transformation that connects all units of the input layer to all units of the output layer. It learns a mapping that would assemble the extracted features, given by a CNN for example, at the end of the neural network to give the final output, such that the network is now trained end-to-end. FC layer also allows the dimensionality of features to be altered to the desired output dimension (such as the number of classes in a classification task).

3.2 Contrastive Loss

Contrastive loss refers to the loss used in the technique of CPC, where this loss is optimized with self-supervised learning to update network weights. We define the contrastive loss specific to the particular model under consideration [22]. Given a waveform input $\mathbf{x}$ of an utterance to be segmented, we obtain its corresponding representation $\mathbf{z}$ by passing it through the neural network, and depending on the length of the input sequence, $\mathbf{z}$ has a total of L frames. Each z_i (the i^{th} frame in $\mathbf{z}$) is a 64-dimension vector that has a receptive field of 30ms, and $\mathbf{z}$ has a frame rate of 10ms of the original waveform input. The loss L_i is then computed on each output frame z_i over each sequence of output representation $\mathbf{z}$, for $i = 1...L-1$. It can be given as an adapted version of Equation 2.2:

$$L_i(z_i,\, D_K(z_i)) = -log \frac{exp(sim(z_i,\, z_{i+1}))}{\sum_{z_j \in \{z_{i+1}\} \cup D_K(z_i)} exp(sim(z_i,\, z_j))} \tag{3.10}$$

where *sim* represents the cosine similarity function over the two vectors, and *exp* is the exponential function. Given a batch size of M, we denote the m^{th} output in the batch by $\mathbf{z}^m$. With reference to [22], the positive sample is defined as the adjacent frame z_{i+1} of z_i, and $D_K(z_i)$ denotes the negative ‘distractor’ samples, which are K randomly selected non-adjacent frames in the same output sequence $\mathbf{z}^m$, such that $D_K(z_i) \subset D(z_i)$ with $D(z_i)$ being the set of all non-adjacent frames to z_i in $\mathbf{z}^m$:

$$D(z_i) = \{z_j : |i-j| > 1,\, z_j \in \mathbf{z}^m\} \tag{3.11}$$

The value of K is set to 1[1] (authors of [22] experimented different values for K but concluded that there were no significant change in performance).

Therefore, the overall loss L across a batch of inputs is:

$$L = \sum_{m=1}^{M} \sum_{z_i \,\in\, \mathbf{z}^m} L_i(z_i,\, D_K(z_i)) \tag{3.12}$$

The minimization of the above contrastive loss implies that the representations have been trained to contain information that optimally discriminate between non-adjacent frames and adjacent frames. Thus, at inference time, when two adjacent frames z_i

[1]Different values for K were tried when reproducing results, but increasing K up to 10 brought improvements of less than 1% on R-value, so the given value found in the code repository was used to report baseline results.

and z_{i+1} give a low cosine similarity value, the model would consider them as 'non-adjacent', meaning that they are more likely to belong to different phoneme segments and a boundary is more probable between them. This is the reason why the peak detection algorithm described in section 3.3 places boundaries where the dissimilarity score $(-sim(z_i,\ z_{i+1}))$ across the output sequence $\mathbf{z}$ exceeds a threshold.

3.3 Peak Detection Algorithm

A peak detection algorithm is applied at inference time on the representations (Z) learned from the neural network, to determine locations of phonetic boundaries [22]. The peak detection algorithm first scores the *dissimilarity* between each pair of adjacent frames z_i and z_{i+1}, for $i = 1...L-1$, as the negative cosine similarity (sim) between them (Equation 3.13).

$$score(z_i) = -sim(z_i, z_{i+1})[22] \tag{3.13}$$

The sequence of these scores signifies places where adjacent frames are significantly dissimilar in terms of their representations (i.e. where the score peaks and its relative differences to scores of nearby frames exceed a certain threshold), which have been optimized during training for capturing acoustic change and recognizing phonetic transitions.

Next, the peaks of these scores are located using the function `scipy.signal.find_peaks`, which we call on the sequence of $score(Z)$ with the prominence, width, and distance parameters, to obtain the indices i of detected peaks. The prominence is defined as the minimal height difference required between the height at current frame and its lowest contour line to consider it as 'peak' [39]. Similarly, minimal width of the peak and minimal distance between adjacent peaks required are set by the width and distance parameters. A range of prominence values in [0, 0.15] with a step size of 0.01 are tried at validation time to find the best prominence value. For either width or distance, only two values, None and 1, are tried for validation.

During testing, we apply a tolerance window of 20ms, which means a predicted boundary within 2 frames in the representation space to a ground truth labeled boundary is considered as correct. This tolerance has been validated and commonly used in previous works, that 20ms is in general the span which manual segmentations agree with each other [41].

3.4 Data

This section introduces the datasets used in our experiments.

3.4.1 TIMIT

Previous works on phonetic speech segmentation almost all evaluated their systems on the TIMIT dataset, which has been a standard dataset for comparing performances on phone-based speech processing tasks such as phone classification and recognition.

TIMIT is a corpus of read speech for acoustic-phonetic studies [10]. It contains sentences read by 630 speakers of eight dialect regions of American English, with a 7:3 male:female gender split overall. Each speaker contributed to the recordings of 10 sentences. There are three types of sentences in TIMIT: (1) dialect sentences, (2) phonetically-compact sentences, and (3) phonetically-diverse sentences. (1) are read by all speakers to exemplify the dialects, and are conventionally excluded from training and testing. (2) are designed specifically for better coverage of phonetic contexts, and (3) are selected from existing text corpora to add diversity. Overall, there are 3696 utterances (i.e. sentences) in the training set (approximately 3.14 hours), and 1344 utterances in the testing set (around 0.81 hour). Of all utterances in the training set, we used a 90% training and 10% validation split.

The utterances are recorded with a sample rate of 16kHz with 1 audio channel, saved in SPHERE format. TIMIT offers the manually labeled phonetic, word, and orthographic transcriptions aside from each waveform file. We use the phonetic transcription (.phn) files as the ground truth segmentation boundaries when evaluating the model on TIMIT. The phonetic transcriptions use a revised set of the ARPAbet phonetic transcription codes, containing 20 vowels, 38 consonants (with each of the 6 stops split into the release and the closure parts), and 3 labels for silence [10]. In total, these 61 phones form the original complete TIMIT phone set. Although in many phonetic recognition tasks, phone set reduction is applied to reduce confusability, the actual phone labels were irrelevant for the segmentation task in this project and the complete set was not reduced. Mappings of TIMIT phone labels to IPA symbols are given in Appendix A.

3.4.2 LibriSpeech

LibriSpeech (LS) is also a corpus of read English speech, but it is more often used for the speech recognition task and has achieved great performances compared to other ASR corpora [26]. LS contains speech read from audio books (from the part of the LibriVox project). Data in the corpus is gender-balanced, and speakers are divided into two groups – ‘clean’ and ‘other’, based on their respective WER (word error rate) evaluated using an acoustic model trained on a subset of the WSJ corpus. Speakers with lower WER are assigned to the ‘clean’ set, while those with higher WER are assigned to the ‘other’ set. In general, the ‘clean’ set of speakers are those that gives a higher recording quality and with dialects similar to standard American English. There are in total 960 hours of training audio data, which is split into the train-clean-100, train-clean-360, and train-other-500 subsets. Besides, there are two development sets (dev-clean and dev-other, each read by 40 speakers) and two testing sets (test-clean and test-other, each read by 33 speakers), each containing around 5 hours of speech [26].

LS uses the same 16kHz sample rate for recordings, and provides transcriptions at the word level. In our experiments, we take only the (.flac format) audio files in LS to use as unlabeled data for training the neural network model, but not for validation or testing.

3.4.3 Forced Alignment

There are few English speech corpora with transcriptions at the phonetic level. Past works on automatic phonetic segmentation in English are almost all evaluated on the TIMIT corpus, and only sometimes on Buckeye (a phonetically labelled conversational speech corpus). More commonly, popular ASR corpora such as LibriSpeech and WSJ offer transcriptions at the word level. Although they can be used for training blind segmentation models, there must be some "ground truth" labels of phonetic segments in order to conduct evaluation on these corpora. These phonetic transcriptions may be generated with forced alignment, where word-level transcriptions are used to align and label the audio data at the phonetic level.

Kaldi, an open-source speech recognition toolkit, was used to generate forced-aligned phonetic transcriptions for both TIMIT and WSJ [28].

3.4.4 WSJ

We also train and test the performance of the segmentation model on the Wall Street Journal (WSJ) corpus. The spoken portion of WSJ, first released in 1992, was the first general-purpose, large-vocabulary English speech corpus, and continues to provide various performance benchmarks on the ASR task [27]. We use the standard *SI-284* set for training, and the *dev93* and *eval92* datasets as the validation and testing sets, respectively; all of which have a vocabulary size of 20K. The recordings are stored in SPHERE format, recorded also at a 16kHz sample rate [27]. We use the first channel of audio files (.wv1), and convert them to WAV format before inputting into the model (with the sph2pipe tool in Kaldi). The *SI-284* training set implies speaker-independent data from 284 speakers; it contains around 80 hours of speech with 37318 utterances, which are used without any transcriptions. The *dev93* and *eval92* sets contain approximately 1.1 and 0.7 hours of speech, and 503 and 333 utterances, respectively [16]. These sets were used with transcriptions for validation and testing purposes. Since WSJ provides only transcriptions at the word level, phonetic transcriptions for waveforms in *dev93* and *eval92* were obtained by doing forced alignment on an acoustic model trained on *SI-284* using Kaldi.

3.5 Adaptation of Code and Difficulties Met

To reproduce the model results in [22], the open source code published with the paper on GitHub[2] was used. This training code uses the *PyTorch Lightning* framework, which has the same capabilities as the traditional *PyTorch* framework, only with a much higher-level API. However, due to the use of relatively outdated versions of packages in the original code, the training environment could not be built initially with the original package versions on either the Linux cluster or the local environment. Afterwards, I had built the environment with newer packages, and adapted the code to remove deprecated functions, but an error with the memory usage that occurred after a dozen of training epochs was not resolved. I had decided to rewrite the code with *PyTorch*, because the

[2]https://github.com/felixkreuk/UnsupSeg

high-level encapsulation of functions in *PyTorch Lightning* was harder to understand due to my lack of experience with *PyTorch*. Yet, while debugging the rewritten code, which took a quite long period of time, the environment was finally built successfully on the cluster with the help of a tutor, by changing only the version of *PyTorch* to allow compatibility with the NVIDIA CUDA drivers. Hence, later modifications to the code were done with *PyTorch Lightning* version 0.7.6, with the help of an older manual since the documentation of this version is no longer available on the official site after major changes were made [8].

Chapter 4

Experiments

Each section in this chapter aims to achieve each objective defined in Chapter 1.

4.1 Baseline Experiments

We refer to the self-supervised CNN model proposed in Kreuk *et al.* ([22]) as the "baseline model" that we have based our experiments and analysis upon. First of all, results from [22] are reproduced on the TIMIT dataset, in order to investigate this baseline model closely. In this section, we explain the model architecture in detail, and then describe the steps used and challenges met for reproducing results; afterwards, we show baseline model performance and conduct error analysis on the segmentation outcomes.

We first describe specifications of this model according to the work in [22]. The overall architecture of the model is demonstrated in 4.1.

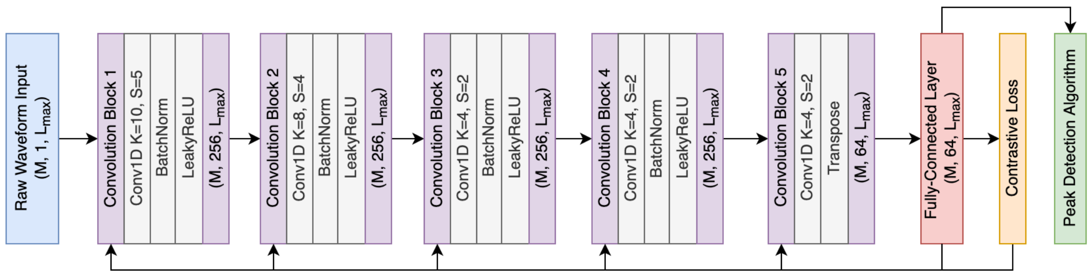

Figure 4.1: Architecture of the neural network model proposed in [22]; values in brackets indicate output dimensions of the layer.

The main body of the neural network is a 5-layered CNN, which is followed by a fully connected layer to output learned speech representations on a given waveform input. Specifically, each of the five convolution blocks that together constitute the CNN contains three layers – a 1D convolution layer, a batch normalization layer, and an activation layer. The convolution layers operate with the parameters (kernel size, stride size) of: (10, 5), (8, 4), (4, 2), (4, 2), (4, 2), (4, 2), respectively in order. The number of output channels is 256, the same for each convolution layer. Each output channel is then normalized over the input batch by a Batch Normalization layer. In the activation layer, the Leaky ReLU function is applied on batch normalized convolution outputs to add non-linearity.

At the input layer, audio files were directly loaded with `torchaudio` in raw waveform format; they have a sampling rate of 16kHz (i.e. one second of audio corresponds to 16,000 frames of input) and 1 audio channel (i.e. the input sequence is a 1-dimensional real vector of T frames, where T is different for utterances that have different lengths). We denote each input sequence of frames by $\mathbf{x} = [x_i]$ for $i = 1...T$. A batch size of 8[1] was used, so that input waveforms from 8 audio files are processed together for a single update of network weights; and the input waveforms in a particular batch are padded to the same length – the maximum length among inputs of the current batch (T_{max}), using a zero-padding strategy.

We use the same notation for the learned representations as in [22], and denote outputs of the network by Z. Output $\mathbf{z}$ corresponding to each input $\mathbf{x}$ has a length of L frames, such that $\mathbf{z} = [z_{ij}]$ for $i = 1...L$ and $j = 1...64$, so each frame z_i of the representation $\mathbf{z}$ is a 64 dimensional vector. The 256 dimensional output from the CNN blocks is mapped to 64 dimensions to obtain Z via a Fully Connected layer.

The actual length L for each $\mathbf{z}$ (ignoring padded units) can be computed from the actual length T of the matching input, using Equation 3.2 with kernel and stride sizes of the convolution layers. In our case, one second of input waveform, which is equivalent to $T = 16,000$ frames, would give an output length of $L = 98$, which means that the new frequency as in the representation space (frequency $f = \frac{L\,frames}{n\,seconds}$) is now reduced to 98Hz. This means that the frame of $\mathbf{z}$ is approximately 10ms ($\frac{1}{f} = \frac{1}{98} = 0.0102s$) of the audio in the input space. We can also determine the receptive field of each frame of the representation via the inverse computation of Equation 3.2 backwards through the convolution layers. In this case one frame of $\mathbf{z}$ is processed from 465 frames of $\mathbf{x}$, which represents approximately a receptive field of 30ms ($\frac{465}{16kHz} = 0.029s$).

The technique of CPC is employed here such that a contrastive loss, as explained in Chapter 3, is used to optimize network weights in a self-supervised fashion. The loss L_i on each output frame z_i, for the m^{th} output $\mathbf{z}^m$ in the batch of size $M = 8$ is computed via Equation 3.10, and the loss across a input batch is summed over all frames in all utterances in the batch.

At training time, weights of the network are updated on each batch of inputs via back-propagation with the objective to minimize the contrastive loss. The TIMIT dataset is used to train, validate, and evaluate the baseline model (with the 'SA' dialect sentences removed). The model was trained for 50 epochs, and as TIMIT is a quite small corpus, our experiments showed that the model converged fast. The Adam optimizer was used with a learning rate of $1e-4$, and a `StepLR` scheduler[2] was used without applying

[1]Different batch sizes were tested, though slightly larger batch size improved performance, the improvement (within 1% in R-value) can be neglected and the original batch size of 8 as in the paper was used.

[2]For experimentation, a better scheduler `ReduceLROnPlateau` was tested, but the improvement was trivial from the original setup

Model	Data	prominence	Precision	Recall	F1	OS	R-val
Kreuk *et al.*	TIMIT	0.05	83.34	83.60	83.47	0.31	85.89

Table 4.1: Performance of the baseline model on TIMIT.

learning rate decay.

The TIMIT phonetic transcriptions were used for both validation and testing, while the former is for saving the model with the maximum validation R-value, and the latter is for reporting evaluation metrics by comparing predicted boundaries to the ground truth. Validation was performed after every training epoch, and testing was conducted only once after training has finished.

The reproduced results on the TIMIT test set are reported in Table 4.1. There was only a 0.13% difference in the R-value to the reported result in the paper (86.02%), so that the reproduction of results had been successful. As mention in research motivations, these results outperformed all previous methods. Particularly, it gives similar precision and recall, which suggests that the high rate of correct detection of boundaries did not come from over-segmentation, such that the model had indeed learned effectively the information to distinguish the subtle variations within a phone and the transitions between phones.

We also illustrate the workings of the baseline model with an example utterance from TIMIT, shown in Figure 4.2.

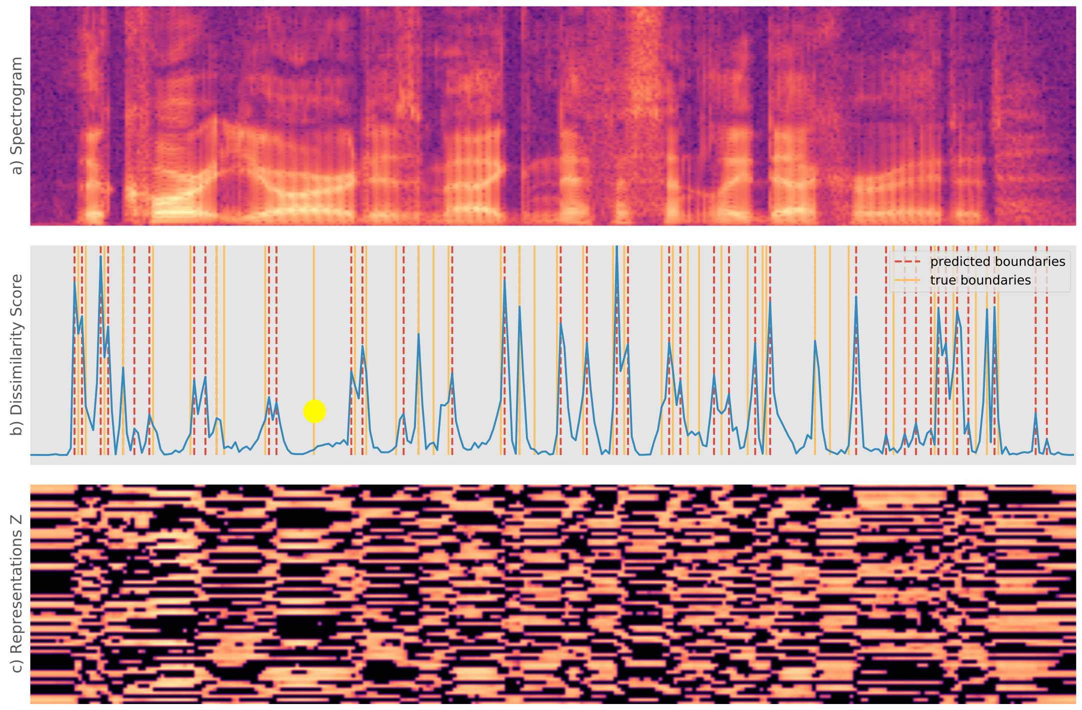

Figure 4.2: Visualisation of model outputs on the utterance *"The courtyard is magnificently decorated"*, with a) its original spectrogram; b) dissimilarity scores on adjacent output frames with true and predicted boundaries labeled; c) its output representations.

Firstly, by comparing the original sepectrogram in (a) of Figure 4.2 to the learned representations in (c), we can see much more salient displacements of the horizontal stripes in (c) at the times where phonetic transitions occur (according to the orange vertical lines that represent true boundaries in (b)). This indicates the learned representations are capable to better capture the acoustic change that signifies phonetic transitions.

The predicted (red) and true (orange) boundaries were also plotted in the plot of dissimilarity scores (b). The red dashed lines corresponds to the prominent peaks of the dissimilarities between each pair of adjacent frames, demonstrating how the location of boundaries were determined by the model. We may observe that there is a high correspondence between locations of the red and orange vertical lines, contributing to the high R-value on segmentation. However, there were also instances where the model failed to detect a boundary, such as the one spotted with a dot in Figure 4.2, which was missed by the model due to an insignificant dissimilarity value at this point. The actual transition here is from /ɑ/ to /r/ in the word *courty**ar**d*, which seems unclear as if they were a single phone to human listeners as well. Hence, to further investigate the strengths and weaknesses of the model, we conducted error analysis on model performance with respect to different phones.

4.1.1 Error Analysis

Boundaries predicted by the model on all utterances in the TIMIT test set are compared to their true locations in the phonetic transcriptions, and statistics are summarized separately on each phoneme in the 61-phone set to give: (1) percentage occurrence of the phoneme (*N*); (2) its over-segmentation rate (*OS*); (3) under-segmentation rate (*US*); (3) exact-segmentation rate (*ES*). A phonetic segment (as in the ground truth) is overly segmented if more than two boundaries are predicted within the duration of the segment with tolerance considered; it's under segmented if less than two boundaries are predicted; and it is considered to be *exactly* segmented when there are only two boundaries predicted for that segment, while both are correct (one marks the left boundary of the phone, the other marks the right boundary).

To make the analysis clearer and more meaningful, we used generic categories of the phonemes based on the work in [31], which divided phonemes by their distribution in a confusability matrix. These classes are presented in Table 4.2

Group Name	TIMIT Phone Labels	IPA Labels
Vowels	iy ih eh ae aa ah ao uh uw ux ax ax-h ix	i ɪ ɛ æ ɑ ʌ ɔ ʊ u ʉ ə ə̥ ɨ
Diphthongs	ey aw ay oy ow	eɪ aʊ aɪ ɔɪ ɔʊ
Semi-vowels	l el r w y er axr	l l̩ r w j ɝ ɚ
Stops	b d g p t k jh ch	b d g p t k dʒ tʃ
Fricatives	s sh z zh f th v dh hh hv	s ʃ z ʒ f θ v ð h ɦ
Nasals	m em n nx ng eng en	m m̩ n ɾ̃ ŋ ŋ̍ n̩
Silence	dx bcl dcl gcl pcl tcl kcl h# pau epi q	ɾ b̚ d̚ g̚ p̚ t̚ k̚ n/a n/a n/a ʔ

Table 4.2: Categorization of the TIMIT 61-phone set into generic classes based on [31].

Phones under each class were grouped together, occurrences of phones in one class were summed, and the percentage OS, US, and ES for each class were averaged over the number of distinct phones under that class. The resulting statistics are visualized in Figure 4.3, from which we make several observations on the distribution of segmentation errors.

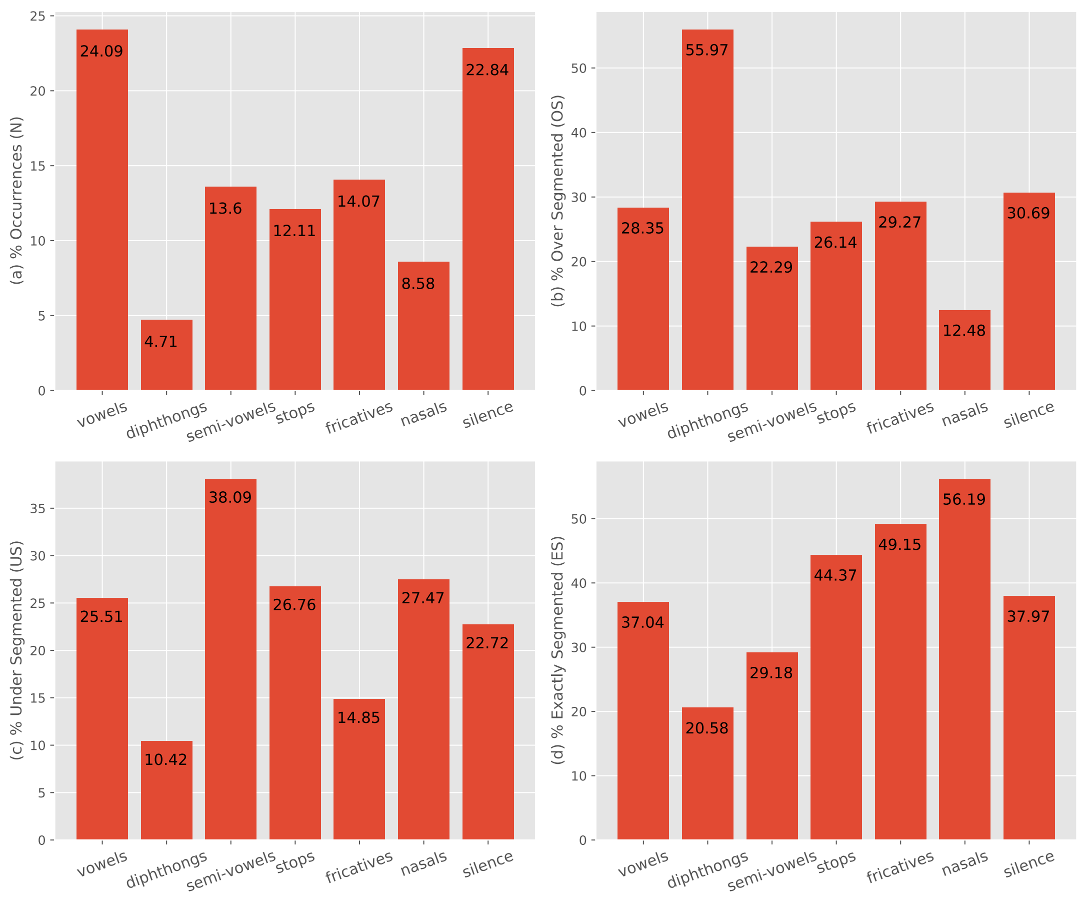

Figure 4.3: Statistics of model performance on each phone class, with (a) % occurrences in test set; (b) average % over segmentation of each phone in a particular class; (c) average % under segmentation; (d) average % exact segmentation.

In general, the TIMIT test set is not balanced across the phonetic classes, nor are the utterances in natural language that might be spoken in daily life. The variations in the distribution of phonetic classes here (plot a) might have played an important role when computing a single performance score overall. If most errors lie in a phonetic class that occurred least frequently in the data which the model is evaluated on, the errors are unlikely to impact the performance significantly. For instance, the *diphthongs* class occupied only 4.71% of the total phone count, which was the least frequent among the classes, so even though it had the highest OS error rate (as shown in b), its proportionate contribution to the total amount of erroneous boundaries predicted by the model would be relatively less significant compared to error rates of highly frequent phones (such as *vowels*).

The high occurrence frequency of *silence* here can be mainly attributed to the fact that the closure portions of stops were also categorized as ‘silence’. There were actually only three labels in the TIMIT transcriptions that marked true silences (h#, pau, and epi) which were not part of any phonetic pronunciations, which together took only 6.54% out of the 22.84% frequency of all phones categorized as *silence*.

Balancing the amounts of OS and US has been a challenging task for automatic phonetic speech segmentation systems, which requires the segmentation model to be not too sensitive to subtle acoustic change, while also being able to discover inherent transitions that are not salient from acoustic or spectral features. From plot (b) in Figure 4.3, we may observe that *diphthongs* have the highest OS rate of 55.97%, which means more than half of the diphthongs in test utterances are overly segmented. This would be expected since diphthongs are inherently dynamic in nature, such that their pronunciation starts with one vowel sound and slides towards another vowel sound, but are labeled with a single phonetic symbol. Hence, it implies that the representations learned via the CNN are not particularly good at recognizing *diphthongs* when two consecutive vowel sounds can be either represented by two *vowels* or one *diphthong*. It’s hard for the segmentation algorithm to decide when to consider two consecutive vowel sounds as a whole, and perhaps more contextual information from nearby phones would be helpful.

On the contrary, *nasals* was the least overly segmented class, which might mean that nasal phones are more stationary during the period of articulation. It was also the class that reported the highest rate of exact segmentation (56.19%). This might indicate that nasal phone segments stand out more from their adjacent phones compared to other classes, and that nasal sounds would be harder to be confused with other sounds of different manners and places of articulation.

On the other hand, the highest rate of US goes to *semi-vowels* (38.09%), possibly because semi-vowels tend to be non-syllabic, which is when the phone is not the nucleus of a syllable. For example, the /l/ in *”felt”* is non-syllabic. These phones are then less salient compared to adjacent phones, which can be a reason why they were often under-segmented compared to other phonetic classes. This might be an issue for a global peak detection algorithm, where a single prominence is applied over all frames of dissimilarity scores, such that in some cases like the beginning of a semi-vowel, an insignificant ‘peak’ may also indicate a phonetic transition. The example missed boundary we described earlier as shown in Figure 4.2 would be a case where the start of the semi-vowel /r/ is not recognized by the model as the curvature of the dissimilarity score plot at this point is relatively ‘flat’. However, though a peak detection algorithm with dynamically varying prominence might seem a solution to this problem, it is extremely hard as we have no knowledge of the phonetic class nor other phonetic information that could guide the change in prominence in the unsupervised setting, which can be a challenging task to explore further in the future.

Comparing (b) and (c) in Figure 4.3, we can notice that there are some correlation between rates of OS and US. The top three classes that gave the highest OS rates correspond to the three classes that gave the lowest US rates, which is also true vice versa. This phenomenon suggests that the model’s segmentation behavior across phonetic classes is quite consistent, where phones that are likely to be overly segmented are also less likely to be under segmented, as implied by the features of each phonetic class.

Thus, the boundaries are determined non-randomly and according to the characteristics of the phonetic transitions. It also indicates that the learned representations produced by the model contain meaningful information that leads to consistent and explainable segmentation.

The exact segmentation (ES) rate removes any potential increase in the recall of boundaries due to high OS rate, so the plot in (d) illustrates the phonetic classes that the model is the best/worst at segmenting given that no OS/US occurred. The top three classes that gave the highest ES rates are *nasals, fricatives*, and *stops*. A high ES rate means correct identification of the beginning and end of the particular phone where phonetic transitions occur. Apart from the *nasals* class discussed prior, the *fricatives* and *stops* are classes of phones that involve the creation of turbulent airflow due to constriction in the vocal folds during articulation, which may be characterized by stochastic patterns in the waveform or noise at the upper frequency range in the spectrogram. Though normally in phonology, stops have both the closure and the burst portions, they are labeled separately in the TIMIT phone set, where phones in the *stops* class here only represent the burst portion, while the closure portions are classified as *silence* (labels ending in 'cl'). This separation has made segmentation easier for these phones, since transitions from closure to burst present strong contrast in the spectral features (can be observed in the spectrogram in Figure 4.2 that voicing is mostly absent in closures). Therefore, *fricatives* and *stops* gave reasonably high ES rates, given strong acoustic contrast at transitions provided by the above mentioned characteristics. The learned representations are good at finding these characteristics, so that the higher ES rate implies greater confidence of the model to determine boundaries of phone segments. Meanwhile, it's reasonable to say that the model yields better performance on phones that offer more salient transition signals accompanied by less confusion with adjacent phones.

Overall, ES rates potentially reflect the capability of the model on the particular phone class in terms of the extent to which the learned representations understand the internal acoustic features that signify transitions for phones in that class. Therefore, we may conclude that the characteristics at transitions for *diphthongs* and *semi-vowels* are less significant to the model, while those for *nasals*, *fricatives*, and *stops* are more easily identified as the segment boundaries. These observations also infer a resemblance between the model's segmentation behavior and how easily we can obtain information from only the spectrogram visualization of the utterances, and in this sense the model is quite "intelligent". Besides, the segmentation pattern and performance on different phone classes are consistent and explainable. Yet, it's hard to design improvements based on errors from each phonetic class without knowing which phonetic class is currently under consideration during segmentation. Nevertheless, we might make improvements from the approach of learning better speech representations by altering the current network model, which could possibly induce overall improvements when learned representations capture acoustic information that's more suited to this particular task; and with these distributions of errors in hand, design changes based on the current model would become more informative.

4.2 Analysis of intermediate representations

It is hard to understand the internal workings of deep neural networks since the feature extraction process is hidden during training through the hidden layers. The reason why the application of CNN on the unsupervised phonetic speech segmentation task has improved performance significantly from conventional approaches could be further explored. Visualizing the learning procedure could be a straightforward way for understanding how the final speech representations are learned. Figure 4.4 shows the visualized feature maps from each of the five convolutional blocks, on the previously presented example utterance.[3]

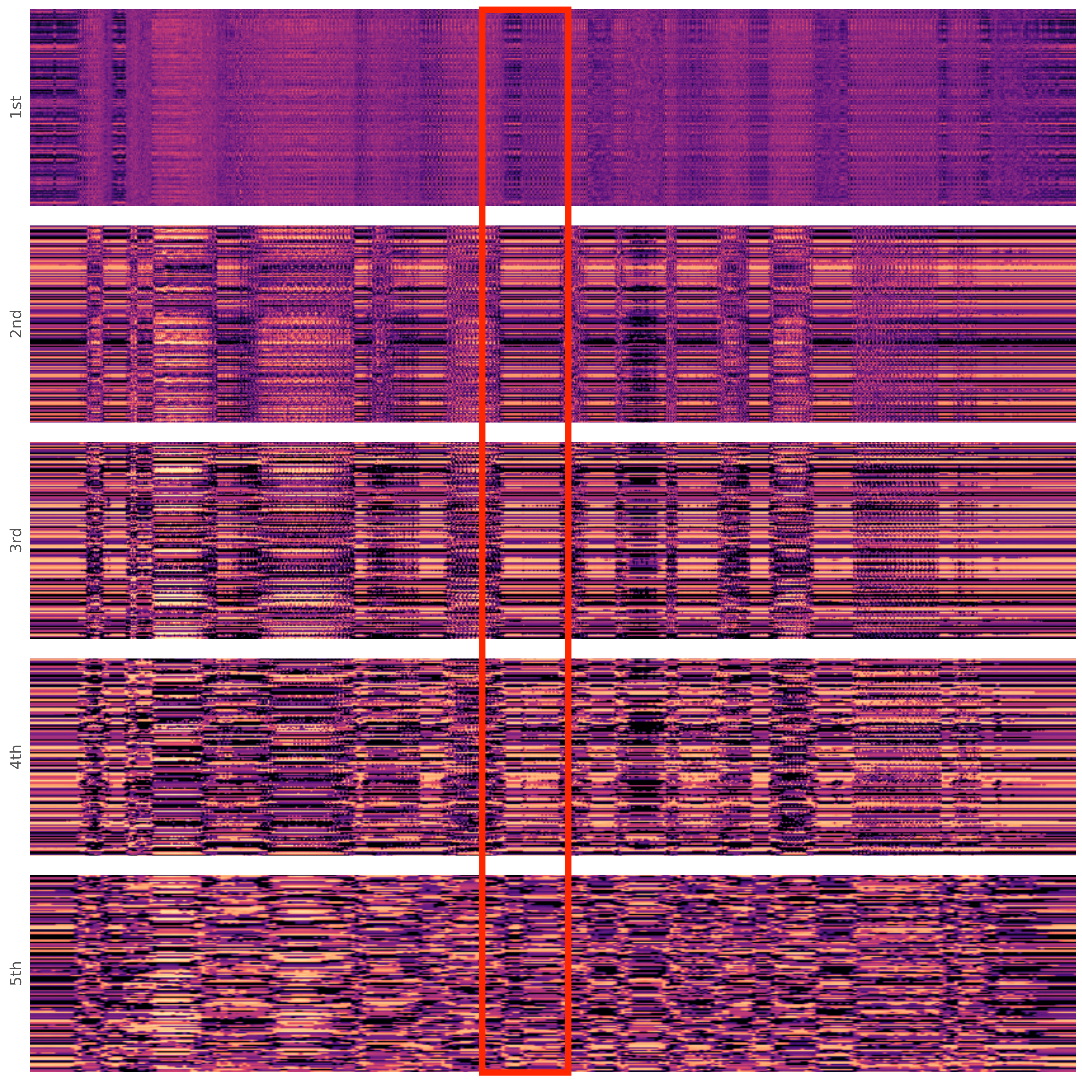

Figure 4.4: Visualisation of the feature maps from the 5 convolutional blocks in order, on the example utterance *"The courtyard is magnificently decorated"*.

We can observe "vertical stripes" from these feature maps that appear due to significant

[3]The plots were stretched to the same length for comparison, while the length of the utterance in the representation space is being reduced at each convolution layer. So, the 5th plot may seem more chaotic because it gives the highest resolution.

differences in colors of adjacent vectors, which indicate the positions of significant spectral variations. The figure demonstrates that as a raw waveform is passed through the network, the contrast in color at these vertical stripes is emphasized. Initially in the outputs of the first convolutional block, these stripes appear to be very vague, and we can notice that some areas of less 'opaque' shadings are removed in the outputs from the second block. Thus, in the first three CNN blocks, the model seems to be trying to 'clean' the extracted features to keep only those that are salient enough, which are further enhanced. From the outputs of the fourth and fifth blocks, we notice that the feature maps show a gain in the details within all potential segments, such that the number of vertical stripes increases again. As an example, such decrease and increase of detail can be illustrated by the variations inside the boxed area.

CNN block	Precision	Recall	F1	R-val
1	0.41	0.04	0.07	29.25
2	37.71	8.78	14.24	35.28
3	44.77	20.51	28.13	42.95
4	52.02	44.01	47.68	56.62

Table 4.3: Performance of evaluated on intermediate representations from each CNN block (prominence = 0.05 and tolerance = 20ms).

To demonstrate how effective the neural network is learning, we may also evaluate segmentation performance on these feature maps directly. Table 4.3 gives performance scores when segmentation is done directly on the intermediate representations from each of the 4 hidden CNN blocks. We can clearly observe that values for all the metrics were increasing as the depth convolution increased. Starting from very low precision and recall of the boundaries on outputs from the first block, layers in the network had effectively and gradually learned to achieve better segmentation performance, and the fourth layer offered the greatest increase in R-value to a score of 56.62%.

Unlike image processing, it is hard to visually understand the patterns given by each of the 256 convolution channels, but we can still compare the learning of representations via the CNN to the original spectrogram. As illustrated by Figure 4.2 (b) and (c) earlier, these vertical stripes correspond to the model's predictions of phonetic boundaries, and to a large extent also to the ground truth boundaries. Looking back at the spectrogram in Figure 4.2 (a), although it also shows the pattern of "vertical stripes" due to the voicing of phones, they only indicate a portion of the boundaries and do not align as clearly or accurately to the ground truth boundaries as the learned representations. Yet, they contain information in a different form from the feature maps given by CNN. Thus, as an experiment later in Section 4.4, we attempt to explore the effect of the participation of spectrogram-based feature engineering in the learning of representations using CNN.

4.3 Addition of Gated Recurrent Unit

On the basis of our error analysis, we have designed this experiment to investigate the effect of longer contextual information at each representation frame on the segmentation performance. Currently, the receptive field of the CNN is restricted to 30ms, which is relatively small. According to [32], the average duration of each phone segment is 80.8ms in TIMIT, which is twice as long as the current receptive field for a pair of adjacent frames of the learned representation that were considered for calculating the dissimilarity score. The amount of context considered with this receptive field size might be less than optimal for phonetic segmentation, since the transitions could also depend on pronunciations of nearby phones. Specifically in previous analysis, we found cases where context that considers a window longer than adjacent phones could be helpful for reducing OS/US. In the discussion on the *diphthong* class, we mentioned that whether to place a boundary in consecutive vowel sounds might depend on phones before or after the vowels. For instance, vowels in words such as *queuing* are not diphthongs and should be segmented in-between into two vowel segments, and knowing the context of *'ing'* could have helped in this case, to make it less probable to consider *'i'* as part of the diphthong. Experiments to incorporate a longer context into the learned speech representations were implemented considering two varied settings: the position of the recurrent layer (before/after the CNN blocks), and the direction in which context was taken (uni-directional/bi-directional). Four experiments were conducted by the combinations of these two settings. In each case, the hidden dimension of the network was kept the same, so that there were still 256 feature dimensions. The `torch.nn.GRU` module was used, and bias was set to True by default; and in the bi-directional case, the *bidirectional* argument was set to true, meaning that outputs were from two directions concatenated.

We had hypothesized that a GRU layer for capturing longer context in the time domain could have helped in this case, because hidden states in the recurrent units could learn to retain information that is important from a longer time span. However, results in Table 4.4 showed a decrease in general segmentation performance from the baseline model after GRU was applied.

Model	prominence	Precision	Recall	F1	R-val
bi-directional GRU after CNN	0.02	79.57	78.15	78.86	81.95
bi-directional GRU before CNN	0.02	82.02	80.07	81.03	83.76
uni-directional GRU after CNN	0.01	79.00	81.43	80.20	82.94
uni-directional GRU before CNN	0.02	80.95	83.57	82.24	84.67

Table 4.4: Performance of the model with the addition of a GRU layer on TIMIT.

In general, adding a GRU layer before the CNN blocks brought better performance than that after the CNN; and also using uni-directional GRU gave slightly better results than bi-directional GRU layer at respectively the same place in the network. These further disproved our hypothesis that contextual information is helpful for determining phonetic boundaries. The reason being that in theory, a bi-directional GRU would incorporate more context (both left and right context) into the learned representations, while placing the GRU layer after the CNN would have a more direct impact on the final outputs without being adjusted by the CNN. The decrease in performance due to extra context might have been caused by the collection of information from a longer time span for the current frame, which was shown to be beneficial for tasks such like phoneme recognition [42], such that undesirable distractive contextual information beyond the scope of segmentation was provided.

Thus, we might say that additional context globally added to all frames would not help with the overall segmentation performance. Still, we might investigate its effect on OS and US, since we first intended to solve the OS problem with diphthongs. We investigate this by summarizing statistics on the model (*"bi-directional GRU after CNN"*) that contained most contextual information in outputs representations.

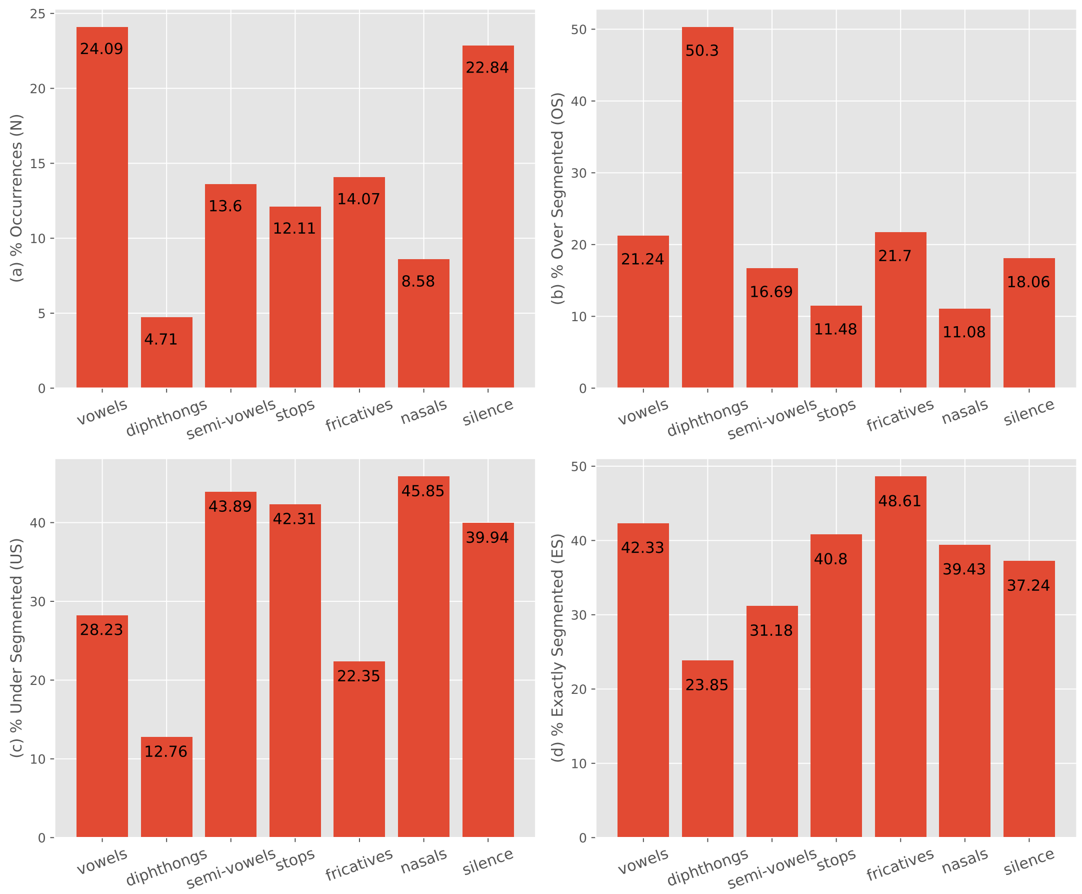

Figure 4.5: Statistics of model performance on the model *"bi-directional GRU after CNN"*.

Comparing the statistics in Figure 4.5 to those for the baseline model in Figure 4.3, the OS rates for the current model with GRU were generally lower on all phonetic classes. It has indeed shown significant improvement in terms of reduced OS as expected. However, the US rates generally increased as well, which counteracted reductions in OS rates and led to lower performance overall. This implies that a wider context considered by each frame of the speech representation might be 'blurring' the boundaries where spectral changes get more flattened out.

Potentially, in order for the overall performance of the model to increase (i.e. achieving lower OS rate while not increasing US rate), instead of adding context globally to all frames before segmentation, we might try to designate the model to first segment phones with the shorter receptive field as in the baseline, with a relatively lower prominence so that OS might be high. Then, as a tuning procedure, we can apply a learned contextual model to look at each boundary and remove those that indicate a high resemblance to special classes of transitions which are prone to OS, such as diphthongs. Nevertheless, the design of the context model would require further exploration as we are not given any information to exemplify these ‘special’ transition classes in the unsupervised setting.

Overall, our experiments imply that the actual transitions between phones depend mostly only on the adjacent phonetic sounds, and extra contextual information that captures features from phones outside the “adjacent pair” at the particular transition might be distracting for the phonetic segmentation task. On the other hand, such context could be helpful in terms of the recognition of phones, which depends not only on the spectral change at the transitions, but also on left and right phonetic context.

4.4 Combination with Manual Feature Engineering

In previous works on speech feature encoding with CNN, raw waveform inputs were used as a convention that had proven to be successful [33, 3, 4]. The engineered features that are traditionally used in speech processing just have the same purpose as CNN encoded speech representations – both are to be used as features in downstream tasks. However, few had experimented with both techniques combined together to produce speech features. It is known that with the FFT to transform waveforms into spectrograms, and with other feature engineering techniques such as further steps to extract MFCC vectors, we have control and understanding of the extraction of the information desired. Also, the previous visualization of the spectrogram on an utterance example showed salient features (such as the presence of voicing) that are helpful for humans to determine and recognize phonetic segments.

In this section, we investigate whether a combination of feature extraction techniques, with and without the neural network, would be helpful or not on the blind segmentation task. To achieve this goal, we based the learning of speech representations on extracted Mel-spectrogram features. The first convolutional block was replaced with the extraction of Mel-spectrogram features, on top of which representations were then learned through the remaining 4 layers of CNN.

In order to maintain meaningful receptive field size and the duration of audio represented by each output frame in the final representations, the window and hop lengths for obtaining the Mel-spectrogram were carefully crafted. By inversely computing frame frequency and receptive field sizes backwards down to the input layer, we chose a *hop_length=5*, and two window lengths were experimented: *win_length=400* and *win_length=800*, with the number of FFT bins equal to window length (*n_fft=win_length*). A total of 128 Mel bands were applied and produced a 128-dimensional vector at each time step. The actual implementation was done with the `librosa.feature.melspectrogram` function, such that Mel-spectrograms were obtained from the input waveforms and were then passed as inputs to the second convolutional block in the baseline model (the input channel dimension for the filter of this convolution block was set to the number of Mel bands). When *win_length=400* (25ms), the output representations Z had a receptive field of 53.4ms and a frequency of 95Hz (each frame representing 10.53ms of audio); when *win_length=800* (50ms), the receptive filed and frequency of Z were 78ms and 93Hz (10.75ms audio per frame), respectively. These two different settings aimed to explore the effects of different lengths of context per frame on segmentation performance — 53.4ms was relatively closer to that of the baseline model, and 78ms was slightly less than the average duration of TIMIT phone segments reported (80.8ms). Other settings of the neural network were left unchanged.

An extracted Mel-spectrogram with *win_length=800* is visualized in Figure 4.6 on the previously used example utterance, from which we can observe the patterns of "vertical stripes" that might have the same implications as those observed in intermediate representations given by hidden CNN layers.

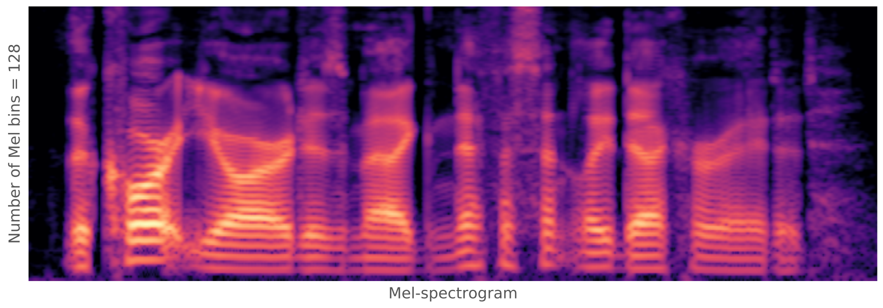

Figure 4.6: Mel-spectrogram on the example utterance *"The courtyard is magnificently decorated"*, with *n_fft=800* FFT bins and a stride of *hop_length=5*, producing *n_mels=128* Mel bands.

To better isolate the effect of CNN, segmentation performance was first evaluated directly on the extracted Mel-spectrograms, with the same peak detection algorithm. Results of experiments are reported in Table 4.5.

Method	win_length	prom.	Precision	Recall	F1	R-val
Mel-spectrogram only	400	0.03	44.29	47.05	45.63	52.42
Mel-spectrogram only	800	0.02	38.40	38.90	38.65	47.38
Mel-spectrogram with CNN	400	0.03	80.20	82.28	81.23	83.86
Mel-spectrogram with CNN	800	0.03	78.27	80.41	79.33	82.22

Table 4.5: Performance evaluated on TIMIT when Mel-spectrogram features participated in the production of speech representations; Mel-spectrograms were extracted with *hop_length=5* and *n_fft=win_length*, with *n_mels=128* Mel bands.

With either window length, the final R-value (83.86% or 82.22%) for combining Mel-spectrogram with CNN was lower than the baseline (85.89%). Yet, the reduction in R-value was not very significant, and still the better result (83.86%) outperformed previous models that didn't employ representation-learning with CNN.

In general, a window length of 400 produced better performance sores, which is the most commonly used window size (25ms) in speech processing. This implies that a smaller receptive field is more helpful on this particular task, which is similar to the discoveries in our experiments with GRUs, where longer learned contextual information did not help with the overall segmentation results.

To look at the effects of Mel-spectrogram and CNN separately, we first observed that direct segmentation on Mel-spectrogram led to significantly lower R-values compared to other models, which means extra algorithms needed to be applied to use Mel-spectrograms to conduct phonetic segmentation. As [12] stated, good representations are important to achieving great performances in any ML tasks, and in this case, Mel-spectrogram should not be considered as 'good' to use directly for this task.

On the other hand, direct segmentation on Mel-spectrogram produced significantly better results (52.42% and 47.38% R-value) than the intermediate representations from the first CNN block (29.25%), but still, the final segmentation performance after replacing that CNN block with Mel-spectrogram was lower than the CNN-only baseline model. This suggests that while each individual convolution layer may not capture enough important features in speech for recognizing phonetic transitions, the stacking of CNN layers would be much more powerful to capture those underlying features, which agrees with the rationales of deep learning. We may also reasonably claim that the extraction of Mel-spectrogram features had hidden away some details in the waveform that can be captured by the multi-layered CNN and can contribute to the determination of phonetic segment boundaries.

4.5 Performance on different datasets

In this section, the model was evaluated on different datasets. This experiment was designed because of the general lack of variation in evaluation datasets for the automatic phonetic speech segmentation task. Previously, segmentation methods for the English language were evaluated most commonly on TIMIT, or sometimes with Buckeye, which are the only two popular English corpora that provide manual phonetic transcriptions. However, due to the variations in speech data (speaker dialects, transcription quality, conditions of recording, etc.), the same model might achieve quite different results on different corpora. Besides, given the capability of self-supervised learning, training with unlabeled data should be taken advantage of. Therefore, the goal here was to investigate the impacts of changing the training/testing data on the model's segmentation performance, and to prove/disprove the robustness of the model on the different datasets evaluated.

Four different sets of experiments were performed, as listed in Table 4.6 to analyse performance from different aspects: (a) experiments with additional training data; (b) experiments of training on different datasets and testing on TIMIT; (c) experiments to evaluate model performance when trained and tested on WSJ; (d) evaluations of the baseline model (trained on TIMIT) tested on WSJ. The original model architecture was used for training and/or testing in all sets of experiments.

4.5.1 Additional Training Data with LS (a)

In the work by Kreuk *et al.* [22], expanded training data was also experimented on the original model, which reported an increase of 0.38% in the R-value evaluated on TIMIT, when training was done with additional LS train-clean-100 dataset alongside TIMIT.

Training Data	Validation	Testing	prom.	P	R	F1	R-val
TIMIT (baseline)	TIMIT	TIMIT	0.05	83.34	83.60	83.47	85.89
experiment (a)							
TIMIT + LS dev-other	TIMIT	TIMIT	0.06	84.20	84.83	84.52	86.78
TIMIT + LS dev-clean	TIMIT	TIMIT	0.05	84.71	84.84	84.77	87.00
experiment (b)							
LS train-clean-100	TIMIT	TIMIT	0.06	84.08	84.61	84.34	86.64
WSJ SI-284	TIMIT	TIMIT	0.07	80.46	81.24	79.69	83.29
experiment (c)							
TIMIT	TIMIT-FA	TIMIT-FA	0.06	75.73	74.81	75.26	78.91
WSJ SI-284	dev93	eval92	0.05	69.97	71.66	70.81	74.91
experiment (d)							
TIMIT	TIMIT	TIMIT-FA	0.05	74.72	71.64	73.15	77.10
TIMIT	TIMIT	dev93	0.05	71.27	62.49	66.59	71.35
TIMIT	TIMIT	eval92	0.05	71.15	70.24	70.70	75.04

Table 4.6: Performances of the model trained and/or tested on different datasets (% is omitted, and *FA* means the forced aligned transcriptions were used for TIMIT).

This experiment setting aims to explore further the idea that training on additional data would improve model performance, in terms of the quality of the added data. Since LS has provided subsets of data in the 'clean' and 'other' categories separately based on the quality of speech and the simplicity of recognizing the utterances as in ASR, training on data in these two categories might reveal the extent to which each could help with segmentation on the clean read speech in TIMIT.

Due to limited computing power, we added the smaller development sets (dev-other and dev-clean) instead of the large LS train sets for training. From results in section (a) of Table 4.6, we see that the R-value has increased by approximately 1% for both experiments with dev-other or dev-clean. In particular, the performance when dev-clean was added to training is slightly better than adding dev-other, with a difference of 0.22% R-value. This gap was expected since we were testing on TIMIT, which is a very clean dataset, so training also on relatively cleaner data would likely yield better testing results. In general, our results also proved the effectiveness of the addition of training data, which might suggest a positive relationship between the amount of training data and the model's segmentation performance.

4.5.2 Training on other Datasets, Evaluation on TIMIT (b)

In this experiment, we try to analyse the generalization ability of the model by training on LS train-clean-100 and WSJ SI-284, respectively, and then testing on TIMIT. We also use TIMIT as the validation set, because validation (i.e. for saving the best checkpoint and finding the best prominence value) requires labels of phonetic boundaries, but LS and WSJ did not have manual labels at the phone level.

Training with unlabeled LS and WSJ training sets showed adequate performances on

TIMIT, achieving 86.64% and 83.28% R-value, respectively (Section (b) of Table 4.6). Using LS train-clean-100 for training even offered a performance gain (0.75%) on TIMIT, in the case where training and testing were done on different corpus, compared to the baseline model where training and testing were both done on TIMIT only. On the other hand, though the result on TIMIT when training with WSJ did not outperform the baseline, there was only a 2.6% gap, and still an R-value of 83.29% outperformed all previous models on the task from this one. Thus, the model can generalize well on testing data that was not used for training.

The results give meaningful implications on the usage of the model in practice, where model's generalization ability is important, such that large amounts of readily available unlabeled data can be used directly to train the model, while adequate segmentation performance can still be achieved on other data that doesn't have enough audio to form a stand-alone training set.

4.5.3 Model Performance on WSJ (c)

In the previous experiments, the model was still validated and tested on TIMIT, due to the lack of phonetic transcriptions for the LS and WSJ corpora. Therefore, we now attempt to assess the model completely on the WSJ corpus by obtaining transcriptions at the phone level via forced alignment. An LDA-MLLT-SAT triphone acoustic model was trained on the WSJ SI-284 training set using Kaldi, and phonetic transcriptions on dev93 and eval92 were attained by aligning the respective data to corresponding orthographic transcriptions based on this model. In addition, in order to make a fair comparison, the baseline model was trained again on the similarly forced-aligned TIMIT data for validation and testing ('TIMIT-FA'). Results are presented in section (c) of Table 4.6.

The segmentation performance was significantly reduced to 74.91% R-value when WSJ was also used as the validation and testing set. On one hand, forced-aligned phonetic boundaries — based on a trained model that achieved around 11% WER on dev93 — may not be as accurate as manually labeled boundaries. This phenomenon was validated by the drop in R-value by 6.98% from the baseline result when we use transcriptions from forced-alignment instead of the manual labels for TIMIT. On the other hand, the WSJ corpus was shown to be generically harder than TIMIT, so a reasonable amount of performance decrease was expected. Nevertheless, acceptable segmentation performance on WSJ suggests that the current model architecture can be effectively applied and produce relatively consistent segmentation behavior across different datasets.

4.5.4 Baseline Model Evaluated on WSJ (d)

In addition, we also report the performance of the baseline model when not tested on TIMIT, to show that if without any additional training, the baseline model can be directly taken to segment other speech data. Firstly, looking at the test R-value when TIMIT test transcriptions were obtained via forced-alignment, we notice that it has significantly dropped to 77.1%. Hence, we may say that the actual performances on the WSJ datasets are likely to be higher than the reported values due to extra errors from force-alignment. Secondly, when tested on the dev93 and eval92 datasets of WSJ, we obtained similar R-values compared to that on the forced-aligned TIMIT test set (respectively a difference of -5.75% and -2.06% in the R-value). Overall, we can reach the conclusion that the training of the baseline model with TIMIT was effective, and it can also perform adequately well on the WSJ corpus.

Besides, since the model trained on TIMIT led to generally lower test performances on WSJ, we may again say that WSJ is a more challenging dataset on the phonetic segmentation task compared to TIMIT.

In practice, when we want to apply the model to segment large amounts of unlabeled speech (with or without training the model with this data), we could sample a very small portion of the target data to create reliable transcriptions and evaluate it on the TIMIT-trained baseline model; so that by comparing performance scores to the baseline result on TIMIT, we would get a general expectation on how well the target data could be segmented.

Furthermore, because of the self-supervised mechanism of the neural network model, adding unlabeled data to training is almost effortless. It means whenever we have good quality audios, we may add them to training the model, and the resulting performance on a clean test set would likely be improved.

In the future, we could obtain more accurate phonetic transcriptions for datasets that possess different traits, especially those that are noisier than TIMIT, to evaluate the usage of the model in real-life speech conditions, and hence, to better fulfill the purposes of phonetic segmentation in the unsupervised setting.

Chapter 5

Conclusion and Future Work

The aim of this project was to investigate closely the first CNN-based self-supervised representation learning model ([22]) for the task of unsupervised phonetic speech segmentation, for answering the primary research question as to “why” such an approach works best on this particular task. For fulfilling this aim, experiments were conducted based on various aspects of this CNN model, and in-depth analysis were carried out with respect to the mechanisms of and the challenges on phonetic speech segmentation.

We verified and demonstrated the effectiveness of the original model by visualizing the final output representations and intermediate feature maps, from which significant displacements, or visually the vertical stripes, were observed that corresponded with locations of predicted phone boundaries. Intermediate representations produced by hidden convolutional blocks deeper into the network yielded better segmentation performance, demonstrating the build-up of useful information on phonetic transitions at each convolution layer.

The efficacy of speech feature extraction through CNN layers was further investigated by designing and experimenting with a network that combined manual feature engineering with CNN representation learning. Segmentation results showed that the replacement of one CNN layer with the extraction of Mel-spectrogram features reduced performance by approximately 2% in R-value, but the achieved 83.86% R-value still outperformed previous blind segmentation models. It suggested that the convolution operation with learned kernel weights was more effective than manual feature engineering operations, for discovering acoustic change information at the phone level. On the other hand, it confirmed that the given CNN was more effective when applied directly on raw waveform inputs.

The impacts of context on determining phonetic boundaries were investigated by adding a GRU layer to the original network. The overall lowered performances on models with GRU indicated that longer contextual information was not helpful in general for segmentation at the phone level, and that the local receptive field of 30ms given by the convolution mechanism was proved to be more suitable for this task. We also discovered that adding context to each output unit globally with GRU had significantly decreased OS rate, but at the same time increased US rate.

This model was also able to generalize well across different datasets by achieving adequate performances when the training and testing sets were different, which reflected that filters learned by the CNN were independent of the characteristics of the particular training data. The model worked best with clean corpora, with generally better performances on TIMIT, and a maximum of 87% R-value were achieved when the high quality *dev-clean* dataset from LS were added to the training of the model alongside TIMIT data.

Even though this model was trained with unsupervised data, validation and testing were still done with the help of phonetic transcriptions. Critically, no phonetic transcription were provided at the first place in the definition of blind segmentation. Therefore, we may also try in the future to make the training stage truly unsupervised with unsupervised validation, potentially by re-formulating the loss so that it has a stronger correlation with the R-value metric.

The importance of this project lie in its implications on the future developments of unsupervised phonetic speech segmentation systems. The shift from using conventional engineered speech features to learned speech representations via neural networks had induced significant performance gain. Such learned representations are desired, as they have been experimentally proved to be more powerful and intellectual. As for numerous other intellectual tasks, the application of deep learning would likely become the future focus on the development of phonetic speech segmentation models as well, and the model investigated in this project laid the foundations for the shift towards such approaches at the very starting stage.

Indeed, the new state-of-the-art model proposed in 2021 on this task was an extension based on this model, with the addition of a segment level encoder, and achieved around 1% improvement in the R-value [5]. Moreover, another model with the addition of quantization to the speech representations based upon the architecture of this model was developed also last year, though it achieved a lower performance [21]. These attempts further illustrated the significance of understanding the strengths and weaknesses of this model, for introducing meaningful improvements.

We had revealed the model's performance on each generic phonetic class, and analyzed the phonetic classes that the model generally performed well and those that required improvements to reduce OS or US rates, such as on *diphthongs* and *semi-vowels*, respectively. We had seen the effects of longer contextual information on OS and US in our experiments, from which we propose that perhaps a separate tuning algorithm on top of the ordinary segmentation outputs to further adjust the number of predicted boundaries based on local features could be helpful and can be implemented in the future. Likewise, other critical design decisions can be made more informatively in the future given the current weaknesses, and our next step would be to exploit these weaknesses and seek for improvements by modifying the current model architecture.

Meanwhile, [22] also showed that the model trained with TIMIT with additional 100 hours of clean data from LS achieved good performances on Hebrew and German. As a next step, we may further investigate the model's ability to transfer well across other non-Semitic or non-Germanic languages to better develop the model for use on under-resourced languages.

Bibliography

[1] Yossi Adi, Joseph Keshet, Emily Cibelli, and Matthew Goldrick. Sequence segmentation using joint rnn and structured prediction models. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2422–2426. IEEE, 2017.

[2] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. Effectiveness of self-supervised pre-training for speech recognition. *arXiv preprint arXiv:1911.03912*, 2019.

[3] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019.

[4] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[5] Saurabhchand Bhati, Jesús Villalba, Piotr Żelasko, Laureano Moro-Velazquez, and Najim Dehak. Segmental contrastive predictive coding for unsupervised word segmentation. *arXiv preprint arXiv:2106.02170*, 2021.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Yago Pereiro Estevan, Vincent Wan, and Odette Scharenborg. Finding maximum margin segments in speech. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–937. IEEE, 2007.

[8] William Falcon et al. *Pytorch-Lightning Documentation, Release 0.7.6.*

[9] Sadaoki Furui. *Digital speech processing, synthesis, and recognition.* CRC Press, 2018.

[10] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.

[11] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

[14] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[15] Dac-Thang Hoang and Hsiao-Chuan Wang. Blind phone segmentation based on spectral change detection using legendre polynomial approximation. *The Journal of the Acoustical Society of America*, 137(2):797–805, 2015.

[16] Takaaki Hori, Jaejin Cho, and Shinji Watanabe. End-to-end speech recognition with word-based rnn language models. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 389–396, 2018.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[18] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[19] Biing H Juang and Tsuhan Chen. The past, present, and future of speech processing. *IEEE signal processing magazine*, 15(3):24–48, 1998.

[20] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009.

[21] Herman Kamper and Benjamin van Niekerk. Towards unsupervised phone and word segmentation using self-supervised vector-quantized neural networks. *arXiv preprint arXiv:2012.07551*, 2020.

[22] Felix Kreuk, Joseph Keshet, and Yossi Adi. Self-supervised contrastive learning for unsupervised phoneme segmentation. *arXiv preprint arXiv:2007.13465*, 2020.

[23] Felix Kreuk, Yaniv Sheena, Joseph Keshet, and Yossi Adi. Phoneme boundary detection using learnable segmental features. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8089–8093. IEEE, 2020.

[24] Victor Kuperman, Mark Pluymaekers, Mirjam Ernestus, and Harald Baayen. Morphological predictability and acoustic duration of interfixes in dutch compounds. *The Journal of the Acoustical Society of America*, 121(4):2261–2271, 2007.

[25] Paul Michel, Okko Räsänen, Roland Thiolliere, and Emmanuel Dupoux. Blind phoneme segmentation with temporal prediction errors. *arXiv preprint arXiv:1608.00508*, 2016.

[26] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[27] Douglas B Paul and Janet Baker. The design for the wall street journal-based csr corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[28] 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.

[29] Okko Johannes Räsänen, Unto Kalervo Laine, and Toomas Altosaar. An improved speech segmentation quality measure: the r-value. In *Tenth Annual Conference of the International Speech Communication Association*. Citeseer, 2009.

[30] Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE, 2020.

[31] Patricia Scanlon, Daniel PW Ellis, and Richard B Reilly. Using broad phonetic group experts for improved speech recognition. *IEEE transactions on audio, speech, and language processing*, 15(3):803–812, 2007.

[32] Odette Scharenborg, Vincent Wan, and Mirjam Ernestus. Unsupervised speech segmentation: An analysis of the hypothesized phone boundaries. *The Journal of the Acoustical Society of America*, 127(2):1084–1095, 2010.

[33] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[34] Manish Sharma and Richard Mammone. " blind" speech segmentation: automatic segmentation of speech without linguistic knowledge. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 2, pages 1237–1240. IEEE, 1996.

[35] Jorge Sola and Joaquin Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, 44(3):1464–1468, 1997.

[36] Adriana Stan, Cassia Valentini-Botinhao, Bogdan Orza, and Mircea Giurgiu. Blind speech segmentation using spectrogram image-based features and mel cepstral coefficients. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 597–602. IEEE, 2016.

[37] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with

contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[38] Jan P van Hemert. Automatic segmentation of speech. *IEEE Transactions on Signal Processing*, 39(4):1008–1012, 1991.

[39] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[40] Yu-Hsuan Wang, Cheng-Tao Chung, and Hung-yi Lee. Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries. *arXiv preprint arXiv:1703.07588*, 2017.

[41] M-B Wesenick and Andreas Kipp. Estimating the quality of phonetic transcriptions and segmentations of speech signals. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 1, pages 129–132. IEEE, 1996.

[42] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*, 2021.

[43] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. Visualizing and comparing convolutional neural networks. *arXiv preprint arXiv:1412.6631*, 2014.

Appendix A

TIMIT phone set

TIMIT label	IPA label	TIMIT label	IPA label
iy	i	ih	ɪ
eh	ɛ	ey	eɪ
ae	æ	aa	ɑ
aw	aʊ	ay	aɪ
ah	ʌ	ao	ɔ
oy	ɔɪ	ow	ɔʊ
uh	ʊ	uw	u
ux	ʉ	er	ɝ
ax	ə	ix	ɨ
axr	ɚ	ax-h	ə̥
jh	dʒ	ch	tʃ
b	b	d	d
g	ɡ	p	p
t	t	k	k
dx	ɾ	q	ʔ
s	s	sh	ʃ
z	z	zh	ʒ
f	f	th	θ
v	v	dh	ð
m	m	n	n
ng	ŋ	em	m̩
nx	ɾ̃	en	n̩
eng	ŋ̍	l	l
r	r	w	w
y	j	hh	h
hv	ɦ	el	l̩
bcl	b̚	dcl	d̚
gcl	ɡ̚	pcl	p̚
tcl	t̚	kcl	k̚

Table A.1: Mapping of TIMIT phone labels to IPA symbols, excluding silence labels (h#, pau, epi).