The Thesis Committee for Luke Vincent Strgar
certifies that this is the approved version of the following thesis:

Phoneme Segmentation Using Self-Supervised Speech Models

Committee:

David Harwath, Supervisor

Greg Durret

Phoneme Segmentation Using Self-Supervised Speech Models

by

Luke Vincent Strgar

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

The University of Texas at Austin

August 2023

Dedication

For all their unconditional love, support, and inspiration, this work is dedicated to my parents, Wendy and Franc, my sister, Emma, and my brother, Ian.

Acknowledgments

To my parents - Wendy and Franc - I feel both deeply grateful and incredibly lucky to have your continuous encouragement. For all the times when you reminded me to keep trusting and believing in myself, thank you. To my sister Emma, thank you for always making me smile and being a role model for doing what inspires you. Finally, I would like to express my deepest thanks to my supervisor at the University of Texas at Austin, David Harwath. I have thoroughly enjoyed working with and learning from you, and I remain very grateful for your kindness and support from the beginning to the end of the research process.

Abstract

Phoneme Segmentation Using Self-Supervised Speech Models

Publication No. _____

Luke Vincent Strgar, M.S.Comp. Sci.
The University of Texas at Austin, 2023

Supervisor: David Harwath

We apply transfer learning to the task of phoneme segmentation and demonstrate the utility of representations learned in self-supervised pre-training for the task. Our model extends transformer-style encoders with strategically placed convolutions that manipulate features learned in pre-training. Using the TIMIT and Buckeye corpora we train and test the model in the supervised and unsupervised settings. The latter case is accomplished by furnishing a noisy label-set with the predictions of a separate model, it having been trained in an unsupervised fashion. Results indicate our model eclipses previous state-of-the-art performance in both settings and on both datasets. Finally, following observations during published code review and attempts to reproduce past

segmentation results, we find a need to disambiguate the definition and implementation of widely-used evaluation metrics. We resolve this ambiguity by delineating two distinct evaluation schemes and describing their nuances. We provide a publicly available implementation of our work on Github [1].

[1] https://github.com/lstrgar/self-supervised-phone-segmentation

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Phoneme Boundary Detection

Phoneme boundary detection, also referred to as phoneme segmentation, involves labeling the temporal boundaries between discrete phonemic units in a speech signal. Previously, phoneme segmentation has been studied and benchmarked in the supervised [7, 13, 20, 23] and unsupervised settings [3, 12]. In the former case, models are allowed to leverage a ground truth reference segmentation - a vector of phoneme onset, offset times - during training. In the latter case, the model only sees the input speech signal and is thus tasked with producing a segmentation by relying on the statistics of the underlying data alone. A third setting, known as forced-alignment or text-dependent phoneme segmentation, extends the supervised case by adding a temporally ordered list of phonetic identities to the model input. Conditioning on categorical phonetic identity means that model performance in the forced alignment setting typically supersedes text-independent supervised phoneme segmentation, which supersedes unsupervised predictions. In this body of research, we focus on and report results for the unsupervised and text-independent supervised cases.

The phonetic boundary detection task and our research thereof can be broadly contextualized and made significant within two threads of ongoing

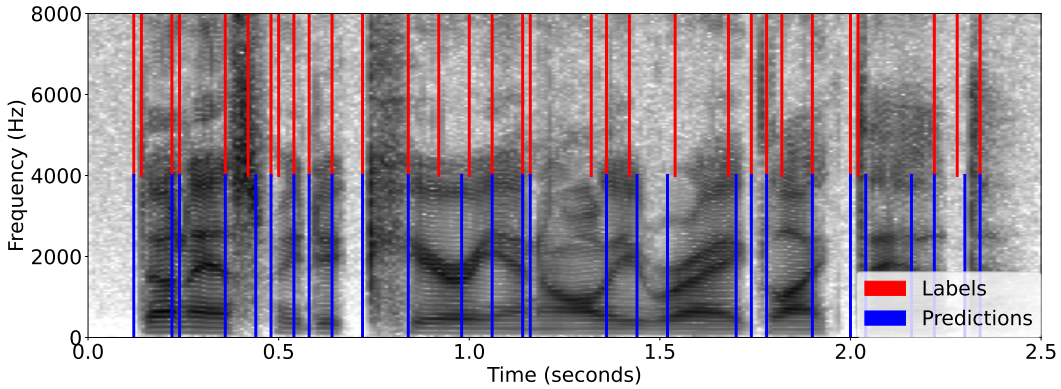

Figure 1.1: Example spectrogram with ground truth and supervised *readout*-mode model predicted boundaries.

work in automatic speech processing and linguistic science. The first of these is language documentation and related tools for automated or computer-assisted linguistic annotation. The second relates to the development of strategies to identify discrete speech units for the purposes of tokenizing raw speech audio. We introduce the topics and discuss their relevance to our work below.

1.2 Language Documentation & Linguistic Annotation

Language documentation and annotation relates to a set of activities and techniques for creating lasting and thorough records of distinct human speech communities, which describe groups of people with some degree of shared community structure and shared linguistic practices [28]. There are several motivations for modern work in language documentation including but not limited to language endangerment, continuity of research efforts, and ver-

ifiability of analyses [10]. Language endangerment is a particularly pressing challenge: a staggering 40% of the worlds languages are in danger of going extinct as their population of speakers shrinks [25]. Closely related are low resource languages (LRLs), which, among other possible characteristics, have received less research attention and have relatively fewer corpora, both digitized or not and annotated or not [22].

By providing researchers more leverage in processing data and conducting analyses, automated tools can accelerate the documentation and annotation process [1]. This is all the more essential for endangered languages where time to collect and process data may be a resource in short supply. Incidentally, modern techniques in automatic speech processing such as self-supervised pre-training (Sections 1.4, 2.3) also render the development of automated tools for low resource languages (read: low data regimes) in some cases tractable. Irrespective of a language's status as endangered or low resource, automated tools can also increase the intelligibility and reproducibility of research efforts by streamlining data and record formatting and reducing opportunities for human error.

Among the various tasks involved in language documentation, transcription of speech recordings is essential and among the most time intensive [1]. It has been thought that approximately 1000 hours of speech should be recorded and processed to facilitate a sufficiently thorough study of an endangered language [19]. In addition, others have estimated that for every minute of recorded speech an hour of work is required for a trained linguist to

perform phonetic transcription [24]. Our work does not go so far as to propose a system for automatic phonetic transcription; however, phonetic segmentation could be considered a pre-processing step in systems designed to augment human workers. For example, trained specialists could examine, assert, and correct automatically generated phonetic boundaries superimposed on spectral representations or audio recordings. In addition, the methods we develop and present here are applicable to low-resource settings wherein little or no manually segmented data is required to obtain highly accurate results. Today there exist a number of computational linguistic analysis tools to which an implementation of our method would be a natural extension [4, 36, 39].

1.3 Discrete Speech Unit Discovery

Effective discovery of discrete speech units from raw audio enable the translation of pseudo-continuous digital speech data into a tokenized representation. Researchers in the automatic speech processing field will be aware of the broad successes of pre-trained language models (PLMs) in the domain of natural language processing (NLP) [30, 41]. PLMs, when having a sufficiently large paramter space, are often referred to as large language models (LLMs), and recently the speech field has demonstrated interest in applying PLM/LLM inspired techniques to digital speech recordings or some derivative thereof. Critically, PLMs and LLMs are designed to operate over tokenized data. In NLP, there are well-known and time-tested techniques for converting unstructured textual data into a set of discrete tokens with a tractable

cardinality. Though digital speech data is technically a discretized representation in computer memory, the use of floating point values combined with high sampling rates renders this data far too high-dimensional to be easily tokenized. Thus, solving the problem of discrete speech unit discovery (an effective tokenization) is a precursor to the successful application of many language-modeling methodologies to digital speech data.

Historically, many speech processing systems - for a variety of different tasks - have relied on an intermediate textual representation, which is easily tokenized. However, the process of transcription, whether automatic or not, necessarily results in a loss of information contained in the speaker's prosody and tone. One may expect then that a tokenized representation of raw speech audio could be more expressive and useful in a variety of speech processing tasks than their text-based counterpart. This idea has led to the advent of a new suite of methods sometimes referred to as "textless" speech processing where tokenized speech is used as a simple drop-in replacement for text in applications of PLMs and LLMs [15,17,21]. So far, effective approaches to the discrete speech unit discovery problem have been primarily based on derivatives of data representations learned via a self-supervised objective [2,11,26,37].

Since spoken languages are considered to be composed of a discrete set of phonetic units, a phonetic segmentation may prove useful for downstream tasks requiring a tokenized input. That is, a phonetic segmentation could provide a natural candidate representation over which PLM/LLM techniques can be applied for a variety of downstream speech processing and generation

tasks. In our work we acknowledge that effective discrete unit discovery in speech data - whether directly related to phonetic units or not - has and will likely continue to advance the abilities of modern speech processing technologies. Proposing to use well-defined and well-understood phonetic categories as the basis of such a discrete representation implicitly nods to the history and rigor of classical linguistic science. Incidentally, for some arbitrary input speech, using phoneme-based units to represent the data may enable greater explainability and interpretability of PLM/LLM behavior, which is notoriously opaque.

1.4 Self-Supervised Pre-Training

Closely related to the aforementioned approaches for discrete speech unit discovery [2, 11, 26, 37], the models and methodologies put forth in this work are motivated by the broad successes of self-supervised learning techniques. Self-supervised learning is a subclass of unsupervised learning in which supervised training targets are derived from the input data itself circumventing the need for manual data annotation. In Section 2.3 we discuss self-supervised pre-training in greater depth.

This work explores the utility of self-supervised representation learning in the phoneme segmentation task. Specifically, we utilize pre-trained model checkpoints for two well-known and widely used self-supervised speech models, wav2vec2.0 [2] and HuBERT [11], and apply different strategies to refine these models' frame-wise representations for phoneme segmentation. In one case,

we freeze the model's weights and extend its architecture with strategically placed, trainable, convolutional probe layers to manipulate and synthesize hierarchical features. In a separate case, we append a simple projection layer to the pre-trained model's encoder and train the projection layer as well as all model weights end-to-end. Both models output a binary predictor for each frame corresponding to the presence of a boundary.

1.5 Experimental Setup & Results

We evaluate and report results using the TIMIT [8] and Buckeye [33] speech corpora and find our model eclipses previous state-of-the-art performance on both datasets in the supervised and unsupervised settings. Unsupervised training is accomplished by furnishing a noisy label-set with the predictions of a separate model [12] that was trained in an unsupervised fashion using contrastive predictive coding [29] on a next frame prediction task. In supervised training, we also explore the label efficiency of our approach by sweeping over the amount of labeled training data used and find that the model surpasses previous SotA performance with as little as 10% of the training set. In summary, we demonstrate a successful application of self-supervised pre-training to the phoneme boundary detection task and offer a new SotA benchmark in the unsupervised and text-independent supervised settings.

1.6 Thesis Structure & Overview

The organization of this work is as follows:

- In Chapter 2 we rigorously define the phoneme boundary detection task and describe details of previous approaches to solving the aforementioned. Later, we introduce background information related to self-supervised pre-training. Specifically, the techniques used to train models like Wav2Vec2.0 [2] and HuBERT [11], which our work builds on.

- In Chapter 3 we detail two neural-model architectures we employ and describe the optimization framework used to train these models.

- In Chapter 4 we describe the challenges with accurate evaluation of the phoneme boundary detection task. We conclude this section by defining two distinct evaluation schemes with the aim of reducing ambiguity for the research community moving forward.

- In Chapter 5 we detail how we have pre-processed the TIMIT and Buckeye corpora to encourage both reproducibility of our results and maintenance of standard benchmarks. Later we discuss the details of our various experiments and report our key results.

- In Chapter 6 we emphasize the primary conclusions and takeaways of our research and describe several promising opportunities for future work.

Chapter 2: Background and Related Work

In this chapter we begin by providing a mathematically rigorous definition of the phonetic segmentation task. Later, we offer a short history and summary of past work on supervised and unsupervised phonetic segmentation models. Finally, we discuss self-supervised learning in speech processing and detail two self-supervised pre-trained models, which our methodology builds on.

2.1 Formalizing Phonetic Segmentation

In the following three subsections we formalize the phonetic segmentation problem with clear mathematical notation defining model inputs, outputs, and reference output.

2.1.1 Input Data & Preprocessing

In phoneme segmentation the input is a raw speech waveform $x \in \mathcal{X}$ represented as $x = (x_0, x_1, ..., x_N)$ where each x_i is a single floating point value representing relative pressure in the transmission medium. Typically, x will be pre-processed and temporally down-sampled by some transformation $f_x : \mathcal{X} \rightarrow \mathcal{Z}$ to produce $f_x(x) = z = (z_1, z_2, ..., z_T)$ where $T << N$ and $z_i \in \mathbb{R}^d$. Here, $z \in \mathbb{R}^{Txd}$ can be thought of as representing a series of acoustic

feature frames and T now encodes the temporal resolution we desire to make predictions with.

2.1.2 Target Output

Each input speech sample is paired with a label sequence of time stamps $y = (y_1, y_2, ..., y_K)$ where each y_i indicates the presence of one boundary and is represented as a single floating point value encoding the time units relative to the beginning of the utterance. Similar to the down-sampling of x, one might choose to bin y such that each element is converted to units of acoustic feature frames. We call this representation $\bar{y} = (\bar{y}_1, ..., \bar{y}_K)$ and note that K, N, and T may vary across input, label pairs.

2.1.3 Predicted Output & Evaluation

Automatic phoneme segmentation thus asks for a prediction, $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_{\hat{K}})$ that closely matches the ground truth label $\bar{y}$. Classically, the closeness of a reference and predicted segmentation is evaluated with the precision, recall, F1, and R-Value metrics [35]. Section 5 describes these quantities as well as their nuances in detail.

2.2 History of Phoneme Boundary Detection

The phoneme boundary detection has been explored using a variety of different model types and under various levels of supervision. Below we describe a curated selection of past approaches to solving this problem, which

we believe captures the most relevant variety of work. Specifically, we discuss models trained with a supervised objective both with (text-dependent) and without (text-independent) symbolic textual conditioning. We also discuss models using a purely unsupervised objective. Of specific interest is an approach based on the noise-contrastive estimation principle [12], which our work in unsupervised phoneme segmentation directly leverages. None of the unsupervised models we review condition on a symbolic textual input.

2.2.1 Supervised, Text-Independent

In the text-independent, supervised setting, the most relevant work revolves around the usage of recurrent neural network models (RNNs). RNNs have been both used as binary predictors [7] and feature learners for a subsequent structured prediction task [13].

In the former case [7], a bi-directional LSTM-based RNN model is proposed that, for each frame, outputs a boundary confidence score and is optimized using a weighted binary cross entropy objective. The sequence of confidence scores is processed by a peak-picking algorithm and a variable threshold is applied to the discovered peaks during boundary calling. Frame-wise inputs to this model are 26 dimensional feature vectors containing 12 Mel-Frequency Cepstrum Coefficients, the log energy, and a first derivative approximation.

More recently, bi-directional LSTMs have been used to learn features for use in an energy based model [13]. Similar to [7], frame-wise inputs consist of a vector containing 13 mel-frequency cepstrum coefficients (MFFCs) along with

delta and delta-delta features. The feature vector at time j also includes four features corresponding to the euclidean distance between MFCCs at time $j-t$ and $j+t$ for $t \in 1, 2, 3, 4$. Feature representations at each frame are extracted from the RNN and used to construct secondary features by summing all frame-wise outputs that fall between boundaries. Finally, the original and secondary features are summed to form a feature vector with which the objective is computed. The objective itself is a derivative of the hinge loss, effectively learning a function that maps "good" segmentations to higher values, and the actual predicted segmentation is simply defined as the argmax.

2.2.2 Supervised, Text-Dependent

Also known as forced alignment, text-dependent phoneme segmentation provides a boundary predictor with both audio input and a corresponding orthographic transcription as input. In this setting, probabilistic models such as HMMs have been applied [23], and recently a multitask learning framework was proposed using pre-trained model features was proposed [20].

Among the most well-known probabilistic models for forced alignment is the Montreal Forced Aligner (MFA) [23], which utilizes a joint Gaussian mixture model (GMM) and hidden Markov model (HMM). The GMM is used to learn a probabilistic model from a set of hand-designed acoustic input features to the phonetic category, while the HMM learns sequential information in the form of transition probabilities between phonetic classes. Input features include 13 MFCCs along with delta and delta-delta features resulting in a

39-dimensional feature vector.

Neural models have also been applied to solve the forced alignment problem. One such high-performing method uses a cross attention mechanism to synthesize acoustic and categorical (phoneme class) features [20]. Here, a pre-trained model is used as an acoustic encoder, and numeric vector representations of phoneme classes are learned separately. Also, information related to the relative placement of a phonetic unit within a word is also provided as a positional encoding. Finally, these features are synthesized and used to model a distribution over phonetic class at each audio frame.

2.2.3 Unsupervised

In the unsupervised setting, signal processing based approaches were initially dominant [6, 9], but recent research has focused on learning-based methods [3, 12, 16]. For example, [16] proposed a nonparametric Bayesian approach wherein unsupervised phonetic segmentation, unit discovery through clustering, and generative modeling of discovered units are all learned jointly. More recently, neural models have been successfully applied using contrastive and multi-task learning.

In [12] the noise contrastive estimation principle was applied to optimize the similarity of adjacent frames while making distant frames dissimilar using a neural encoder. Other work has applied contrastive learning at multiple levels by jointly optimizing both phoneme and word segmentation models [3]. Both models operate on raw speech audio input and directly learn a frame-wise

representation using a convolutional neural network. With the contrastive loss [12] obtains a data representation that differentiates adjacent and non-adjacent frames and extracts segment boundaries where the model mistakenly encodes adjacent frames as dissimilar (presumably due to spectral changes contained in a phonetic boundary/transition).

Building on [12], [3] also uses a contrastive loss to learn a frame-wise representation for phoneme boundary detection but replaces the peak picking algorithm with a differential boundary detector. Having discovered phoneme boundaries, each segment (sequence of frames belonging to the same phonetic class) is synthesized into a fixed-size representation by averaging all frames in the segment and passing the result through a neural encoder. Finally, the sequence of segments are used to construct contextualized representations using an RNN, and a second contrastive loss is applied to identify word boundaries. This model is optimized end-to-end, but the authors emphasize the trade-offs of including the secondary contrastive loss term early in training as opposed to first learning phoneme boundaries exclusively and later including words.

In subsequent sections of this thesis, we describe the use of an off-the-shelf unsupervised model to bootstrap noisy boundary labels. For this task, we used the unsupervised model from [12] described above.

2.3 Self-Supervised Pre-Training

In Section 1.4 we briefly describe concepts related to self-supervised learning and self-supervised pre-training. Here, we emphasize the importance

of these concepts for our work, point to numerous successful applications of self-supervised pretraining for downstream speech processing tasks, and detail the methodology of the wav2vec2.0 [2] and HuBERT [11], two self-supervised speech models which our work directly utilizes.

The impacts of self-supervised learning on speech processing research cannot be understated. Whereas traditional supervised learning requires manually annotated datasets, often requiring hours of expert time, self-supervised approaches use a variety of heuristics to derive the target directly from the inputs. As a result, self-supervised methods can leverage large unlabeled datasets that supervised approaches otherwise could not. Another observed and very relevant benefit of self-supervised pre-training are the resulting context-rich and general features. A great deal of thought has been devoted to developing self-supervised objectives that challenge models to learn rich structural relationships within the data that can later be exploited for a variety of specific tasks.

In a typical self-supervised learning paradigm, various heuristic strategies are employed in a so-called model pre-training phase, and later pre-trained models are fine-tuned or transfer learning is applied on specific downstream tasks. In addition to the problem of discrete unit discovery discussed in Section 1.3, many speech processing tasks have achieved new state-of-the-art (SotA) performance via the application of fine-tuning and transfer learning to the representations learned using self-supervised objectives. These include but are not limited to automatic speech recognition [2, 11, 40], emotion recog-

nition [27, 38], and speaker verification [5, 38], among others.

Borrowing ideas from research in natural language processing and computer vision, two of the most prominent self-supervised models for speech - wav2vec2.0 and HuBERT - are trained to reconstruct masked input from unmasked representations. In the following, we describe the methodologies underlying both approaches, emphasizing their points of difference.

2.3.1 wav2vec2.0

Given a raw audio input, the wav2vec2.0 model can be understood as a sequence of three steps:

- First, a 50Hz neural encoder (stacked layers of 1D convolutions) is applied to the input waveform to produce a frame-based representation of latent feature vectors.
- Latent vectors are subsequently quantized by matching features against a multiplicity of codebooks and concatenating the results.
- After randomly masking randomly sampled frames (masked frames are replaced with a learned model parameter) the frame-wise latents are passed through a multi-layer transformer network, which outputs contextualized representations of both the masked and unmasked frames.

The primary objective to be optimized is a contrastive loss that encourages each masked frame's contextualized representation to be similar to

its quantization and dissimilar to a random set of distractor quantized representations. Since neither the latent nor the quantized representation of the masked frames is provided as input to the transformer, the model is effectively tasked with reconstructing masked input and learning an expressive vector space of quantized encodings. A secondary loss is computed as the average entropy over codebook distributions, thereby encouraging the model to spread out probability mass over the entire codebook. With these two loss terms, the entire neural model is optimized end-to-end.

The wav2vec2.0 paper presents two model configurations: a base model and a large model. Architecturally, the two differ only in that the base model uses 12 transformer layers with an embedding dimension of 768 whereas the large model uses 24 layers with an embedding dimension of 1024. During training, the base model sees batches of roughly 15 second duration, while the large model sees 20 seconds. Models relevant to the work presented in this thesis are trained on the Librispeech corpus [32] containing 960 hours of audio.

2.3.2 HuBERT

HuBERT largely borrows the model architecture used by wav2vec2.0, including a convolutional feature encoder and a transformer-based context network. HuBERT differs primarily in its approach to quantization. In addition, while wav2vec2.0 computes the contrastive loss only for masked frames, different versions of HuBERT optimize against a contrastive loss, which is a weighted combination of loss terms for masked and unmasked frames.

Instead of maintaining and learning codebooks, HuBERT creates quantized targets for contrastive learning using an offline K-means clustering step. Initally, targets are bootstrapped by running K-means over Mel-frequency cepstrum coefficients feature vectors computed from the training set audio. 13 coefficients and their first and second derivatives are included creating feature vectors with 39 dimensions. The bootstrapped targets are used for the first full training run. Once complete, the resulting context network representations (from some intermediate transformer layer) are used as input to a second run of K-means to construct new targets for a second training. In the first and second run of K-means, the numbers of centroids (read quantized targets) are set to 100 and 500, respectively.

Not unlike the two formulations of wav2vec2.0, HuBERT is proposed in a base, large, and extra-large format each with 90M, 300M, and 1B parameters, respectively.

Chapter 3: Modeling Framework

In this chapter we describe and formalize the two models we have developed and investigated. Later, the learning framework is discussed and the training objective stated.

3.1 Backbone Networks

Inspired by the success of self-supervised learning in numerous speech processing tasks we adopt pre-trained wav2vec2.0 and HuBERT model checkpoints to compose the backbone of a frame-wise binary classifier. Both pre-trained models share a similar architecture, which we formalize below.

3.2 Pre-Trained Encoder

For our purposes we consider only the encoder of the two prospective backbone models, which we denote by the function composition $g \circ f$. Elaborating on the component functions of this composition, $f : \mathcal{X} \to \mathcal{Z}$ is commonly referred to as the convolutional feature extractor, which processes raw waveform input and outputs a time series of latent speech representations. Thus, f acts like the previously defined f_x; however, f is not strictly a pre-processing step since it is learned during end-to-end training of wav2vec2.0 and HuBERT. Meanwhile, $g : \mathcal{Z} \to \mathcal{C}$ is known as the context network, which applies learned

attention masks to synthesize a context-aware representation $c_i \in \mathcal{C}$ from each $z_i \in \mathcal{Z}$. g is itself a compositional function built from a cascade of n transformer self-attention blocks. Thus, we can also write $g = g^n \circ g^{n-1} \circ ... \circ g^1$. Note that functions f and g may be initialized by either wav2vec2.0 or HuBERT; however, for any instance of our model both f and g must come from the same pre-trained model.

3.3 Classifier Models

We develop two separate classification model formulations built on-top of the pre-trained network backbone, which we denote as fine-tune and readout mode, respectively. Below, we describe and define both models.

3.3.1 *Fine-tune* Mode

Fine-tune mode, appends a single linear projection layer, h^{ft}, to the output of the pre-trained model. As the name suggests, in this setting, the entire pre-trained model and added projection receive gradient updates, and the model can be formalized as the function composition $h^{ft} \circ g \circ f$.

3.3.2 *Readout* Mode

Readout mode, is depicted in Figure 3.1. Here, we freeze the pre-trained model and apply learned, layer-specific convolutions, $h^{c,1}, h^{c,2}, ..., h^{c,n}$, to feature representations extracted from each g^i. The outputs are then summed and

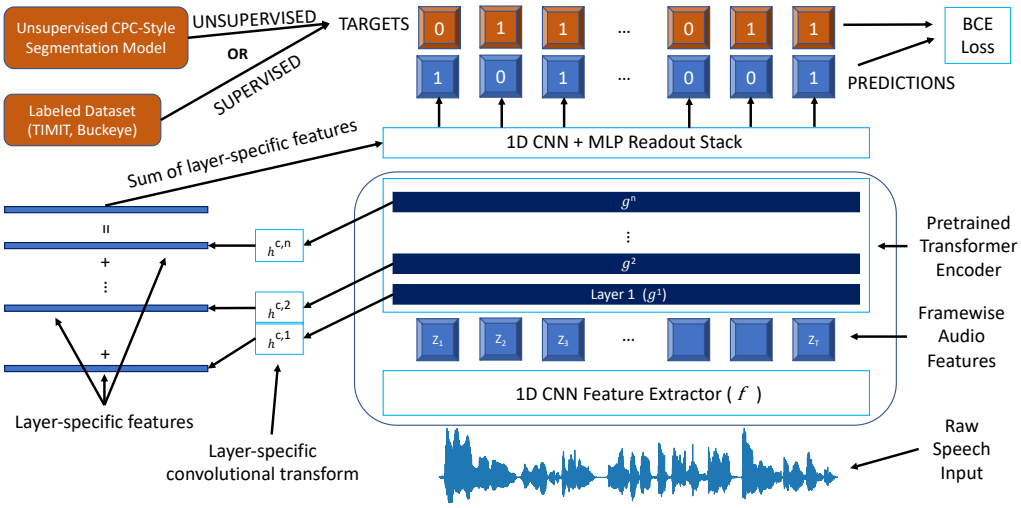

Figure 3.1: *Readout* model architecture schematic. A pre-trained model extracts hierarchical features from the raw waveform. Features are processed by a series of convolutional networks and probability scores are computed. Finally, binary cross entropy loss is evaluated using model predictions and either ground truth labels or noisy labels estimated in an unsupervised manner.

passed through a final series of convolutions and perceptron layers, denoted h^{ro}. Empirically, we discovered that applying a learned weight parameter to each layer's processed features before computing the summation improved performance; however, we omit these terms in the following expression for simplicity. Denoting the outputs of each g^i as c^i, the *readout* model can be formalized as $h^{ro}(\sum_i h^{c,i}(c^i))$.

3.4 Model Prediction & Loss Formulation

Both models output a series of frame-wise floating-point numeric values to which the Sigmoid activation function - Equation 3.1 - is applied to derive

boundary probability scores. Probability scores > 0.5 are transformed into a 1 and 0 otherwise where a 1 is interpreted as the occurrence of a boundary. The resulting binary string is denoted $\hat{y}_b$. Given a training set of input utterances $\mathcal{S} = \{x^i, y_b^i\}_{i=1}^m$, loss is computed and models are updated according to the binary cross entropy (BCE) objective function in Equation 3.2. Here, the term w^* is a strictly positive weight value assigned to the loss associated with frames where the reference ground truth indicates the presence of a boundary.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$\mathcal{L}_{BCE} = \sum_{i=0}^{m} \sum_{j \in y_b^i} w^* y_{b,j}^i \log(\hat{y}_{b,j}) + (1 - y_{b,j}^i) \log(1 - \hat{y}_{b,j}) \tag{3.2}$$

3.5 Dual Learning Framework

We believe one of the most interesting aspects of our work is the successful application of the above objective in both the supervised and unsupervised settings. In the former case, we simply rely on the time-aligned transcriptions that come as part of the TIMIT and Buckeye corpora to construct supervised targets. In the latter case, we first perform inference over the TIMIT and Buckeye training sets using the pre-trained unsupervised model provided by Kreuk et al. [12]. Later, the predicted labels then are taken as the supervisory signal used during training, and optimization occurs in a manner nearly identical to the purely supervised case.

Chapter 4: Evaluation

Previous work has articulated the challenges of conducting representative evaluations of phoneme segmentation results [35]. These issues are further confounded by the wide range of (prediction to label) temporal tolerance levels found in the literature for true positives identification. Here, we consider a 20 millisecond tolerance window on either side of a ground truth label for true positive calling, which is consistent with recent work [3, 12, 13, 20].

We report our results in terms of precision, recall, F1, and R-Value according to their definition in [35]. These metrics and their interpretation are widely cited in the phoneme segmentation task literature; however, based on a combination of published code review and attempts to reproduce results we believe there remains meaningful ambiguity in the calculation of precision, recall and their derivative quantities (e.g., F1 and R-Value). By elaborating on these definitions and their implications below, we hope to align the community around shared standards for reporting phoneme boundary detection results.

4.1 Sources of Ambiguity

When computing precision the primary source of ambiguity revolves around interpreting multiple positive boundary predictions falling within the tolerance window of a ground truth boundary. For recall, the parallel situation

arises where a single predicted positive boundary falls within the tolerance window of more than one ground truth boundary. See Figure 4.1 for a visual illustration of the ambiguous situations arising during evaluation.

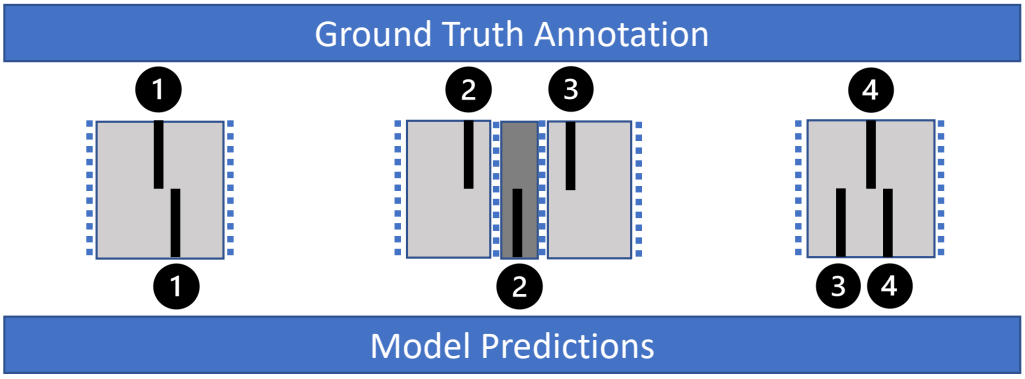

Figure 4.1: Illustration of ambiguities during phoneme segmentation evaluation. Vertical black stripes indicate ground truth (top) and predicted (bottom) boundaries. The light gray regions correspond to ground truth boundary tolerance windows and the dark gray region shows where two tolerance windows overlap. Predicted and ground truth boundaries 1 match. Ground truth boundaries 2, 3 both match predicted boundary 2 while predicted boundaries 3 and 4 both match and ground truth 4.

4.2 Worst Case Metrics Deviation

Without loss of generality, consider a boundary predictor operating at 50Hz (i.e. predictions correspond to the occurrence of a boundary in a 20 millisecond window). While computing the precision of this model's predictions with a 20 millisecond tolerance window, one could encounter an isolated ground truth boundary at frame p and model boundary predictions at $p-1$, p,

$p + 1$. Different approaches to this calculation could result in a three-fold difference in performance, and the situation would be exacerbated by an increase in the predictor's frame rate. In practice, the statistics of English language phoneme duration and presentation render a three-fold performance difference highly unlikely; however, others have reported differences of up to 5% [34], and our results consistently show deviations of 3-4% in the supervised setting and 5-7% in the unsupervised setting.

4.3 *Strict* & *Lenient* Evaluation Schemes

Past occurrences of and future potential for dramatic deviation in key evaluation metrics reporting demand an explicit and rigorous statement of our evaluation framework. We go further and propose to delineate two distinct evaluation schemes - *strict* and *lenient* - for the community to use moving forward. Put simply, the *strict* scheme prohibits double counting and the *lenient* scheme allows it. Specifically, while computing the hit rate [35] of an automated phoneme segmenter, in the *strict* scheme once a ground truth boundary is matched by a predicted boundary the ground truth is removed from consideration for matching additional model predictions. On the other hand, in the *lenient* scheme the same ground truth boundary may match multiple predicted boundaries so long as they fall within the tolerance window. Further, in the *lenient* scheme, the hit rate used for computing precision and that for recall may differ since more than one predicted boundary is allowed to match a the same ground truth and visa versa.

One could, in theory, imagine alternative evaluation schemes that fall somewhere between the extremes of our proposed *strict* and *lenient* definitions. Incidentally, for different applications of a phoneme boundary detection tool (or a related frame-wise, binary sequence classification task) there may be greater or lesser tolerance for false positives, false negatives, etc. Put another way, a specific use case could argue for an alternative, hybrid evaluation scheme that, for example, penalizes double counting in precision but not recall computations or visa versa. It is our position that, in the case of the phoneme boundary detection task, such an evaluation scheme could be proposed but its interpretation and complexity ought to be rigorously justified. In this section we discuss the ways in which accurate evaluation of model predictions - as evidenced by inconsistent results reporting - is not straightforward. The *strict* and *lenient* schemes are, in their dichotomy, easy to interpret and can serve as a common, well-defined measuring stick for the community. In general, we believe the *strict* scheme provides a truly unambiguous representation of performance and should be preferred when reporting phoneme boundary detection results.

4.4 Past and Future Results Reporting

In later sections, we denote all results following the *strict* scheme with a * and then define F1* and R-Value* as those metrics computed with their *strict* counterparts P* (precision*) and R* (recall*). Our code reviews and efforts to reproduce past results indicate that certain previous SotA methods used the

lenient scheme. For parity, our results tables below include both *strict* and *lenient* scores for our models. In some cases, where we were able to reproduce previous published results, we also add *strict* scores. In other cases, it was not possible to verify the exact evaluation framework used by some authors. However, all these papers explicitly describe sharing evaluation methodology with the aforementioned previous SotA, against which they benchmark their model performance. Accordingly, we assume they also evaluate performance using the *lenient* framework.

Chapter 5: Experimental Setup and Results

5.1 Datasets

We used the TIMIT [8] and Buckeye [33] speech corpora to train and evaluate both the *fine-tune* and *readout* model. For the benefit of future research with these corpora, in the phoneme boundary detection task or otherwise, below we describe our approach to constructing training, validation, and testing datasets from the raw data.

5.1.1 TIMIT

For TIMIT, we used the standard training and testing data split and sampled 10% of the training data at random to construct a validation set. Out of the box, TIMIT is composed of single sentence recordings, and we found it unnecessary to split, truncate, or in any way shorten data samples. Accordingly, we did not find the need to split individual recordings into shorter segments or modify them in any way.

5.1.2 Buckeye

For Buckeye, we followed previous work [7, 12, 13] in our training, validation, and testing set construction. First, we split the corpus at the speaker level, reserving 80%, 10%, 10% for training, validation, and testing, respec-

tively. We used speakers *s07, s03, s31, s34* for testing and speakers *s40, s39, s36, s25* for validation. All remaining speakers were used for model training. This particular assignment of speakers was selected to identically match the splits used by [12, 13], the leading supervised and unsupervised phoneme segmentation methodologies at the time we conducted our experiments.

Unlike TIMIT, Buckeye is composed of many multi-minute recordings, some of which contain extended periods of non-speech silence or noise. It is impractical to load large batches of long audio recordings into memory during training and wasteful to train on non-speech audio. Accordingly, we followed the procedures used by [7,12,13] to extract relevant speech recordings from the Buckeye raw dataset. Recordings were split during vocal noise, non-vocal noise, and silence into shorter continuous speech segments such that each segment starts and ends with no more than 20 milliseconds of non-speech. In cases where a selected segment's annotated transcription contained tokens we were unable to interpret the segment was dropped. Additionally, segments were required to contain no less and no more than a pre-specified minimum and maximum number of phonemes. Again following [12, 13] we set the minimum and maximum to 20 and 50, respectively. In some cases, the transcribed annotations showed tokens that began or ended following the actual end of the corresponding audio clip. These cases were also ignored, and the segments were thrown out.

5.1.3 Pre-Processing Code Availability

As referenced in the abstract of this thesis, we provide a publicly available implementation of our methods including data pre-processing. The scripts that we used to construct both the TIMIT and Buckeye training, validation, and testing sets are available there. To facilitate direct comparison of methodologies and results we encourage future research on this topic to consider these scripts a starting point.

5.2 Experimental Setup

Below we state parameter values and details of the core experiments we conducted. Secondarily, we provide summary analysis of investigations related to the relative importance of pre-trained model features and the effects of positive loss weighting on the precision-recall tradeoff.

5.2.1 Pre-Trained Checkpoints

Experiments conducted with HuBERT used the base architecture and those with wav2vec2.0 used the small architecture. Both model checkpoints were pre-trained on Librispeech [32] and collected from Fairseq [31]. We explored the effectiveness of larger model architectures (e.g. wav2vec2.0 large, HuBERT large/x-large) but found they offered no boost on final performance metrics. For our unsupervised experiments, we used model checkpoints made available with the code accompanying [12] to bootstrap labels for TIMIT and Buckeye.

5.2.2 Optimization and Model Parameters

All models were trained on an NVIDIA Quadro RTX 8000 with a batch size of 16 for 50 epochs. The Adam optimizer was used with a learning rate of 1e-3 and 1e-4 while training in *readout* and *fine-tune* mode, respectively. Models were regularly evaluated during training using the validation set's R-Value* and the best performing model was saved for testing.

In *readout* mode the layer specific convolutions, $h^{c,i}$ were defined with a kernel size of 9, stride of 1, and 768 input and channels. The output architecture h^{ro} is a depth five convolution stack with a shared kernel size of 3 and stride of 1 followed by a linear projection. As we mentioned previously, in this setting we also added a parameter to learn a weighted sum of the layer-specific features before application of h^{ro}. The layer-specific weight values are initialized equally to $\frac{1}{12}$, as both backbone models contain 12 layers. Figure 5.2 shows the learned weights for each layer for a wav2vec2.0 *readout* model trained on TIMIT and Buckeye, separately. The weight distribution indicates that the early layers of the stack contain the most relevant features at the phonetic timescale.

5.2.3 Layer Subset ***Readout*** Models

To verify our interpretation of the learned, layer-specific weighting factors we examined performance of *readout* models that synthesized only a subset of all available layer features. Based on Figure 5.2 we trained wav2vec2.0 *readout* models on TIMIT and Buckeye using only those layers that together con-

stituted $> 85\%$ of the total absolute magnitude of weighted sum factors that were learned with all available layer features. For models trained on TIMIT and Buckeye, we used layers 1-6 and 1-7 which together make up approximately 87% and 86% of the total absolute weight magnitude, respectively. Per the results reported in Table 5.1, in all metric categories these reduced models are competitive with those that work over all 12 transformer-layer features. In some metric categories, the reduced set actually outpace the scores obtained in the full *readout* setting. However, since we did not observe any substantial differences in performance in these and related experiments, we chose to focus on reporting on the full *readout* models, which are, in theory, fully capable of learning layer nullifying weights without manual intervention. Nevertheless, future work could examine in much greater depth the semantics of features contained in various pre-trained model layers. In doing so, it is likely one could further optimize the performance - in both metrics and computational cost - of the modeling framework we propose here.

5.2.4 Positive Class Loss Weighting

Throughout our experiments, we explored various values of w^* - the loss weight applied to frames labeled as boundary positive. In all supervised experiments, w^* was ultimately set to 1 for the entire duration of model training. We made anecdotal observations that setting $1 < w^* < 2.5$ tended to speed up model convergence; however, as expected, with a positive weight > 1 we observed particularly high recall scores while precision suffered, and w^*

had to be subsequently turned down to obtain the best performance metrics. Notably, the vast majority of false positives occurred near true boundaries. That is, most of the false positive predictions of the model were not erroneous, and the *lenient* evaluation metrics often pointed favorably to a larger w^*. We explored several techniques to regularize redundant positive prediction near true boundaries, including local response normalization [14], L1 sparsity regularization, and differentiable binarization [18]. In all cases, we were unable to bring model precision scores within 20%-25% of those obtained with $w^* = 1$ and no additional regularization.

In the unsupervised setting, we found that, relative to ground-truth labels, noisy labels scored substantially lower in recall than precision. Acknowledging the need then to induce positive predictions, we swept values of w^* and obtained optimal test set performance using $w^* = 1.4$.

5.2.5 HuBERT Discrete Unit Baseline

In Section 5.3 - see Table 5.2 - we report results for our models and compare them directly to those reported in previous work on the phone boundary detection task. In addition, we provide an alternative unsupervised baseline for comparison that is based on matching frames to discrete units discovered during the HuBERT training. As discussed previously, the HuBERT training protocol uses K-means clustering to identify training targets for masked frames. We hypothesized that one might obtain a reasonable segmentation result by treating transitions in the discrete unit sequence as phoneme bound-

aries. Also, a baseline derived exclusively from the backbone models we use can underscore both the rich information contained within them and the value-add of our approach to refine their representations.

To test our hypotheses, we downloaded the pre-trained K-means model from Fairseq [31], which comes from the second iteration of clustering using transformer layer 9 representations. We then simply extract the layer 9 HuBERT representations for the TIMIT and Buckeye test splits, assign each frame to its closest K-means centroid, and call boundaries at the beginning of each new contiguous centroid label segment. We also computed evaluation metrics where boundaries are called at the end of each block of contiguous labels but observed generally lower performance metrics. Those familiar with run-length encoding will recognize its similarity to our procedure.

In summary, we find that the HuBERT baseline obtains very high recall scores - under both the *harsh* and *lenient* schemes - but suffers from very low precision. This result suggests the units derived from layer 9 representations likely correspond to transitions at a more rapid timescale than typically observed for phonetic transitions. Still, the high recall scores indicate that HuBERT has effectively decomposed phonetic segments but does so with smaller, sub-phoneme building blocks.

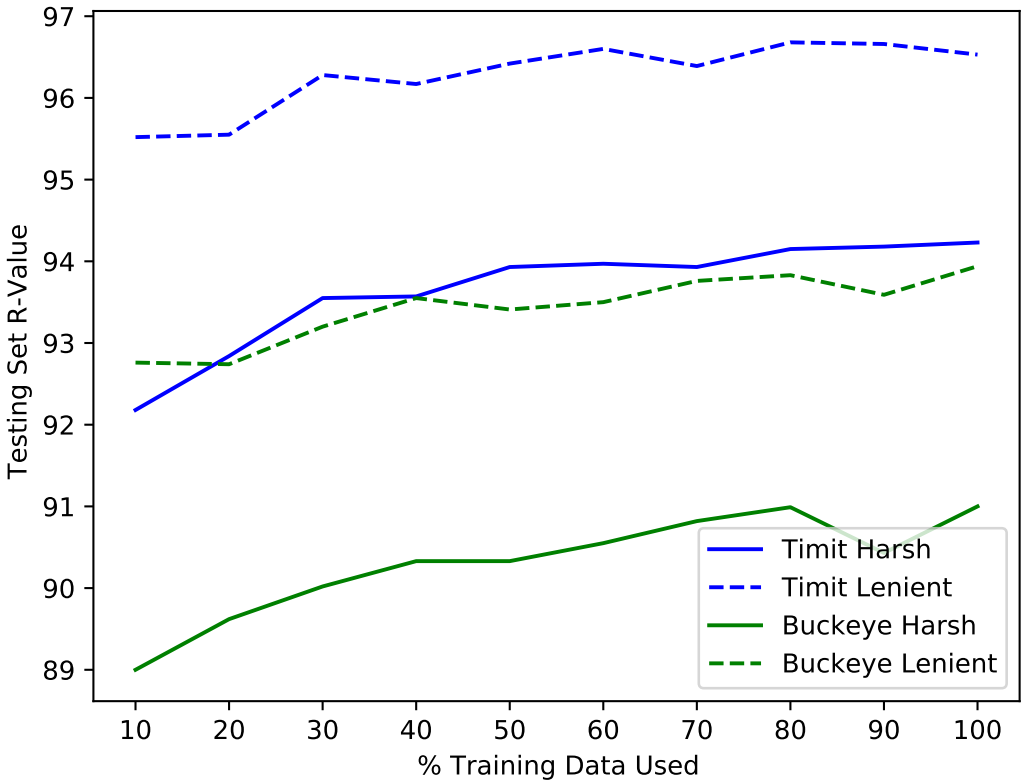

Figure 5.1: Supervised model with wav2vec2.0 backbone in *readout* mode trained from scratch on incrementally larger fractions of labeled data. Vertical axis shows testing set R-Value performance.

5.3 Results

The following sections present results for both *readout* and *finetune* models composed using wav2vec2.0 and HuBERT. We report results in the form of two tables - each of the two corresponding to the supervised or unsupervised case - that highlight the superiority of our methodology in terms of precision, recall, F1-score, and R-value. In addition, we show the data efficiency of our technique for a wav2vec2.0-based *readout* model.

5.3.1 Supervised Setting

In Table 5.1 we report results for our models in the fully supervised setting. We also include reported scores from Lin et al. [20], Kreuk et al. [13], which stand as previous benchmark results in text-dependent and text-independent phoneme segmentation, respectively. Another result we include is our attempt at reproducing Kreuk et al.'s [13] results for TIMIT - here we are able to share both the *harsh* and *lenient* evaluations. We were unable to reproduce comparable scores for Buckeye using the model from [13]. Altogether, results indicate that the best of our four models - composed through a selection of a backbone pre-trained network and *fine-tune* or *readout* mode - eclipse previous SotA in every metric category for both TIMIT and Buckeye. With few exceptions, all four of our models outpace previous SotA, and we emphasize that our top performing model, which was trained in the text-independent setting, surpasses the performance of SotA text-dependent [20].

Figure 5.1 highlights the small amount of labeled training data required to surpass previous SotA performance. Results reported in this figure come from experiments with a *readout* mode model trained with a wav2vec2.0 backbone. For both TIMIT and Buckeye we obtain R-Value SotA using only 10% of the labeled data from the respective training sets.

Data	Model	Precision	Precision*	Recall	Recall*	F1	F1*	R-Value	R-Value*
Buckeye	Lin et al. [20] $^{\underline{oo}}$	88.49	NA	90.33	NA	89.40	NA	90.90	NA
	Kreuk et al. [13] $^{\underline{oo}}$	85.40	NA	89.12	NA	87.23	NA	88.76	NA
	W2V2 finetune	**94.01**	**90.56**	93.08	**90.28**	**93.54**	**90.42**	**94.41**	**91.81**
	HuBERT finetune	93.83	89.81	**93.11**	**90.28**	93.47	90.05	94.37	91.51
	W2V2 readout	93.38	89.14	92.74	89.66	93.00	89.40	93.99	90.96
	W2V2 6-layer readout	93.42	89.74	92.58	89.47	93.00	89.60	93.96	91.11
	HuBERT readout	93.37	89.30	92.95	89.94	93.16	89.62	94.13	91.15
TIMIT	Lin et al. [20] $^{\underline{oo}}$	93.42	NA	95.96	NA	94.67	NA	95.18	NA
	Kreuk et al. [13] $^{\underline{oo}}$	94.03	NA	90.46	NA	92.22	NA	92.79	NA
	Kreuk et al. [13]	92.94	92.14	92.31	89.26	92.63	90.68	93.66	91.71
	W2V2 finetune	96.90	**94.35**	**96.30**	**93.91**	**96.60**	**94.13**	**97.04**	**94.96**
	HuBERT finetune	**96.93**	94.31	96.09	93.68	96.51	94.00	96.92	94.83
	W2V2 readout	96.67	93.75	95.56	92.65	96.11	93.20	96.55	94.10
	W2V2 7-layer readout	96.38	93.70	95.73	92.78	96.05	93.23	96.57	94.15
	HuBERT readout	96.50	93.23	95.93	93.47	96.21	93.35	96.71	94.33

Table 5.1: Results obtained in the fully supervised setting. * Indicates application of the strict evaluation framework and $^{\underline{oo}}$ denotes author reported scores. The NA placeholder is used where results are not available. Bolded values indicate highest score for the specific metric and dataset. W2V2 6-layer and 7-layer readout models correspond to the experiments which synthesized only a subset of available pre-trained features - see Section 5.2.3 for full context.

Data	Model	Precision	Precision*	Recall	Recall*	F1	F1*	R-Value	R-Value*
Buckeye	Bhati et al. [3] ^{oo}	76.53	NA	78.72	NA	77.61	NA	80.72	NA
	Kreuk et al. [12] ^{oo}	75.78	NA	76.86	NA	76.31	NA	79.69	NA
	Kreuk et al. [12]	77.17	72.21	79.71	75.55	78.42	73.85	81.39	77.28
	W2V2 finetune	82.15	75.56	**85.13**	**79.47**	83.61	77.47	85.81	80.33
	HuBERT finetune	83.09	76.62	84.47	78.75	83.77	**77.67**	86.11	80.79
	W2V2 readout	**84.24**	**77.92**	82.88	77.41	83.55	**77.67**	85.92	**80.95**
	HuBERT readout	83.35	75.29	84.68	79.37	**84.01**	77.28	**86.31**	80.13
	HuBERT baseline	66.48	39.01	92.72	91.88	77.44	54.77	63.41	-0.1868
TIMIT	Bhati et al. [3] ^{oo}	84.63	NA	86.04	NA	85.33	NA	87.44	NA
	Kreuk et al. [12] ^{oo}	83.89	NA	83.55	NA	83.71	NA	86.02	NA
	Kreuk et al. [12]	85.27	81.42	83.48	76.53	84.36	78.90	86.57	81.71
	W2V2 finetune	88.93	82.16	**88.60**	80.83	88.76	81.49	90.40	84.18
	HuBERT finetune	89.05	82.07	88.44	80.70	88.75	81.38	90.37	84.08
	W2V2 readout	90.69	**84.92**	86.78	78.52	88.69	81.59	89.90	83.69
	HuBERT readout	**90.98**	82.44	88.48	**81.18**	**89.71**	**81.81**	**90.98**	**84.45**
	HuBERT baseline	69.01	42.36	93.67	92.70	79.47	58.15	66.98	-4.12

Table 5.2: Results obtained in the unsupervised setting. A noisy label-set was furnished using publicly available checkpoints from an unsupervised segmentation model [12]. HuBERT baseline refers to our experiments that use HuBERT's pre-trained discrete units and run length encoding to segment phonemes - see Section 5.2.5 for full context.

5.3.2 Unsupervised Setting

Table 5.2 reports results for models in the unsupervised setting along with other previous SotA results. As in the supervised case, our best performing model achieves a new SotA result for both TIMIT and Buckeye in every metric category. Notably, wherein the supervised setting a typical deviation between the *lenient* and *harsh* schemes is in the 2-3% range, in the unsupervised setting we observe deviations of, in some cases, more than 8%. As the Kreuk et al. [12] and Bhati et al. [3] unsupervised models reported here perform inference through a peak-picking algorithm over a learned representation, it is possible that over prediction near boundaries stems from the difficulty of enforcing temporally precise transitions in the learned representation. Similarly, as our models are trained using a noisy label-set boostrapped from [12], our model is liable to the same failure modes.

During experiments with noisy (unsupervised) label-sets, we explored the impact of multiple self-training loops to refine the labels and improve final model test performance. Ultimately, we observed marginal gains that did not inspire a deep exploration of how bootstrapped labels could be refined in an unsupervised fashion. In fact, in *fine-tune* mode, performance declined after multiple self-training loops. Incidentally, throughout our experiments in the unsupervised setting, *readout* models tended to perform better than their fully fine-tuned counterparts. Relevant metrics observed during training indicated that the more expressive fine-tuned models were much more liable to over-fit label noise than their *readout* mode counterparts.

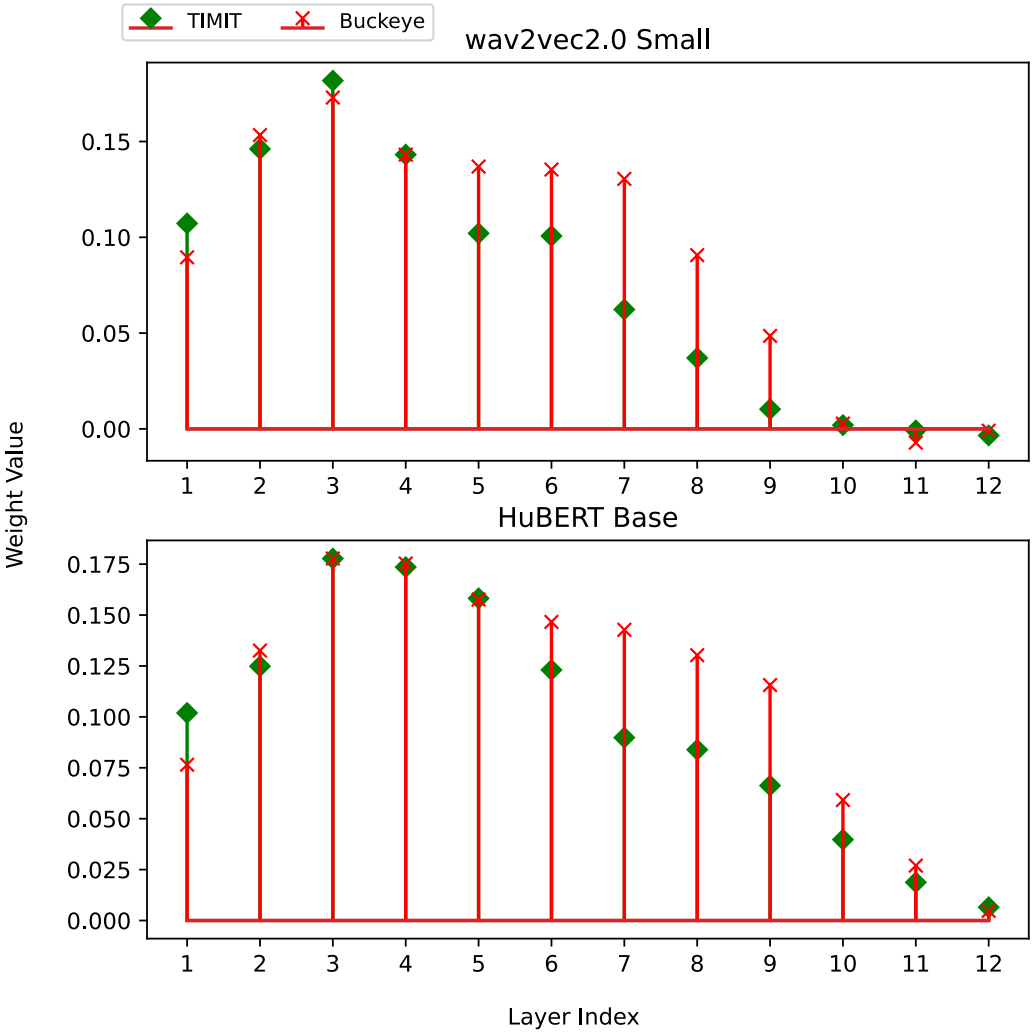

Figure 5.2: Learned layer-specific weights from *readout* models after training on the TIMIT and Buckeye corpora. Later layers contribute less relative weight in the model's feature summation than early and middle layers.

Chapter 6: Conclusions and Future Work

6.1 Recap

Here we introduced a new model formulation based on self-supervised pre-training and transfer learning to perform phoneme boundary detection in the supervised and unsupervised settings. Using pre-trained, self-supervised models wav2vec2.0 and HuBERT as the backbone, we propose two novel model formulations titled *finetune* and *readout* mode that learn end-to-end or freeze and synthesize pre-trained features for a downstream phonetic boundary segmentation task.

We highlight common ambiguities that arise in evaluating performance of boundary detectors and show that different strategies for computing key evaluation metrics can lead to substantial deviations in reported results. Thus, our work brings to the community's attention a need for shared implementation strategies for key evaluation metrics. In response, we propose two evaluation frameworks titled *lenient* and *harsh* with clear interpretations as unequivocally allowing or disallowing double counting during precision and recall computation. We believe these schemes can be used to alleviate future confusion and align the community around common standards.

Under the *strict* and *lenient* evaluation schemes our methodology sets new SotA benchmarks in both the supervised and unsupervised settings on

standard datasets used for the task - the TIMIT and Buckeye speech corpora. In fact, we show that our approach is highly data efficient, and, in the supervised setting, is able to set new SoTA performance metrics using only 10% of the labeled training set. In the unsupervised setting, we leverage an existing unsupervised segmentation method to furnish noisy pseudo-labels for use in our simple BCE-based supervised learning framework. Surprisingly, our model is able to overcome label noise and significantly improve upon the teacher.

Broadly speaking, the contributions of this thesis can be contextualized under ongoing areas of research in discrete unit discovery for pseudo-continuous speech audio and computer-assisted language and linguistic documentation. Speaking of the former, the successes of pre-trained language models in NLP have motivated researchers in automatic speech processing to seek out new methods for transforming (tokenzing) speech recordings into a symbolic sequence over which PLMs and LLMs can operate. A phonetic segmentation is a natural basis from which discrete units can be discovered and a speech recording tokenized. Furthermore, it can be studied and contextualized in the rigor and history of classical linguistic science.

In the latter case, effective documentation and annotation for low resource and endangered languages is paramount to the preservation of humankind's cultural diversity. Several tools are in use today to assist experts in phonetic transcription and general speech annotation, which together make up a very large fraction of the man-hours required to effectively document a language. Incorporating a technology for automatic phonetic segmentation

into these tools - such as the one we propose here - could further accelerate aspects of the phonetic-unit discovery and annotation process.

6.2 Future Work

We believe there are several promising directions for future work. Below we discuss four potential areas of inquiry that relate directly to the work presented in this thesis.

6.2.1 Self-Training & Noisy Label Regularization

First, an exploration of regularization and self-training strategies to improve noisy label-sets will likely push unsupervised results further than we have been able to. Somewhat mysteriously, despite using noisy pseudo-labels as supervised training targets, our models were able to learn from and surpass performance of the teacher unsupervised segmentation model. This fact was among the most compelling and confusing of our work, and it raised many questions from reviewers and the community at large.

We believe the phenomena may be result of the representational expressivity and flexibility of the self-supervised approaches used to pre-train HuBERT and wav2vec2.0. The success of our approach – particularly with small amounts of labeled data – indicates that the masked prediction task implicitly encodes frame-wise phonetic identity, among other rich structural information, which may regularize noisy input. Other recent works have shown a self-supervised auxiliary loss applied to the teacher model can improve conven-

tional and noisy knowledge distillation results. In our case, both the student and teacher model are trained/pre-trained in a self-supervised fashion making exploitation of structural relationships in the data possible from multiple angles. In addition, it has been demonstrated in previous works that that student models explicitly regularized with stochastic depth and data augmentation can outperform imperfect teacher models. In our case, we apply different explicit regularizations – layer dropout and loss weighting – but likely benefit from a similar principle.

We did not, however, spend extensive time experimenting with alternative regularization or self-training strategies and believe it may still be possible to compose a virtuous cycle in which the student surpasses and replaces the teacher after successive training run. We have already demonstrated that this is feasible when the student model has been exposed to the right heuristics before learning from the teacher. The question remains as to what the limit may be on incremental improvement and how to best prepare the student and/or the data in subsequent iterations.

6.2.2 Application to Low Resource Languages

As we have discussed, tools for automatic linguistic annotation can accelerate language documentation for endangered - and typically low resource - languages. In the supervised setting we obtained excellent performance even with small amounts of training data, so we are optimistic that low resource languages can benefit from self-supervised pre-training for phoneme boundary

detection.

Still, our work has not dealt specifically with the low resource case, and the methodology proposed here depends significantly on the existence of a pre-trained model, which typically are learned over large corpora of unlabeled data. In the low resource setting, even unlabelled corpora are often limited in size and scope. There is an opportunity then to investigate the construction of pre-trained models useful for downstream phoneme segmentation in low data regimes (annotated or not). At the same time, it is significantly less resource intensive to construct an unlabelled dataset than to annotate said data. So, in addition to exploring techniques for effective self-supervised pre-training in low data regimes, the evidence presented in this work can help motivate investment in data collection and self-supervised model training for low-resource languages.

6.2.3 Significance of Layer-Wise Pre-Trained Model Features

We showed that models trained using all layer features and a learned weighted summation to combine them clearly favored a smaller subset - typically, $> 85\%$ of absolute weight magnitude was contained in 6 of the 12 layers. Also, we documented the high performance metrics obtained by *readout* models trained with only a subset of pre-trained transformer layer features. The fact that, in certain metric categories, a *readout* model trained using 50% of all available information in the pre-trained model outperforms a model using 100% points to the task specific relevance of different layer representations.

In our opinion, a more thoughtful and thorough investigation of the semantics of different layer representations and their relationship to one another would likely provide insights that could improve our model's performance. Alternatively, with more time and computational resource, a purely data-driven and empirical approach could identify the optimal subset of information contained within a pre-trained model for the task at hand, whatever that may be. For example, in Subsection 6.2.4 below we discuss hypothetical applications of our methodology to alternate speech segmentation tasks such as word segmentation and speaker diarization. It is likely that applications of self-supervised pre-trained models to these problems would also benefit from a deeper understanding of which features are most relevant to the discovery of word and speaker boundaries, respectively.

6.2.4 Alternate Speech Segmentation Tasks

Our model formulation may be, with minimal modifications, well-suited to alternate speech segmentation tasks such as word segmentation or speaker diarization - the partitioning of a multi-party speech recording by speaker.

We have conducted preliminary experiments on supervised and unsupervised word segmentation using our model. In the supervised case we rapidly obtained very promising results, even with small subsets of the training set. In the unsupervised case, existing word segmentation baselines perform poorly relative to phoneme segmentation. As a consequence, the initial noisy label set we were able to bootstrap was far noisier, and the model struggled to reliably

or substantially improve on the teacher signal. Here, as previously discussed, an exploration of strategies for label noise regularization and self-training could lead to an effective unsupervised word segmentation model. On a related note, word boundaries are significantly more sparse than phonetic boundaries, so in any dataset there are fewer positive word boundaries to learn from. In our unsupervised experiments we found that larger values of w^*, the positive instance loss weight, helped compensate for fewer positive boundary instances but did so at the expense of dramatic over-prediction near boundaries.

We have not explored speaker diarization in any form but are interested to understand if our modeling framework could apply successfully. An alternative, intermediate stepping stone task might in this direction might involve phoneme or word boundary detection in multi-speaker audio recordings.

Bibliography

[1] Antonios Anastasopoulos. *Computational Tools for Endangered Language Documentation*. PhD thesis, 2019. Copyright - Copyright ProQuest Dissertations Publishing 2019; Last updated - 2023-06-21.

[2] 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.

[3] Saurabhchand Bhati, Jesús Villalba, Piotr Żelasko, Laureano Moro-Velazquez, and Najim Dehak. Segmental contrastive predictive coding for unsupervised word segmentation, 2021.

[4] Paul Boersma. The use of praat in corpus research. 2014.

[5] Zhengyang Chen, Sanyuan Chen, Yu Wu, Yao Qian, Chengyi Wang, Shujie Liu, Yanmin Qian, and Michael Zeng. Large-scale self-supervised speech representation learning for automatic speaker verification. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6147–6151. IEEE, 2022.

[6] Sorin Dusan and Lawrence Rabiner. On the relation between maximum spectral transition positions and phone boundaries. In *Interspeech*, 2006.

[7] Joerg Franke, Markus Mueller, Fatima Hamlaoui, Sebastian Stueker, and Alex Waibel. Phoneme boundary detection using deep bidirectional lstms. In *Speech Communication; 12. ITG Symposium*, pages 1–5. VDE, 2016.

[8] John S Garofolo. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

[9] James Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 13:137–152, 2003.

[10] Nikolaus P Himmelmann et al. Language documentation: What is it and what is it good for. *Essentials of language documentation*, 178(1), 2006.

[11] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[12] Felix Kreuk, Joseph Keshet, and Yossi Adi. Self-supervised contrastive learning for unsupervised phoneme segmentation, 2020.

[13] 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.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[15] Kushal Lakhotia, Evgeny Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Adelrahman Mohamed, and Emmanuel Dupoux. Generative spoken language modeling from raw audio, 2021.

[16] Chia-ying Lee and James Glass. A nonparametric Bayesian approach to acoustic model discovery. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2012.

[17] Xinjian Li, Ye Jia, and Chung-Cheng Chiu. Textless direct speech-to-speech translation with discrete speech representation, 2022.

[18] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11474–11481, 2020.

[19] Mark Liberman. The problems of scale in language documentation. *Plenary talk at TLSX Texas Lin*, 2006.

[20] Binghuai Lin and Liyuan Wang. Learning acoustic frame labeling for phoneme segmentation with regularized attention mechanism. In *ICASSP*

2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7882–7886. IEEE, 2022.

[21] Guan-Ting Lin, Yung-Sung Chuang, Ho-Lam Chung, Shu wen Yang, Hsuan-Jui Chen, Shuyan Dong, Shang-Wen Li, Abdelrahman Mohamed, Hung yi Lee, and Lin shan Lee. Dual: Discrete spoken unit adaptive learning for textless spoken question answering, 2022.

[22] Alexandre Magueresse, Vincent Carles, and Evan Heetderks. Low-resource languages: A review of past work and future challenges. *CoRR*, abs/2006.07264, 2020.

[23] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pages 498–502, 2017.

[24] Alexis Michaud and Eric Castelli. Towards the automatic processing of yongning na (sino-tibetan): developing a 'light' acoustic model of the target language and testing 'heavyweight' models from five national languages. 05 2014.

[25] Sarah Moeller. *Integrating Machine Learning into Language Documentation and Description*. PhD thesis, 2021. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-03-08.

[26] Abdelrahman Mohamed, Hung yi Lee, Lasse Borgholt, Jakob D. Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaloe, Tara N. Sainath, and Shinji Watanabe. Self-supervised speech representation learning: A review. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1179–1210, oct 2022.

[27] Omar Mohamed and Salah A Aly. Arabic speech emotion recognition employing wav2vec2. 0 and hubert based on baved dataset. *arXiv preprint arXiv:2110.04425*, 2021.

[28] Marcyliena H. Morgan. *What are speech communities?*, page 1–17. Key Topics in Linguistic Anthropology. Cambridge University Press, 2014.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

[30] OpenAI. Gpt-4 technical report, 2023.

[31] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[32] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. *ICASSP*, 2015.

[33] Mark A Pitt, Laura Dilley, Keith Johnson, Scott Kiesling, William Raymond, Elizabeth Hume, and Eric Fosler-Lussier. Buckeye corpus of conversational speech (2nd release). *Columbus, OH: Department of Psychology, Ohio State University*, pages 265–270, 2007.

[34] Okko Räsänen. Speech segmentation and clustering methods for a new speech recognition architecture. *helsinki university of technology*, 2007.

[35] 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.

[36] Kåre Sjölander and Jonas Beskow. Wavesurfer-an open source speech tool. In *Sixth International Conference on Spoken Language Processing*, 2000.

[37] Benjamin van Niekerk, Leanne Nortje, and Herman Kamper. Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge, 2020.

[38] Yingzhi Wang, Abdelmoumene Boumadane, and Abdelwahab Heba. A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding. *arXiv preprint arXiv:2111.02735*, 2021.

[39] Raphael Winkelmann and Georg Raess. Introducing a web application for labeling, visualizing speech and correcting derived speech signals. In *LREC*, pages 4129–4133, 2014.

[40] Cheng Yi, Jianzhong Wang, Ning Cheng, Shiyu Zhou, and Bo Xu. Applying wav2vec2. 0 to speech recognition in various low-resource languages. *arXiv preprint arXiv:2012.12121*, 2020.

[41] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

Index

Vita

Luke Vincent Strgar was born in Ellensburg, Washington on 17 April 1996, the son of Dr. Franc Strgar and Wendy Strgar. He received the Bachelor of Arts degree in Computer Science from the University of California Berkeley in 2018. Following the completion of his degree, Luke worked in software engineering, technical consulting, and a variety of scientific research roles. These include, among others, developing computational tools to study motor learning in songbirds and applications of computer vision to analyze medical images for cancer diagnosis and treatment monitoring. Luke submitted an application and was admitted to the University of Texas at Austin's Masters of Science in Computer Science program in 2021. Be began his coursework in August of 2021 and later his thesis research in December of 2021. Upon completion of his MSc., Luke will begin his PhD at Northwestern University in the Center for Robotics and Biosystems.

Address: lvs@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.