TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Fred-Eric Kirsi 182867IAPM

END-TO-END PHONEME SEGMENTATION

Master's thesis

Supervisor: Tanel Alumäe
Senior Researcher

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Fred-Eric Kirsi 182867IAPM

TÄIELIKULT NÄRVIVÕRKUDEL PÕHINEV HÄÄLIKUPIIRIDE LEIDMINE

Magistritöö

Juhendaja: Tanel Alumäe
Senior Researcher

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Fred-Eric Kirsi

04.08.2020

Abstract

Corpus phonetics has become increasingly popular in the linguistic research community. Phonetics is a branch of linguistics that studies the properties of human speech and corpus phonetics is the study of language as expressed in samples of the real-world speech, i.e. the corpus.

For the purpose of corpus phonetics, it is vital to have an accurately phonetically segmented speech corpus, but creating one is time-consuming as it requires a lot of manual labour. It is estimated that to annotate 1 second of speech data it requires 100 to 1000 seconds of manual work [1]. The prospect of reducing the need for such manual labour would be of significant benefit to the scientific community and would enable researchers to spend their time doing more worthwhile tasks.

This thesis proposes a novel end-to-end deep learning architecture for phoneme segmentation based on Soft Pointer Networks, which when given an audio segment and an unaligned phoneme label transcription, produces a well-formed phoneme boundary timestamp. The main feature of this model is that it can learn differentiable positions or indices for queries using a position gradient without losing temporal information. The model was trained and evaluated on the sizable and widely cited manually segmented TIMIT speech corpus. The source code is available at Github [2].

The results on the TIMIT Acoustic-Phonetic Continuous Speech Corpus showed that it outperforms previous baseline models on almost all agreement boundaries. Most notably 57.28% on the 5ms agreement boundary, which is a 9% improvement, and 94.61% on the 20ms agreement boundary, which is a 0.69% improvement.

This thesis is written in English and is 76 pages long, including 12 chapters, 34 figures and 16 tables.

Annotatsioon

Täielikult närvivõrkudel põhinev häälikupiiride leidmine

Korpus-põhise foneetika tähtsus on lingvistilises töös aina kasvanud. Foneetika ehk häälikuõpetus on keeleteaduse haru, mis uurib kõneldava inimkeele omadusi ja korpus-põhine foneetika uurib seda lähtuvalt pärismaailma näidete kogumist ehk korpusest.

Korpus-põhise foneetika põhiliseks uurimisobjektiks on suured hääliku tasemel segmenteeritud kõnekorpused, mille loomine aga nõuab palju aeganõudvat manuaalset tööd. Märgendite kiireks genereerimiseks kasutatakse tihti küll automaatseid tööriistu, kuid nende vähese täpsuse tõttu vajavad need käsitsi parandamist, mille tõttu kulub hinnanguliselt 1 sekundi kõne segmenteerimiseks 100 kuni 1000 sekundit [1]. Seetõttu on oluline, et automaatse segmenteerija leitud piirid oleks võimalikult täpsed, kuna see aitab kokku hoida sadu tunde manuaalset tööd.

Selle probleemi lahendamiseks loodi käesolevas töös häälikupiiride leidmiseks uudne, täielikult närvivõrkudel põhinev Soft Pointer Network arhitektuur, mille sisendiks on kõnelausung ja sellele vastav joondamata ortograafiline foneemide jada, ja väljundiks kõnes olnud täpsed foneemipiirid. Selle mudeli märkimisväärseks omaduseks on võime vastata päringutele diferentseeritavaid positsioone ja indekseid kasutades positsiooni gradienti ja ajalisi seoseid sisendis. Mudeli treenimine ja hindamine tehti laialt teadustöös kasutatava ja mahuka manuaalselt segmenteeritud TIMIT kõnekorpusega. Lähtekood on kättesaadav Github-is [2].

Töö tulemusel loodud mudel saavutas TIMIT Acoustic-Phonetic Continuous Speech Corpus kõnekorpuses varasemate lähtetaseme lahenduste võrdluses paremaid või võrdseid tulemusi kõigis tolerantsi piirides. Kõige olulisemalt 57.28%, mis on 9% rohkem 5ms tolerantsi piirides, ja 94.61%, mis on 0.69% rohkem 20ms tolerantsi piirides.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 76 leheküljel, 12 peatükki, 34 joonist, 16 tabelit.

List of abbreviations and terms

ANN	Artificial neural network
API	Application programming interface
ASR	End-to-end automatic speech recognition
BRNN	Bidirectional recurrent neural networks
CNN	Convolutional neural network
DTW	Dynamic time warping
GRU	Gated recurrent unit
HMM	Hidden Markov Model
IDE	Integrated development environment
L1	Manhattan Distance
L2	Euclidean distance
LSTM	Long short-term memory
MSE	Mean squared error
ReLU	Rectified linear unit
ResNet	Residual neural network
RNN	Recurrent neural network
Tanh	Hyperbolic tangent function
TTS	Text-to-speech
Seq-2-Seq	Sequence to sequence translation RNN
WAV	Waveform Audio File Format
Artificial neural network	A computational model inspired by biological neural networks
Attention mechanism	An alternative method to enable RNN to access information over arbitrary time intervals
Decoder	A recurrent neural network to decode the encoded input sequence into the target sequence.
Convolutional neural network	A variation of ANNs where the computations on a tensor are performed by the means of a shifting filter
Dynamic time warping	Similarity measurement and alignment method between two temporally warped sequences
Encoder	A recurrent neural network to encode the input sequence to a

	intermediate state.
End-to-end automatic speech recognition	A model which jointly learns the pronunciation, acoustic and language model components of a traditional speech recognition system
Logit	A function that maps probabilities [0, 1] to (-inf, inf) with a probability of 0.5 corresponds to a logit of 0
Long short-term memory	A specific architecture of an RNN designed to remember values over arbitrary time intervals
Mel-frequency cepstrum	A representation of a sound spectrogram using the nonlinear Mel scale
Phoneme	A distinct unit of sound specific to language to describe speech
Recurrent neural network	A variation of ANNs where the computations are performed along a directed graph
SmoothL1Loss	A loss function that uses a squared term if the absolute element-wise error falls below 1 and an L1 term otherwise.
Softmax	A normalized exponential function that produces probabilities proportional to the exponentials of the input numbers
Spectrogram	A visual representation of sound frequencies of a signal
Speech corpus	A database of audio and transcriptions for speech
Tensor	A generalization of an algebraic matrix used to describe relationships between data
Time-series segmentation	A process to sequence a time-series into discrete segments in order to describe temporal features of the source
Transcription	A systematic representation of language

Table of contents

List of figures

List of tables

1 Introduction

Corpus phonetics has become an increasingly popular method of linguistic research in the Estonian language research community. Phonetics is a language research area, which studies the acoustic bits that make up our language. As the focus in corpus phonetics for studying the properties of human speech, it is vital to have an accurate phonetically segmented speech corpus.

Phonetically segmented speech corpora have a wide area of applications. For example, they are used to train automatic speech recognition and Text-To-Speech synthesis models. Moreover, in the context of linguistic research, they provide a data-driven approach to derive the abstract rules and relations of languages based on real-world samples [3] [4].

Changes in someone's temporal processing of speech can be a sign of age-related changes which is why investigating these patterns is so vital health-related research. With an accurate determination of phoneme boundaries, it may be even possible to modify speech stimuli in the correct regions and improve dysarthric speech [5].

Phonetically segmented speech corpora are also widely used in Estonian phonetic research. For example, the recent study [6] used manually segmented recordings of the Estonian Foreign Accent Corpus [7] [8] to examine the differences of the pronunciation of Estonian between native Estonian and native Finnish speakers. Another recent study [9] explored the acoustic correlates of secondary stress in Estonian. The research was conducted on recordings of six informants and the segmental boundaries for stressed syllables were manually annotated by two experienced phoneticians [9]. Both works were possible because of the use of a manually segmented speech corpus.

Speech segmentation is carried out by a linguistics expert who marks the phonetic boundaries in the speech utterance. For this, the speech spectrograms, energy, duration of various speech sound events and pitch are considered. It has been observed that no two human annotators are likely to produce the same phonetic boundary for a given

speech utterance [10]. Moreover, creating such a corpus is time-consuming as it requires a lot of manual labour. It is estimated that to annotate 1 second of speech data it requires 100 to 1000 seconds of work [1]. The prospect of reducing the need for such manual labour would be of significant benefit to the scientific community and enable researchers to spend their time doing more worthwhile tasks.

There is active research being done to solve this issue for the Estonian language research community. For this reason, the TalTech and WebMAUS systems have been evaluated for phoneme segmentation against a manual generated expert Estonian speech corpus, where they have been shown to be reliable enough for some language research [1]. One recent research thesis was exploring the topic of deep learning end-to-end speech recognition, but it was reported to underperform when compared to the more traditional HMM-based solutions [11]. Furthermore, as seen in Table 1 in paragraph 3, there has been interest internationally. It stands that there is significant interest in these kinds of solutions.

It can be demonstrated that existing ASR systems, such as the TalTech system and WebMAUS, could annotate a whole audio segment in seconds. Those systems are currently used to generate a quick baseline which requires manual corrections, albeit consuming less time than without using it as a starting point. To reduce human involvement in this process would require a more accurate system.

The key reason why those systems might not perform as well as necessary is that they are not designed for this kind of use-case. Currently, what those models are trained for is not to segment individual phonemes but to do general speech recognition. It stands to reason that an application-specific model would perform significantly better.

There are already plenty phonetically segmented speech corpora with hours of hand-aligned transcriptions available that could be used to train this new system. The key problem to solve is to phonetically segment the already gathered research datasets of audio recording with the known unaligned source text.

The goal of this thesis is to research and propose a better end-to-end deep learning segmentation system, which when given an audio segment and an unaligned phoneme transcription would outperform the current phoneme alignment solutions. For

benchmarking and reproducibility, the system is evaluated on the English TIMIT Acoustic-Phonetic Continuous Speech Corpus.

The first part of this thesis explains the background of sound, speech, and language and gives an overview of the current research and technical solutions. The third chapter gives an overview of the previous research done on this topic and the presented findings are used to create the baseline for this thesis. The fourth chapter defines and explores the dataset used in this thesis. It also provides possible training target encodings that arise from the nature of the data and research problem. The next chapter explains the methodology of model design and validation. The next four chapters propose three distinct modes of solving the problem and present the results for the most promising solution. The final proposed Soft Pointer Network architecture with the best results is then compared to the precursor Pointer Network while explaining the distinct improvements over it. Finally, the thesis brings out possible future research for the proposed end-to-end phoneme segmentation system and summarizes the final findings.

2 Background

The main object of this research is speech recordings with phoneme transcriptions. Phonemes are fundamental units of human speech that represent sounds. Phonemes are what let us distinguish different words. Each language has its phoneme system. [12]

In speech recognition, the input speech is produced by the continuous motion of the vocal tract while the words are constrained by syntax and grammar. In many cases, the alignment between inputs and labels is unknown because of this complex relationship. To produce a phonetically segmented speech corpus from the recited source text and audio recordings, it is vital to be able to produce alignment between the labels and the speech. This requires the use of algorithms able to determine the location of the output labels [13].

2.1 Sound Data

The first part of speech recognition is sound. Sound data must be extracted from physical sound by the means of signal capture devices such as a microphone. The electrical analog signal that a microphone produces must be converted to a digital signal to be used in a computer. This is done with an analog-to-digital converter. The result is a waveform consisting of signal intensity over time.

Figure 1 offers an example of audio waveforms representation of an utterance, where the sound amplitude is plotted for each distinct moment in time. In plot 1 the whole utterance is shown where distinct parts of a sentence can be observed. Whereas in plot 2 the chaotic change of amplitude in a short timeframe shows the complexity of sound.

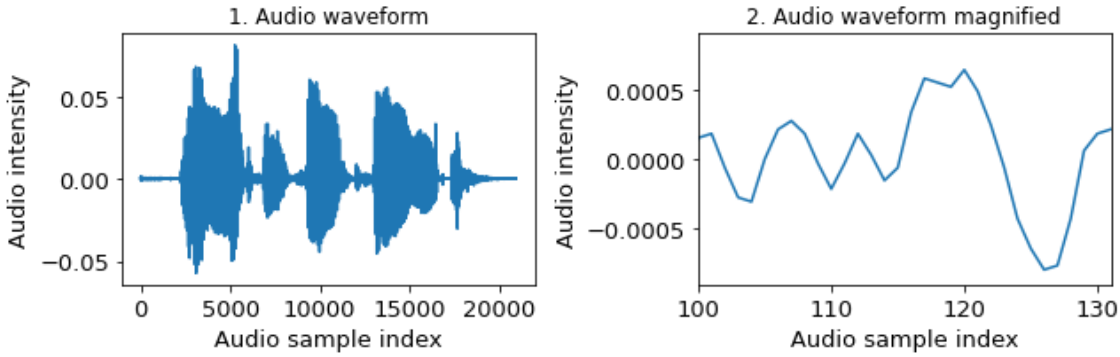

Figure 1 Audio waveform.

While using raw audio has become more common, the research is still largely focused on transformations such as the Mel-scale power spectrum for extracting frequency band features [14] [15]. These raw audio waveforms contain tens of thousands of values per each second of audio. Dealing with this raw audio signal is possible but requires a lot of training. Transformed audio has been used in various fields such as sound and music modelling and speech recognition [16].

The main motivation for using transformations methods such as Mel-scale power spectrum is that audio can be approximated by a stationary process in short timeframes to extract relevant features and reduce bloat. The general idea is to sample the intensities of frequency bands of sound for each 10ms window of the audio recording. This is inspired by the human perception of such signals to approximate the frequency response of a human ear which is not as granular as a recording but has been formed by evolution to focus on the parts that are most important to speech, filtering out all the unheard noise.

The following formulas (1) and (2) shows how to convert between hertz f and Mel m.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \quad (1)$$

$$f = 700\left(10^{m/2595} - 1\right) \quad (2)$$

In Figure 2, the Mel-scale power spectrum transformation has been applied to the previous utterance sample. The total length of the audio segment has been reduced to around 130 feature-rich samples. The change of volume and pitch become more apparent even when viewing the whole audio sample. When tracing the change of sound

frequencies, represented in the transformed audio feature dimension, over the discrete audio sample steps, the unique signature of human speech can be observed.

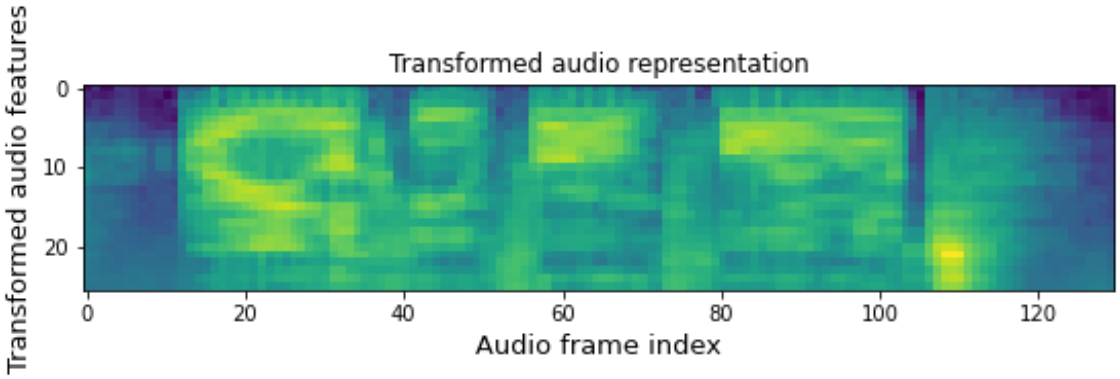

Figure 2 Transformed audio representation

2.2 Measuring Phoneme Alignment

The transcriptions are used to mark meaning to each audio segment. The dataset transcriptions contain the start and end of individual phonemes, as well as pauses and silence tokens. The main objective is to predict these boundary timestamps of the phonemes in the speech recording while knowing only the unaligned phoneme label sequence. This can be solved by a phoneme segmentation model.

For speech recognition and many other real-world sequence labelling tasks, the transcription alignment might be unknown. For example, in the case of collecting speech samples for a corpus, the source text for the speaker is supplied, yet the exact timestamps for the recorded phonetic events are unknown. While in most cases, speech recognition requires the use of algorithms able to determine the location as well as the identity of the output labels, in the segmentation task the main concern is the accurate alignment between inputs and labels [13]. Finding a solution to phoneme segmentation is the main problem in this thesis.

The following Figure 3 shows a real-world case where for an audio segment with known source text “Musta kassi nähes”, two phoneme segmentation tools are compared to a manually annotated phoneme boundaries. The audio has been plotted as a waveform and as a feature-rich spectrogram. As the first step, the phoneme sequence is extracted from the source text. Then the phoneme label sequence is temporally aligned

with the audio segment. Note that the produced alignment must be logically sound even with slight errors in the source text. There are multiple ways of achieving this which are explored in this thesis. The predicted results show a lack of agreement for phoneme and word boundaries between the TTÜ and WebMAUS system [1].

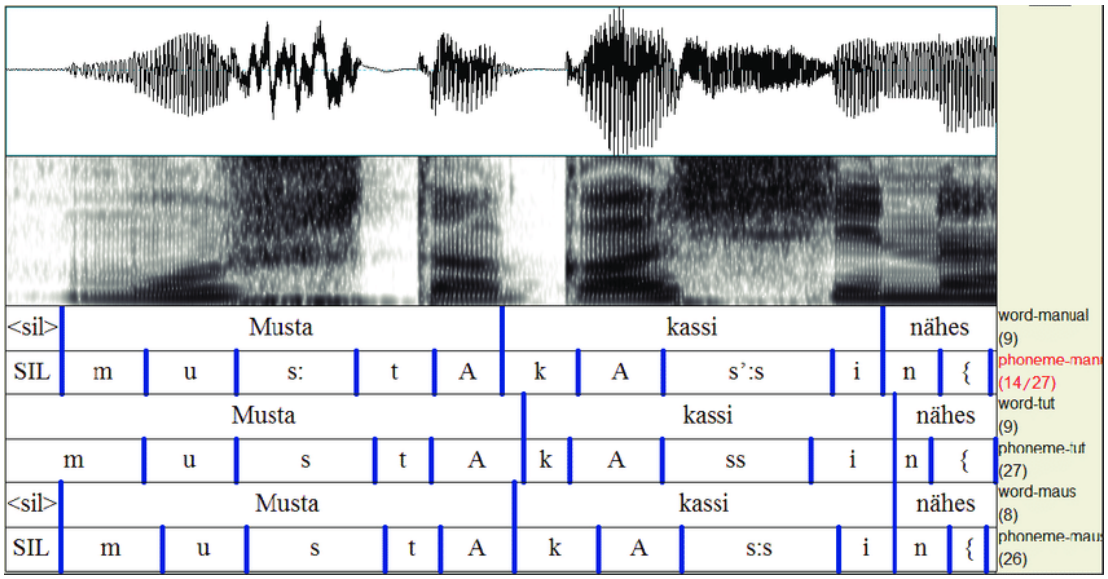

Figure 3 An example of the segmentation problem. The phrase "Musta kassi nähes" has been aligned with the audio with two automatic tools, TTÜ system and WebMAUS, and a human transcriber [1].

The segmentation results are generally reported as the percentage of predicted boundaries positions $C = \{C_1, \ldots, C_n\}$ that reside within a given millisecond threshold B from the manually aligned boundaries $Y = \{Y_1, \ldots, Y_n\}$ as seen from the formula (3). The most common boundary agreement region reported in studies is 20ms but other boundary regions may also be reported. Studies also estimate that segmentation of human experts when compared has an agreement of around 95% within the tolerance threshold of 20ms [4].

$$BoundaryAgreeement(B, C, Y) = \frac{\sum_{i=1}^{n}(1\ if\ |c_i - y_i| < B\ else\ 0)}{n} \times 100\% \quad (3)$$

One of the issues of this method is that it does not account for the magnitude of the errors. Boundary agreement results only considers how many of the errors were below some millisecond threshold while not providing information whether the rest were off by 10ms or 100ms. Furthermore, this measurement has no constraints for the predicted boundaries. The boundaries can be out of order, not in a one-to-one relation with the input phonemes and reside out of bounds of the original audio sequence. This creates an

ambiguous measurement which may not translate well between all published studies. For this reason, this thesis proposes 3 requirements for the segmentation model in paragraph 5.3 and chooses baseline studies with diligence.

2.3 End-to-end speech recognition

Speech recognition is a field of research that develops technologies that enable computers to recognize and translate the spoken language. It is also known as automatic speech recognition (ASR). As a critical part of the research of language itself, it can be used for example in studying the spoken language itself. It can be used to process human speech into queries and commands that personal assistants such as Siri or Cortana can then execute [17]. The field has not only a history of waves of major innovations and academic papers but has also seen an industry-wide backing and adoption.

When viewed in short enough timeframes speech can be approximated by a stationary process. Therefore, most systems, including the ones discussed in this paper, use the speech signal sampled over 10ms as the main input. For this audio sequence, there can be multiple time slices that correspond to a single phoneme creating a unique problem for boundary detection.

A novel emerging field of neural-network-based speech recognition is end-to-end speech recognition. While traditional systems such as Hidden Markov models have been the mainstream speech recognition framework for a long time, they might require the training of an acoustic model, language model, and pronunciation model separately, adding further complexity to the overall architecture [18]. Furthermore, the disadvantage of HMM-based forced alignment systems is that the model does not optimize for phoneme boundaries directly nor are the boundaries explicitly represented in the model [19]. In contrast, the end-to-end systems are much more straightforward to train and provide direct output but require significantly more training data to reach similar or better results. Nevertheless, the advantages encourage further research [20].

One of the first attempts at end-to-end speech recognition was a system based on a combination of the deep bidirectional LSTM recurrent neural network architecture, introduced by Alex Graves of Google DeepMind and Navdeep Jaitly of the University

of Toronto in 2014. The proposed system directly transcribed audio data with text, without requiring an intermediate phonetic representation [21]. A more recent end-to-end system was a large-scale architecture which was presented in 2018 by Google DeepMind achieving 6 times better performance than human experts [22].

2.4 Neural networks

As discussed previously, automatic end-to-end speech recognition uses architectures based on neural networks. Neural networks make few assumptions about the properties of the input data while still being able to predict probabilities for an audio segment. While in the early days of ASR neural networks were rarely successful for continuous recognition tasks as of their limited ability to model temporal dependencies, newer ASR models make use of more advanced architectures to solve this issue.

2.4.1 Deep Neural Network (DNN)

One of the improvements was brought using deep feedforward neural networks. This network combines multiple layers of neurons to model complex relationships. While not effective at sequential tasks, this method allows the network to use raw features to form more descriptive features when compared to hand-crafted methods. For this reason, DNN based networks such as autoencoders have become the go-to models to encode data. Autoencoders are efficient and work in an unsupervised manner. They are great at dimensionality reduction and extracting signals from noise [23] [24] [25].

In the work of Yoshua Bengio, the required qualities for any learning algorithm were described as [24]:

1. The ability to generalize a small training sample dataset of a much larger intricate system.
2. To be able to learn multiple levels of abstraction to model the complex highly varying function.
3. Ability to scale with the increasing amount of data.
4. Ability to work within the constraints of a semi-supervised setting, where not all samples are completely and correctly labelled.
5. Ability to transfer representations across related tasks in multi-task learning.
6. Ability to capture underlying structure in data for unsupervised learning tasks.

In the work [24] mentioned before, these requirements were explored for the topic of deep learning architectures in detail. DNNs were found to be able to learn multiple levels of distributed representations, allowing a rich from of generalization for AI tasks. As deep learning architectures were found to be suitable for such learning tasks, it follows that these qualities also hold for more complex end-to-end neural network models proposed in this thesis.

2.4.2 Recurrent Neural Network (RNN)

In order to work with sequential speech data, a sequential model is required. Recurrent neural networks are designed to take a series of inputs with no predetermined length. It works by applying a DNN on each input element of a sequence, producing a sequence of outputs. RNNs are used by sequence to sequence (Seq2Seq) models to offer the ability to predict label probabilities for each time step.

One of the main improvements for RNNs has been introducing memory mechanisms. They are especially effective in avoiding the problem of vanishing gradients. Vanishing gradient is a problem where a neural network has difficulty backpropagating the error as it becomes vanishingly small. The main reason for this happening in the case of RNNs is the count of created layers that the network must backpropagate caused by the sequential nature of the network. These numerous layers cause repeated partial derivatives with respect to the error function become minuscule enough to be unable to effect a change in the weights of the model. Networks such as the long short-term memory network (LSTM) and the more recent gated recurrent units network (GRU) have proven to be effective in retaining data over long sequences and avoiding this problem.

The long short-term memory network is an RNN that manages long-term dependencies by implementing gates and a memory cell. The memory cell's purpose is to provide a fast lane for information to flow throughout the sequence. Gates are called as such because of their output a weight which when applied to a vector decides how much of its data is passed on. It uses three different gates: input, output and forget. The gates have the same dimension as the hidden state.

The following Figure 4 and formula (4) explain the internal mechanism of an LSTM. The gates are activated by the sigmoid function σ. There are multiple fully connected

layers with weight matrices W and bias b. The first step of a LSTM is for the forget gate f_t to decide what information is worth remembering from the memory cell C_{t-1} and what is forgotten based on the last hidden state h_{t-1} and input x_t. In the next step the input x_t is passed through a $tanh$ activated layer and the new cell value $\tilde{C}_t$ is then combined with the previous cell state C_{t-1} with respect to the input gate value i_t to produce the final cell state C_t. The new cell state is based on the importance of the old and new cell. Finally, the output h_t is formed by squeezing the final cell C_t vector values with the $tanh$ function and weighing them with the output gate o_t value [26].

$$\begin{aligned} f_t &= \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{4}$$

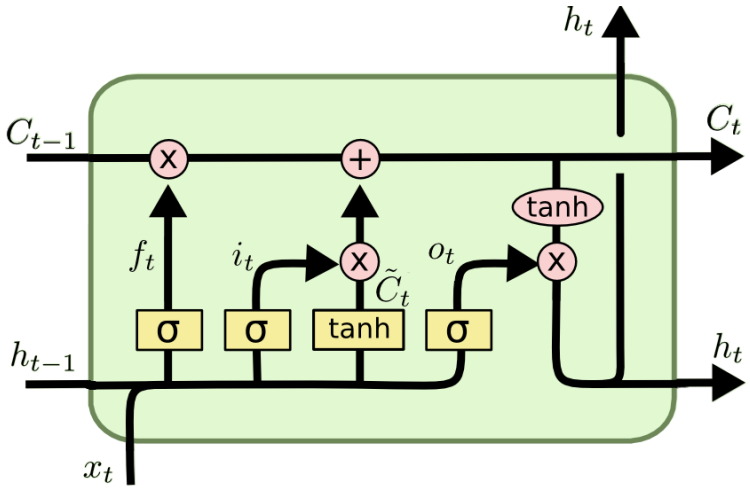

Figure 4 The internal mechanism of an LSTM [26].

A new variation on the LSTM is the Gated Recurrent Unit (GRU). When compared to LSTMs, GRU has two as opposed to three gates and does not use a memory cell. Their reduced complexity has made them a popular alternative for LSTMs.

In Figure 5 and formula (5) the internal mechanism of a GRU is shown. GRUs have two gates, an update gate z_t and a reset gate r_t. The reset gate determines the amount of old state h_{t-1} that is used along with the new input x_t to generate the intermediate state $\tilde{h}_t$.

The update gate z_t produces the final state h_t by using its weight to interpolate the old and intermediate state [26].

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \tilde{h}_t &= tanh(W_h \cdot [r_t * h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{5}$$

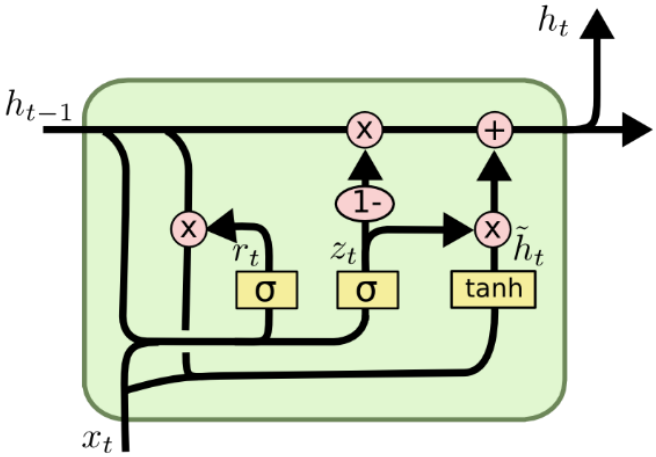

Figure 5 The internal mechanism of a GRU [26].

For the purpose of ASR, the model should make use of the previous and future inputs when dealing with language data to understand the whole context. One such method is to use a bi-directional recurrent neural network (BRNN). As seen in the next Figure 6, it is an abstraction on top of RNNs. The input sequence is processed into two hidden layers, in the forward direction of the input for the first $\vec{h}$ and in the backward direction of the input for the other $\overleftarrow{h}$, and then fed to the common output layer y. By essentially combining two RNNs working in opposite temporal directions, the model provides context information from both directions at every input point [27].

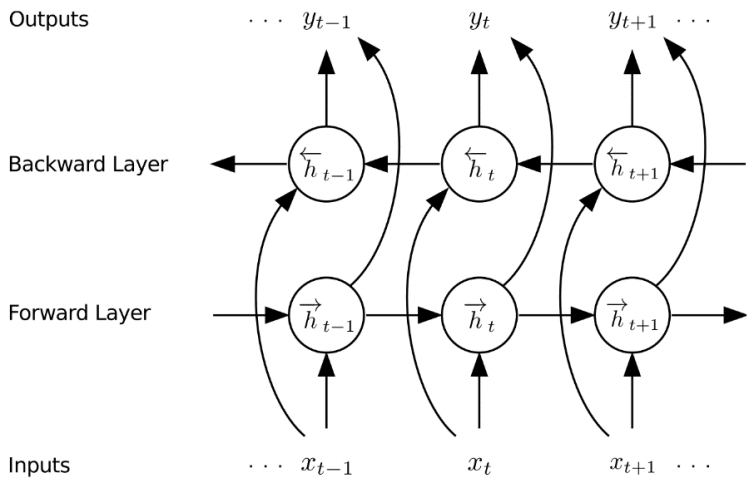

Figure 6 Example of a bi-directional RNN. Image is from Figure 2 in Graves et al., 2013 [28].

Recurrent neural networks can incorporate a variety of different types of networks. This allows the network also to use methods normally found in image-processing. For example, the Seq2Seq model could be augmented with residual blocks (ResNet) to deal with the vanishing gradient problem and with convolutional neural networks (CNN) to allow the model to peek at multiple time steps [29], [30]. Notably, Facebook used the CNN Seq2Seq model for their machine translation model [31].

2.4.3 Sequence tagging

The first attempt at end-to-end automatic speech recognition was using RNNs. The key difference between previous approaches is that this type of neural network uses timing as a variable and is characterized by the problem of there being more observations than there are labels. This creates a unique problem for segmentation tasks as there can be multiple time slices corresponding to a single label.

Sequence tagging is a model that uses a sequence of observations to output a sequence of label probabilities. The problem can be expressed as a mapping task, where the input sequence $X = \{x_1, \dots, x_n\}$ of temporal audio samples needs to be mapped to the output label transcription sequence $Y = \{y_1, \dots, y_m\}$. As mentioned before, the sequence lengths n and m are not necessarily equal or with constant ratio with respect to each other. The output is a likelihood distribution for all possible Y elements for a given element from X. Consequently, training the model is to maximize the conditional probability $p(Y \mid X)$ for the correct answer [32].

For the purposes of audio sequence tagging in this paper, it can be assumed that input and output sequence are of equal length as the training dataset has temporally aligned phoneme transcriptions for training targets. This simplifies the optimization problem for training the sequence tagging model.

To produce a valid label sequence from a sequence of phoneme likelihood distributions, it requires post-processing by an alignment algorithm. Because of this multiple valid label sequences of different alignment can exist. For the purposes of this thesis, the valid order of the label sequence is known, but not aligned with the input audio. This is also the motivation to use dynamic time warping (DTW) instead of beam search. While beam search is commonly used to produce a likely output sequence, DTW is used to find the optimal mapping between two given sequences which enables the system to use the provided label sequence directly. DTW has also been found to be a competitive distance measure for time-series data [33].

The formal definition of DTW shown on the next formula (6). Given the sequences $X = \{x_0, \dots, x_{n-1}\}$ and $Y = \{y_0, \dots, y_{m-1}\}$ of length n and m respectively where $d(x_i, y_j)$ represents the distance between x_i and y_j then the definition shows how one or both of the input sequences are recursive incremented by one step to produce the final alignment distance. The process can also be used to produce an alignment matrix and best alignment path [34].

$$\begin{gathered} DTW(\{\ \},\{\ \}) = 0 \\ DTW(X,\{\ \}) = DTW(\{\ \},Y) = \infty \\ DTW(X,Y) = d(x_0,y_0) + \min \begin{cases} DTW(X, Y[1:-]) \\ DTW(X[1:-], Y) \\ DTW(X[1:-], Y[1:-]) \end{cases} \end{gathered} \tag{6}$$

2.4.4 Attention

One of the shortcomings of sequence tagging is that to infer information over longer sequences they rely on memory mechanisms of the underlying RNN. Memory cannot use future inputs and must remember all previous features. Because of this limitation, it is difficult to make predictions in a Seq-2-Seq model. This constraint can be avoided by using a better mechanism to process information over the whole input sequence,

namely, the attention mechanism. The proposed solution allows the use of the whole input sequence for each output prediction.

While there are multiple attention variants, in this thesis the focus is on Scaled Dot-Product Attention as presented in [35]. The following formula (7) explains the computation of the attention score. The attention score matrix a is calculated using queries (the encoded search terms) and keys (the encoded input features) of dimension d_k. In its simplest form, the score is calculated by taking the dot product between all key vectors in respect to a query, scaling it with the input dimensions $\sqrt{d_k}$ and then applying the softmax function. In practice, the attention score is calculated on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V [35].

$$AttemtionScore(Q,K) = a = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{7}$$

While this also enables this particular attention model to process each output in parallel as the outputs do not depend on the previous queries, this benefit concerns only the attention score calculations. Within an actual machine learning model, the feature encoding might be done using an RNN such as an LSTM and may not enable the whole final model to perform on the inputs in parallel. Moreover, this lack of interdependence might not be exactly desired as presented in this thesis in paragraph 8.4.2.

As a positive side-effect, this process produces attention scores that are very human interpretable. In the following Figure 7, the attention scores are visualized between the source sentence (English) and the generated translation (French) found by the neural machine translation model RNNsearch-50 [36]. The importance of each English word in the whole sentence for each generated French word is expressed as the attention intensity on the horizontal row. It can be seen that the neural machine translation model had to learn that the word order changes when translating "European Economic Area" into "européenne économique zone".

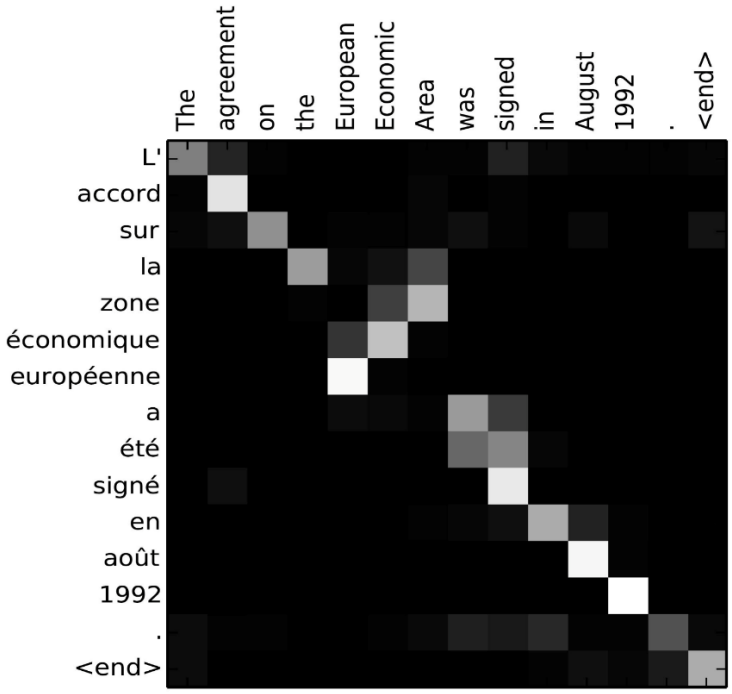

Figure 7 Example of English to French translation, where the computed attention scores exemplify the word order difference between the two languages [36].

2.4.5 Positional encoding

Positional encoding is a synthetically generated gradient which is used to add positional relations to encoded sequences. An example positional encoding is shown in Figure 8. The positional encoding, as seen on the plot, exhibits binary encoding like features in the vertical axis, is a helping mechanism to estimate the distance between the encoding vectors. The high and low-frequency features help the model to estimate the long- and short-term relations.

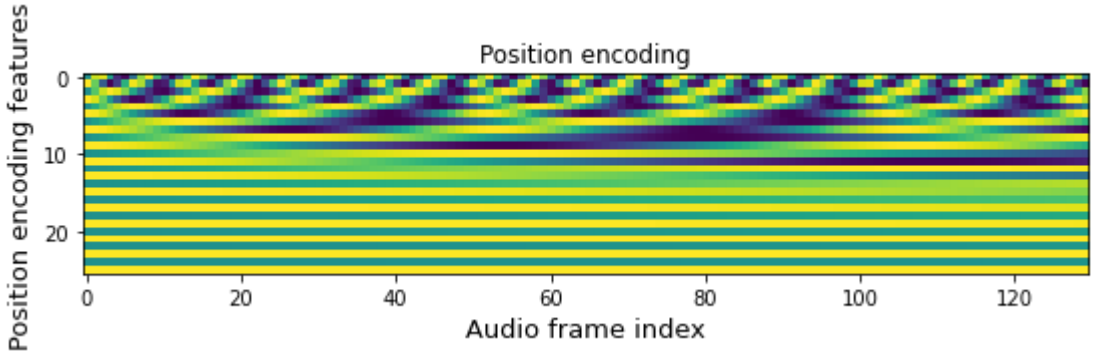

Figure 8 The positional encoding gradient.

Positional encoding $\vec{p}_t$ can also be expressed as pairs of sines and cosines with decreasing frequency ω_t as seen on the next formula (8).

$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1} \tag{8}$$

One of the key features of positional encoding, as seen on Figure 9, is that the individual positional vector on plot 2, which was extracted from the positional encoding gradient on plot 1 at index 57, exhibits the highest dot product with the position vectors close to its index. On plot 3, the further the vectors are to the extracted vector, the lower the dot product activation becomes. Similarly, the scores amplitude is also influenced by the random noise added to the same extracted vector in the same scenario.

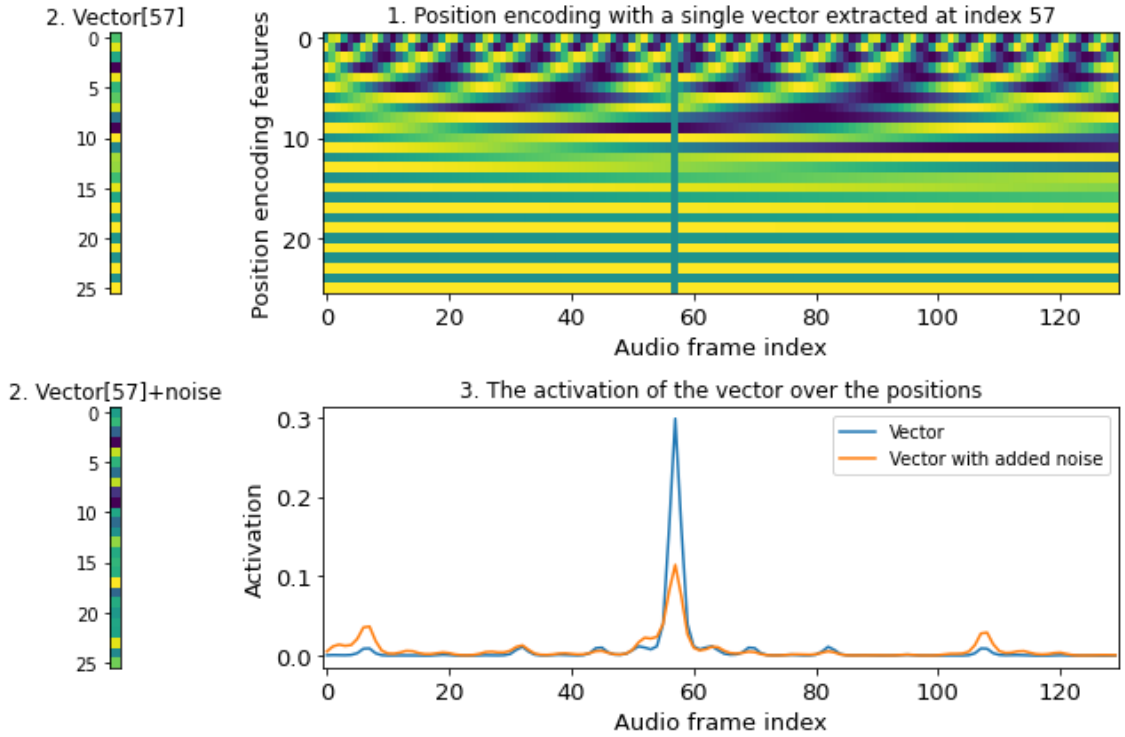

Figure 9 Position encoding activation of a single extracted vector.

The dot product is also how the attention model calculates scores, which is why it

benefits from adding it to the input encoding. The use of positional encoding aids the model to find the best encoding for the queries that would match the phoneme borders on the audio encodings in the same way.

2.4.6 Pointer networks (Ptr-Net)

One of the more recent developments in Seq-2-Seq models were Pointer networks introduced in 2015 [37]. The proposed Ptr-Net is an attention mechanism based neural network architecture that learns to point to positions in an input sequence. It uses attention weights directly to in effect probabilistically output a permutation of the original inputs instead of using the weights to blend the sequence encodings to a context vector at each decoder step.

The main benefit is that it solves a set of combinatorial optimization problems, such as sorting and the travelling salesman problem, which previously could not have been trivially addressed by existing neural network based approaches and that the trained models generalize beyond the maximum lengths that they were trained on. The architecture is meant to handle input sequences of arbitrary order.

For Pointer Networks the most vital part is the attention mechanism from which it is originally derived from. In sequential tasks, the attention mechanism generates a context vector by taking the weighted aggregate of the encoded input tokens. The Pointer network is a natural extension of this method. By substituting the aggregation step with a maximum activation lookup, such as the argmax function, the model produces the identity of the best matching input token for each query.

The following Figure 10 shows an example of the retrieval of original inputs in a new order. The source coordinates $P(x_i, y_i)$ encoded into an intermediate representation vector along with a starting vector labelled $\Leftarrow$. These form the key database. The generator network is initialized with a start token $\Rightarrow$. The generator takes the first token, encodes it into a query vector and finds the key with the highest activation. For the first token the coordinate $P(x_1, y_1)$ is chosen as it has the highest activation as seen from arrow row. For the fourth token, the network chooses to repeat the input coordinate $P(x_1, y_1)$.

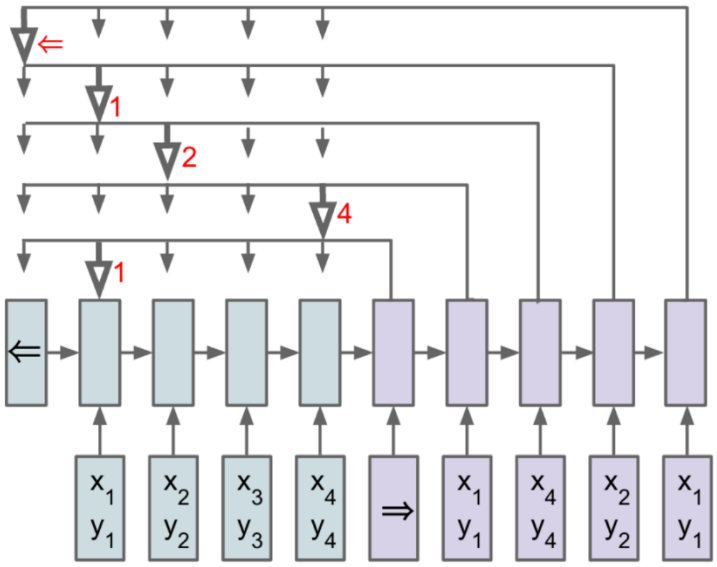

Figure 10 Example of Pointer Network input token reordering [37].

This can be expressed in the formula (9) below. Given the sequence of n input vectors $P = \{P_0, \ldots, P_n\}$ and the sequence of output positions $C = \{C_0, \ldots, C_m\}$ corresponding to an element in input vectors, but in any arbitrary order, then the probability of C_i pointing to P_j is equal to its attention score at a_j^i. A simple strategy to extract the position of C_i is taking argmax of the i-th attention vector $a_i = \{a_0^i, \ldots, a_n^i\}$ as seen on formula (10) [37].

$$p(C_i, P_j) = a_j^i \tag{9}$$

$$C_i = P_j \ \ where \ \ j = \underset{j}{\text{argmax}}(a_j^i) \tag{10}$$

2.4.7 Loss functions

Training a model requires the definition of a loss function which provides a metric to quantify the cost of an event. The purpose of training is to minimize the cost by solving an optimization problem.

Regression models often use mean squared error loss (MSE loss) as seen on formula (11). While it uses L2 regularization, some models also use L1 or a combination of both. One such function is SmoothL1Loss as seen on formula (12). It switches from L1 to L2 regularization when the absolute element-wise error falls below 1. The main benefit is that it is less sensitive to outliers than MSE loss.

$$\text{MSELoss}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{11}$$

$$\text{SmoothL1Loss}\,(x, y) = \frac{1}{n}\sum_{i=1}^{n} z_i \quad where \;\; z_i = \begin{cases} 0.5(x_i - y_i)^2, & if\ |x_i - y_i| < 1 \\ |x_i - y_i|, & otherwise \end{cases} \tag{12}$$

For classification models cross entropy loss (CrossEntropyLoss) is commonly used as seen on formula (13). Cross entropy loss builds upon the idea of entropy from information theory and uses it to measure the performance of models which output probability values for output labels.

$$\text{CrossEntropyLoss}(x, y) = -\sum_{i=1}^{n} x_i \log y_i \tag{13}$$

In some cases where the model is used to produce an embedding such as positional encodings, cosine loss (CosineLoss) is used to measuring whether two vectors are dissimilar. As seen on formula (14) it is implemented using the cosine distance.

$$\text{CosineLoss}(x, y) = 1 - \frac{x \cdot y}{\|x\|\|y\|} \tag{14}$$

2.5 Frameworks and tools

Grand ideas mean nothing if there is no means to build them. The author must acknowledge the great tools which enable the research of speech recognition. For this thesis, 4 core tools are key factors for success: Python, PyTorch, Colab, and GPU acceleration.

Python is currently the go-to programming language for machine learning. Its main role in this thesis is to be a streamlined scripting API for existing machine learning tools. It's effortless to use syntax and wide library support has led to its for industry-wide adoption in many fields. Most importantly it is free to use.

PyTorch is an open source end-to-end machine learning library that is designed to allow easy model building. What distinguishes PyTorch from many of the other neural network libraries is the ability to create a dynamic computation graph while keeping a clean and streamlined API. It is also consistent with the well-established computation

library NumPy. These benefits are arguably a key reason for its increasing popularity in the deep learning community.

One of the great breakthroughs, that has enabled more ambitious machine learning models, has been the use of graphical processing units for accelerating the training of models. A Graphics Processing Unit (GPU) is a microprocessing chip designed to handle Graphics in computing environments. The benefit comes from their highly parallelized architecture which was initially meant for 3d rendering but could also be applied for other computation tasks. Using GPU acceleration significantly reduced the training time during the development of this thesis.

While an IDE might not create an impression of critical importance, Google's Collaboratory was of key importance to the present work. It is much more than a text editor as it combines documentation, version control, and cloud computing. This is one of the best examples of the democratization of technology in recent times. Colab gives the tools and hardware for free to use and is meant for sharing reproducible solutions.

3 Related work

Determining phoneme borders is very important for language research which is why it also is a recurring topic in the research community. There have been multiple papers on the topic of phoneme segmentation on the TIMIT corpus. Their results and methodology are a great resource for the research needed to develop an end-to-end phoneme segmentation model.

When searching for related papers, the main criteria was to find baseline results and methodology, which would be suitable end-to-end model building. The processing of the source recordings and transcriptions should have been documented to be reproducible. If possible, the TIMIT corpus is preferred as the source corpus. The results should be verifiably compliant with the requirements discussed in detail in the following paragraph 5.3 and have been reported for wide range boundaries to ensure the stability of the system in all regions as the evaluation method does not account well for the magnitude of segmentation errors as explained in paragraph 2.2.

The following Table 1 shows some of the works on the topic of phoneme boundary detection. The table lists the corpus and the results of each reference.

Table 1 A selection of related works on the topic of automatic phoneme segmentation.

Year	Title	Results
2019	Phoneme boundary detection from speech: A rule based approach [38]	TIMIT Corpus, IIIT Hyderabad Marathi database and IIIT Hyderabad Hindi database with 10ms boundaries of 95.40%, 96.87% and 96.12%
2018	Automatic Phonetic Segmentation and Pronunciation Detection with Various Approaches of Acoustic Modeling [39]	TIMIT Corpus with 20ms boundary 83.84% and 30 ms 93.12%
2017	Automatic Phonetic Segmentation Using the Kaldi Toolkit [40]	Czech language corpus with 69.4% at 10ms, 89.8% at 20ms and 96.5% at 30ms
2016	Phoneme Boundary Detection using Deep Bidirectional LSTMs [41]	TIMIT with 96.3% and 97.5%, and BUCKEYE 96.5% and 97.5% for boundaries 10ms and 20ms.
2014	Highly Accurate Phonetic Segmentation Using Boundary Correction Models and System Fusion [19]	TIMIT Corpus with 96.8% agreement on 20ms boundary
2014	Phonetic segmentation of speech signal using local singularity analysis [42]	TIMIT with 53.16% agreement on 10 ms.
2012	Towards automatic phonetic segmentation for TTS [43]	TIMIT with 85.2%, 94.9%, 97.8%, 98.9% and proprietary male US English TTS corpus 84.6%, 95.4%, 98.5%, 99.4%, for boundaries of 10ms, 20ms, 30ms, 40ms
2010	Automatic Speech Segmentation Based on Acoustical Clustering [4]	TIMIT with 46.1%, 71.2%, 81.7%, 86.8%, 92.6% and Albayzin 50.2%, 74.8%, 85.6%, 90.7%, 95.5% for boundaries of 5ms, 10ms, 15ms, 20ms, 30 ms

While these works have provided great results, their methodology and availability of different boundary regions limit their usability for this thesis. The most outstanding result is from the recent work of Franke et al. (2016), with 96.5% and 97.6% for the tolerance range of 10ms and 20ms respectively [41], and Ramteke et al. (2019), with 95.40% for the tolerance range of 10ms [38]. For both works, the methodology for data preparation and score calculation was found to be too ambiguous for the purposes and requirements of this thesis. Both works reported accuracy, precision, recall and F1-score for quality measures as both works cause insertions and deletions of borders. It is

evident that these works do not satisfy the segmentation requirements proposed in this thesis in paragraph 5.3 and the correct detection rate is lower. Another concern for the results shown is that the results exceed 95% within 10ms, which is considerably higher than the reported agreement limit between human experts within 20ms [4], and might hint at very dataset-specific adjustments which might not transfer well to a more generic approach.

The following paragraphs introduce two papers on the topic of phoneme segmentation which satisfy the requirements. Both research papers were done using the TIMIT Acoustic-Phonetic Continuous Speech Corpus and were HMM based. The findings of these works build upon each other and they use a common data preparation methodology. Even though the results are not the best-known results among the related work in this area of research, the provided wide range of boundary agreement values and thorough documentation of the data preparation made them good a baseline for this thesis.

3.1 Speaker-independent phoneme alignment using transition-dependent states

The research done by John-Paul Hosom in Oregon Health & Science University proposes a baseline HMM-based forced-alignment system for phoneme segmentation and explores several modifications to improve the results.

The proposed modifications compared to a baseline HMM-based system are:

- The feature set includes four additional energy-based features
- The system uses probabilities of a state transition given that observation
- The system computes the probabilities of distinctive phonetic features and combines them with context-dependent phoneme probabilities

The final agreement results for the proposed model were presented as phoneme boundary agreement percentages within thresholds from 5 to 100 milliseconds as presented in Table 2. The performance of the baseline system on the test partition of the TIMIT corpus is 91.48% within 20ms, and the performance of the proposed system on this corpus is 93.36% within 20ms [5].

Table 2 Reported agreement percentages of baseline 1.

Milliseconds	<5	<10	<15	<20	<25	<30	<35	<40	<45	<50
Agreement (%)	48.28	79.3	89.49	93.36	95.38	96.74	97.61	98.22	98.62	98.92
Milliseconds	<55	<60	<65	<70	<75	<80	<85	<90	<95	<100
Agreement (%)	99.13	99.32	99.45	99.57	99.64	99.7	99.75	99.78	99.81	99.83

3.2 Automatic Phonetic Segmentation using Boundary Models

The joint effort of the University of Pennsylvania, Microsoft Research, and SRI International investigated the possibility to improve automatic phoneme segmentation using the existing HMM framework. The paper explores the benefits of using phone boundary models, precise phonetic segmentation for training HMMs, and the difference between context-dependent and context-independent phone models in terms of forced alignment performance.

The main findings of the paper are:

- A combination of special one-state phone boundary models and monophone HMMs can significantly improve forced alignment accuracy.
- HMM-based forced alignment systems benefit from using precise phonetic segmentation.
- Context-dependent phone models are not better than context-independent models when combined with phone boundary models.

The final agreement results for the final model were presented within thresholds from 10 to 50 milliseconds as seen in Table 3. While the results improve the results over the previous work in almost all boundaries, the 10ms agreement is significantly lower and there is no information on over 50ms boundary agreements. The proposed final system achieves a 93.92% agreement within 20ms compared to manual segmentation on the TIMIT corpus [44].

Table 3 Reported agreement percentages of baseline 2.

Milliseconds	<10	<20	<30	<40	<50
Word boundaries differentiated	77.44	93.92	97.43	98.78	99.35
Word boundaries not differentiated	77.53	93.84	97.39	98.75	99.35

4 Research dataset

The dataset used for evaluating and comparing the results to related work is the American English TIMIT Acoustic-Phonetic Continuous Speech Corpus. The TIMIT corpus transcriptions have been hand-verified. It has been compiled specifically for doing acoustic-phonetic studies and the development of speech recognition systems. It has a pre-specified balanced test and training subsets [45].

The TIMIT corpus includes time-aligned orthographic, phonetic, and word transcriptions as well as a 16-bit, 16kHz speech waveform file for each utterance [45]. The phonemic transcription boundaries are marked by the waveform file sample index. The defining aspect of this dataset is its temporally long audio segment combined with non-trivially numerous output labels.

The phonetic dataset, which is the focus of this thesis, consists of 61 phonemes. The original phoneme counts are displayed in Table 4 below. There is high variation in the phoneme representation in the dataset. This is especially prevalent for the silence token "/h#/".

Table 4 Listing of all phonemes and their occurrence counts in the original dataset.

h#	12600	ix	11587	s	10114
iy	9663	n	9569	r	9064
tcl	8978	l	8157	kcl	7823
ih	6760	dcl	6585	k	6488
t	5899	m	5429	ae	5404
eh	5293	z	5046	ax	4956
q	4834	d	4793	axr	4790
w	4379	aa	4197	ao	4096
dh	3879	dx	3649	pcl	3609
p	3545	ay	3242	ah	3185
f	3128	ey	3088	b	3067
sh	3034	gcl	3031	ow	2913
er	2846	g	2772	v	2704
bcl	2685	el	1294	ch	1081
th	1018	ux	2488	y	2349
epi	2000	ng	1744	jh	1581
hv	1523	pau	1343	nx	1331
hh	1313	en	974	oy	947
aw	945	uh	756	uw	725
ax-h	493	zh	225	em	171
eng	43				

4.1 Input preparation

The dataset was first filtered to remove any dialect calibration sentences to match the work in paragraph 3.1 and 3.2. The resulting dataset consisted of 3,696 utterances for training and 1,344 utterances for testing. The boundaries between two pauses, including stop closures, were excluded from evaluation. As detailed in [5], the syllabic phonemes “/em/”, “/en/”, “/eng/”, and “/el/” were mapped to their non-syllabic counterparts “/m/”,” /n/”, “/ng/”, and “/l/”. The glottal closure symbol “/q/” was merged with the neighbouring voiced phonemes. Pauses less than 20ms with duration were merged into previous phonemes. The resulting dataset had 54 phonemes, matching the previous work. The result of these changes is shown in the following Table 5.

Table 5 Listing of all 54 phonemes and their occurrence counts in the modified dataset.

pau	11991	ix	9871	n	9251
s	8348	tcl	7703	l	7577
r	6529	iy	6436	ih	5686
kcl	5492	t	5312	dcl	5286
k	4997	m	4972	z	4918
ax	4856	eh	4524	pcl	3609
p	3541	axr	3483	dh	3272
d	3254	f	3126	ah	3126
w	3119	aa	3102	ey	3073
ae	3064	b	3039	v	2704
bcl	2665	ao	2626	ay	2620
dx	2498	er	2347	ow	2253
ux	1838	gcl	1782	sh	1777
g	1640	ng	1598	y	1371
jh	1308	hh	1281	ch	1079
th	1004	aw	944	hv	940
nx	925	uh	715	uw	686
ax-h	471	oy	431	zh	222

The source transcription is split into a phoneme label sequence, which is used as the input, and the matching boundary values, which is used as the evaluation target. There were 139703 boundaries in the training set and 50,579 boundaries in the test set for evaluation. Additionally, a development evaluation set was split out of the remaining training dataset, that was used for stopping the training to avoid overfitting the data.

The raw waveform audio input is impractical for training RNN models. Therefore, the audio from the TIMIT dataset is converted into Mel-spectrogram representation. This allows the model to have a more compact yet human-like audio representation as explained in 2.1. The audio conversion was augmented using varying steps size and window size configurations to either enhance training speed or model accuracy. This is done because some speech recognition methods, such as sequence tagging models, may significantly benefit from decreasing the step size as they tie their output resolution directly to the time resolution of the audio input. This will be illustrated in the following chapter 6.3 **"**Window step length manipulation**"** in depth. Furthermore, changing the

sampling logic is also compatible with transfer learning as a pre-trained model adjusts within only 2 to 5 epochs to its original accuracy according to the author's empirical observations during the model training for this thesis. Changing the output dimension size was tested, but it was not similarly beneficial as it required significantly more re-training to reach the initial accuracy of the previously trained weights.

4.2 Dataset overview

Firstly, the main input is audio data. As seen in plot 1 in Figure 11, half of the audio segments are longer than 2284ms and then on plot 2, the average duration is very close to 3000 milliseconds. The minimum duration was 910ms and a maximum of 7780ms. There is a high variation in lengths of the audio segments which will have to be considered when batch processing the data.

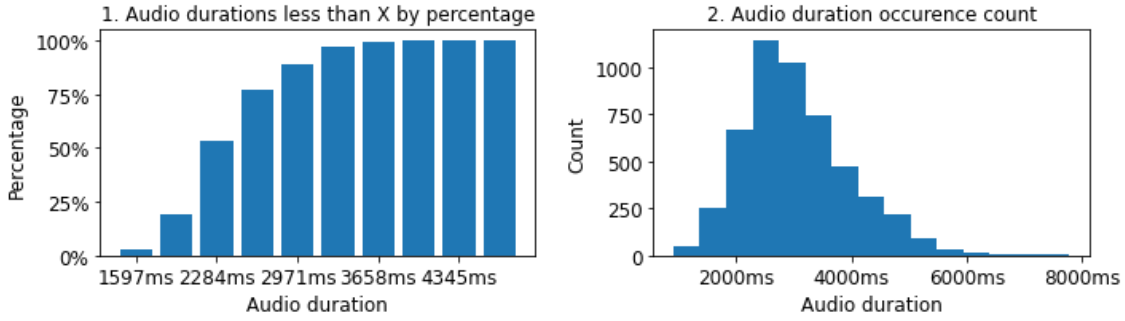

Figure 11 Audio duration distribution.

Secondly, the accompanying phoneme transcription dataset follows with similar high variation. Plot 1 of Figure 12 shows that half of the utterance samples contain more than 25 phonemes and plot 2 that they contain at minimum 9, at most 73, and on average 36.75 phonemes.

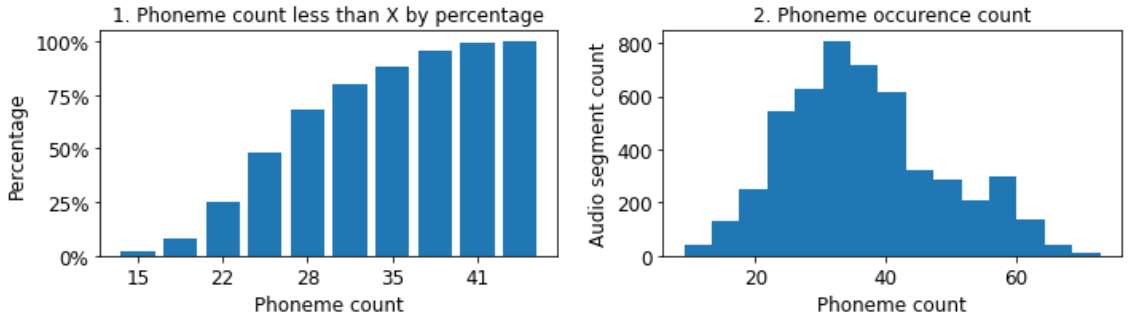

Figure 12 Phonemes per audio segment.

Furthermore, there is almost a ten times difference in the average length of the longest and shortest phoneme. This can be seen in Table 6 where the per phoneme average duration is shown. The longest phoneme in the dataset is silence.

Table 6 Listing of all phonemes and their average duration.

b	18ms	d	25ms	g	27ms
dx	29ms	nx	29ms	ax-h	34ms
dh	36ms	p	44ms	t	49ms
ax	50ms	k	52ms	ix	53ms
gcl	55ms	y	56ms	dcl	57ms
r	58ms	n	58ms	tcl	59ms
v	61ms	kcl	61ms	jh	62ms
w	63ms	ng	64ms	bcl	65ms
hh	65ms	m	66ms	l	68ms
hv	71ms	pcl	72ms	uh	77ms
axr	79ms	ih	80ms	zh	83ms
z	86ms	ch	87ms	th	91ms
ah	92ms	eh	95ms	iy	97ms
ux	99ms	f	103ms	uw	109ms
s	114ms	sh	119ms	er	123ms
ao	127ms	aa	127ms	ey	131ms
ow	132ms	ae	139ms	ay	159ms
aw	169ms	oy	175ms	pau	178ms

4.3 Model inputs and training targets

The goal of this thesis is to predict the timestamps of the phoneme borders. The model has two inputs, the speech audio and the phonemes sequence of the source utterance. From these, the model must learn a representation that allows us to extract the phoneme boundaries of each phoneme on the audio segment.

The first input is audio. The audio segments have been pre-processed beforehand into a more feature dense representation from the raw audio format by using the Mel-scale power spectrum transformation as discussed in paragraph 2.1.

The second input for the model is the transcription label sequence. The phonemes labels provide context information, about which and in what order the phonemes occur in the

utterance without any timestamps. While in the TIMIT dataset the ground truth transcriptions also include the start and end times for each phoneme, those are used for evaluating the result and not the model input.

Depending on the chosen model type and training method there are multiple possible training targets. One option would be to train the sequence tagging model to predict phoneme occurrence probabilities for each audio time step and use a one-hot encoded ground truth phoneme sequence. For the second alternative, the model could also be trained to predict the duration for each phoneme in the input label sequence. For this, it would be necessary to form an array of durations in matching order to the label sequence. Lastly, if viable, the model could predict the actual millisecond timestamps of the border. The position could be expressed as a simple scalar or multi-dimensional position vector.

The example audio segment in the first plot of Figure 13 has 130 encoded frames and each frame represents a period of 10ms. The transcription for this audio segment lists 24 phonemes. For each audio segment frame, there is a one-hot encoded phoneme target in the second plot denoting its occurrence. In the third included plot, each phoneme from the transcription is matched with the end position of its occurrence and in the final plot with the duration of the phoneme.

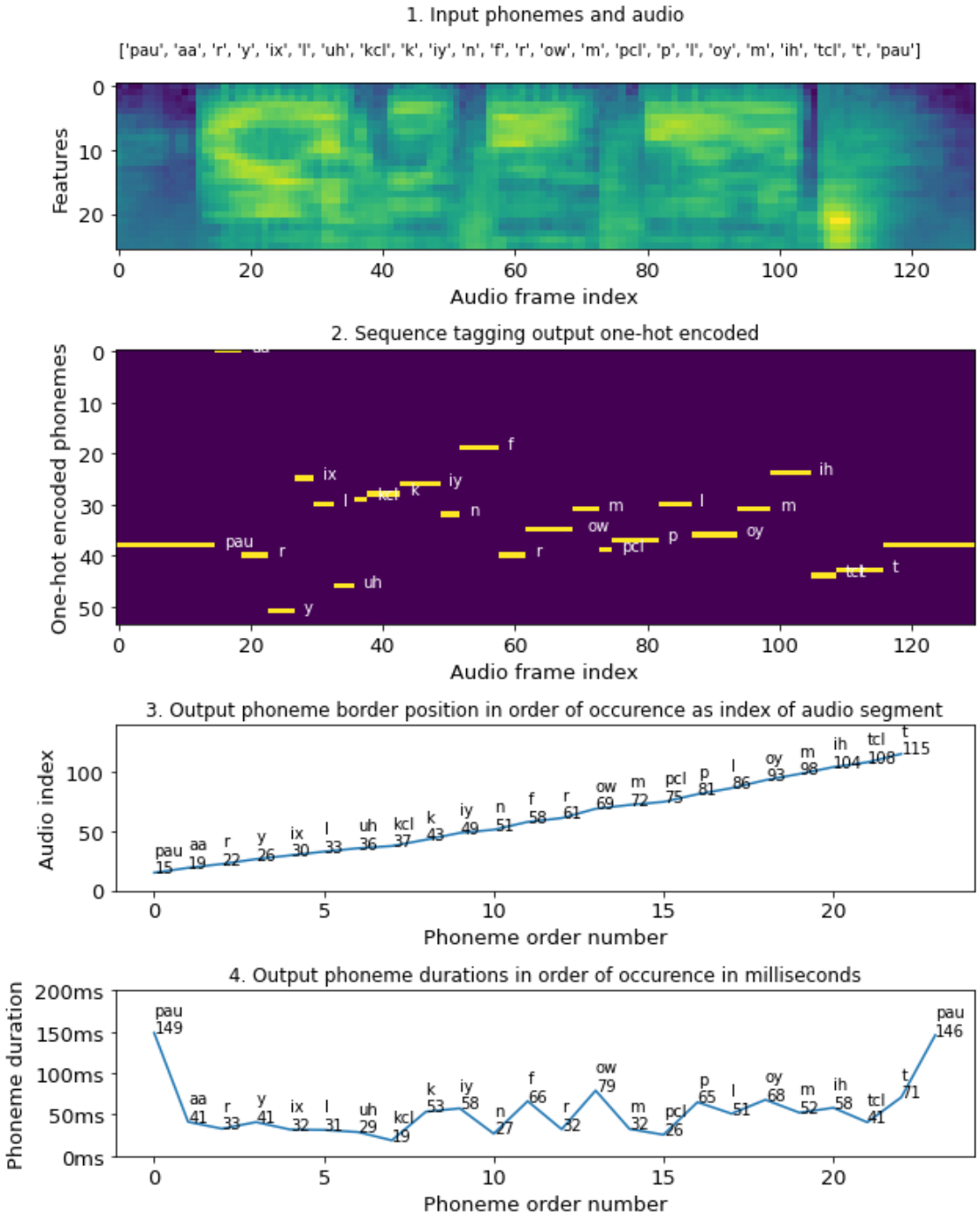

Figure 13 Comparing different possible output targets for the audio segment.

5 Methodology Specification

The goal of this paper is to propose an end-to-end automatic speech recognition model that would improve the accuracy of phoneme segmentation. For this purpose, there are three distinct prediction modes considered:

- Phoneme detection
- Phoneme duration
- Phoneme position

The first solution is audio-focused and will try to predict phoneme occurrence for each temporal audio frame and use the provided phoneme label transcription to extract the most likely phoneme occurrence sequence. The second model is phoneme focused and will try to predict each phoneme's duration based on the sequence of the phoneme labels while using the audio as extra context. The final solution creates a mapping between the two input sources to identify the position of each phoneme label in the audio segment.

The different approaches will be evaluated and compared to the previously set milestones in baselines Table 2 and Table 3. The final findings will conclude whether an end-to-end ASR solution can effectively replace the current HMM-based solutions.

The evaluation measurements are done on the TIMIT dataset using the phoneme border timestamps in milliseconds. The conclusion will be presented as a phoneme boundary agreement percentage within 5 to 100ms.

5.1 Generic model components

Some parameters and processing units were in all the following models without much if any modification. The hidden size for the RNN encoders and decoders was 256 in all models. Each RNN was using a dropout of 0.3. For activation, ReLU and tanh were used and to convert logits into final probability results a softmax activation layer was used. The general notation is that the intermediate tensor sizes are shown in grey boxes with B being the batch size, T_a being the audio length, L_p the phoneme label sequence length. The following 2 figures show the feature generic structure of the encoder and the decoder used in the following experiments.

The first example in Figure 14 is the generic encoder used in the experiments. All the inputs were first passed through a layer of batch normalization before passing them to a bi-directional RNN. Then resulting hidden state was resized to the desired output dimension with a fully connected layer and positional encoding was added when configured.

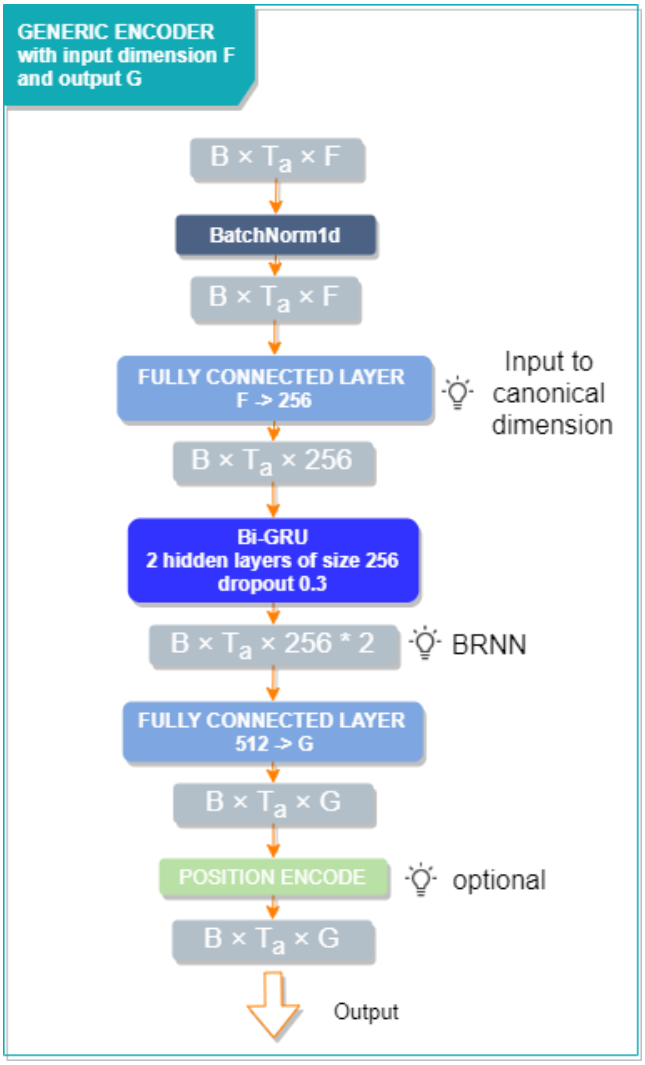

Figure 14 Generic encoder model architecture graph.

The next example in Figure 15 shows the Seq^n-2-Seq decoder used in the following experiments. Chaining multiple of these components together allows the combination of multiple different length sequence into one output sequence. The decoder accepts an input sequence of length N_x and a context sequence of length N_y to produce an output sequence of length N_x by using the attention scores produced between them. The motivation for this architecture and working principle is further explained in paragraph 5.1.

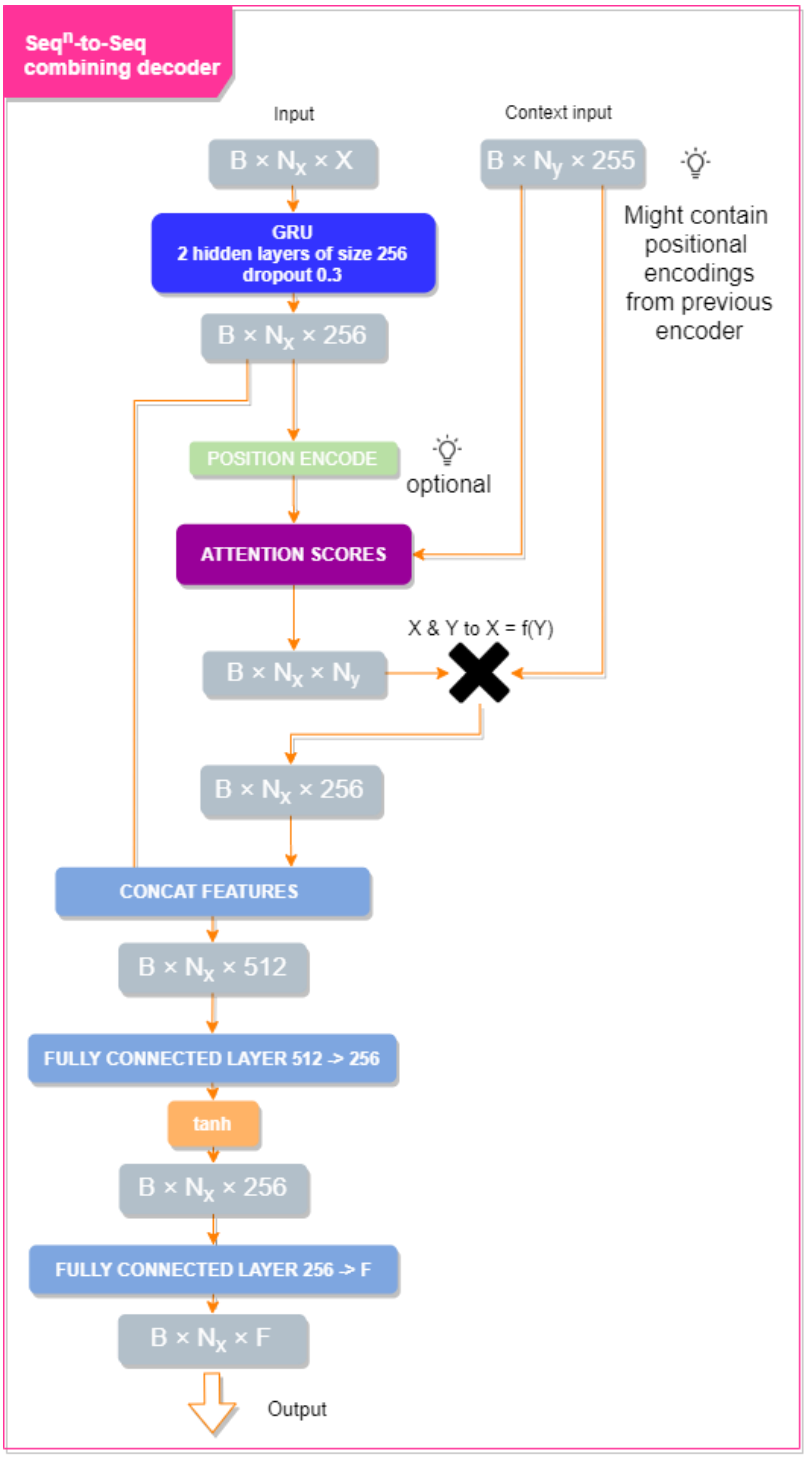

Figure 15 Generic Seq^n-2-Seq decoder model architecture graph.

5.2 Training

As mentioned previously, the American English TIMIT Acoustic-Phonetic Continuous Speech Corpus has a pre-specified balanced test and training subsets. This ensured that

the training and testing dataset could be imported independently for training or evaluation purposes without the possibility of cross-contamination. The training data was split into training and evaluation sets for early stopping.

To avoid overfitting, the models were trained on the training data until training loss stopped decreasing or until the evaluation loss started to regress. While some of the early model evaluations were without early stopping, the presented final results for all models were using early stopping and were retrained from a clean state without the use of previously trained weights to exclude the possibility of cross-contamination and ensure the reproducibility of results.

As the model training was done using Colab's cloud environment over extended periods, the trained weights of the model were exported to Google Drive cloud storage to be evaluated at a later time or inside another instance of the cloud computing environment.

5.3 Requirements

There will be multiple model training architectures evaluated and compared. Each model must meet certain requirements:

1. Predicted border count must equal to the transcription border count.
    - It is impossible to evaluate the results without this.
2. Predicted border timestamps must be bounded in the audio time frame.
    - No border should exceed the audio length or preceded the start.
3. Predicted border timestamps must be monotonically growing.
    - No border can precede the previous border.

The requirements create a constraint for the model to give sensible output even when the source text for the phoneme input might be suboptimal. Furthermore, these restraints allow for less ambiguous comparison between previous and future works.

5.4 Milestones

The proposed model's significance can be quantified by comparing it to the established solutions shown in the previous related work section. While there is a special emphasis on the 20ms agreement percentage, the improvements will also be quantified over the other tolerance regions.

The main milestones to improve over would be:

- Simple HMM 89% agreement within 20ms [46]
    - The baseline model should be reasonably close to this result.
- Human 93% agreement within 20ms
    - Human labellers have an average agreement of 93% within 20ms on various corpora and an agreement of 93.49% within 20ms on TIMIT [44].
    - The proposed model should reach or exceed this milestone to be a viable alternative for manual labour.
- Baseline 1 as boundaries from 5 to 100ms as presented in Speaker-independent phoneme alignment using transition-dependent states [5]
    - The results should be compared to the multiple boundary intervals proposed in the paper. The regions where the model exceeds and underperforms should be inspected.
    - Special notice should be on the 93.36% agreement within 20ms as well as the 79.30% within 10ms
    - It is also the only source to present an agreement (48.28%) within 5ms
- Baseline 2 as boundaries from 10 to 50ms as presented in Automatic Phonetic Segmentation using Boundary Models [44]
    - The proposed system achieves a 93.92% agreement within 20ms and 77.53% agreement within 10ms.

6 Experiments with audio sequence tagging

The first way to solve the phoneme segmentation problem is to train a model to produce phoneme probabilities for each audio observation. The most straightforward method to produce a valid label sequence is then to pick the phoneme with the highest probability for each observation. Then the borders can be inferred from the transition of the predicted phoneme in the sequence.

Simple model architecture for this is shown in Figure 16. The audio input is passed through an RNN encoder, which is further expanded in paragraph 5.1, and the result used to produce a softmax phoneme probability vector for each timestep T_a. The was trained with cross entropy loss.

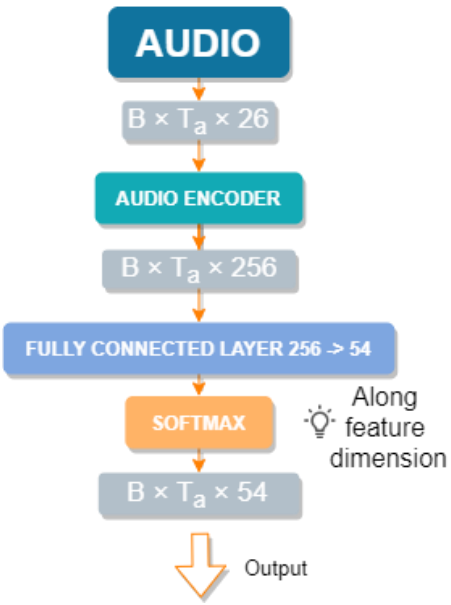

Figure 16 Simple LSTM sequence tagging model architecture graph.

The baseline results in Figure 17 were produced by a simple LSTM sequence tagging model. The first plot is the audio input, the second plot the predicted phoneme occurrence probabilities and the third plot the most likely phoneme. The final plot is the ground truth. The average phoneme classification error for this model was 30.06% which is abysmally bad this model does not satisfy requirement 1 for border count equality.

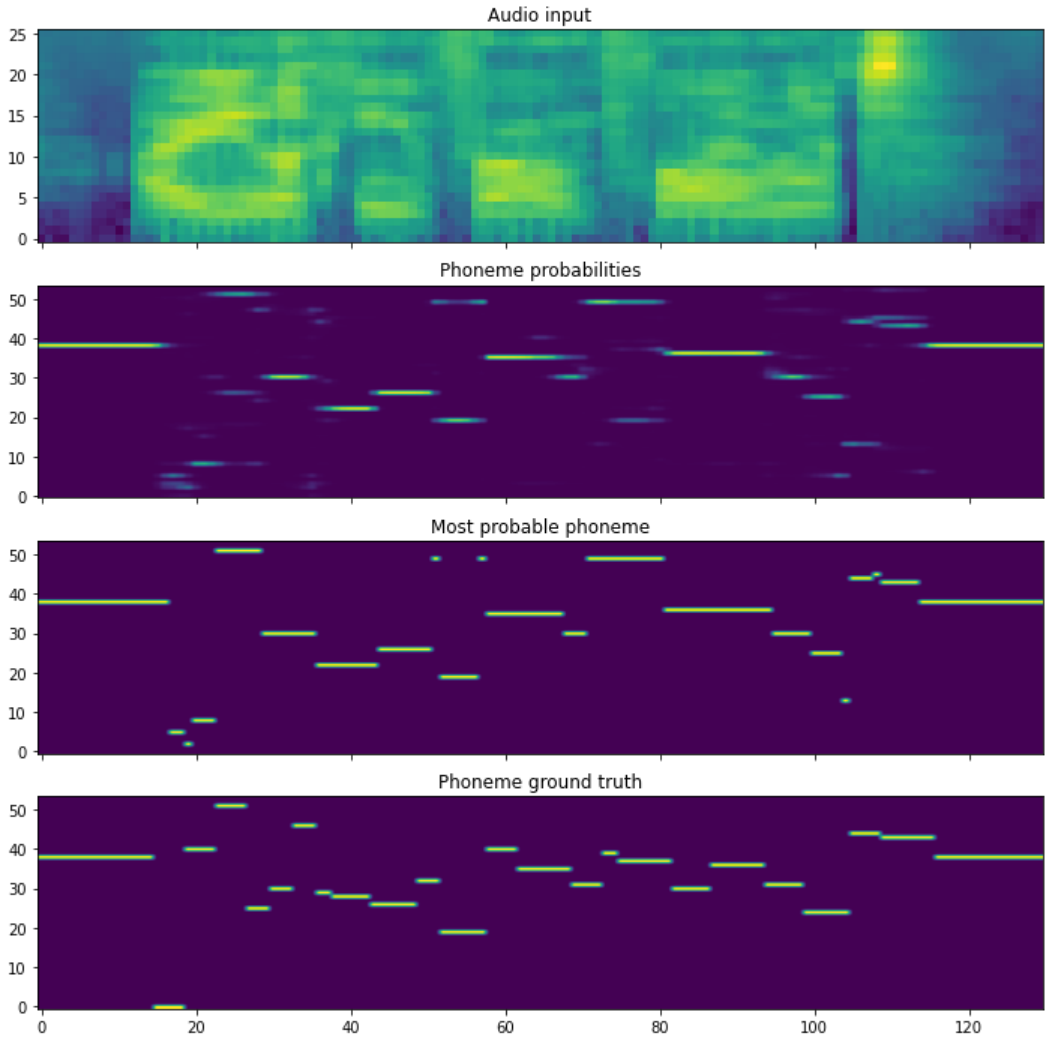

Figure 17 Baseline simple LSTM sequence tagging model output.

6.1 Alignment with the label transcription

While this would be fine for a perfectly trained model it does not necessarily satisfy the first requirement to generate the exact number of borders. In order to fix this the final step is to produce a valid occurrence sequence by post-processing the probability output with the input phoneme label sequence by an alignment algorithm, such as the dynamic time warping algorithm. The input label sequence does not have any information about the duration or position of the phonemes, but it still helps to enforce the correct occurrence order and count.

When using an alignment algorithm there are multiple possible label sequences, differing in durations or order, that could be produced from the probabilities. Wrong sequences can be eliminated by choosing the one with the best alignment with the label input sequence. This ensures that the requirements 1, 2 and 3 are met. If the produced

probabilities were sensible then also the duration accuracy follows. The improvements can be seen in the following Figure 18 below. The predicted probabilities on plot 1 are the same as in Figure 17, but in comparison, the post-processed occurrence sequence shows a higher resemblance with the ground truth occurrence map.

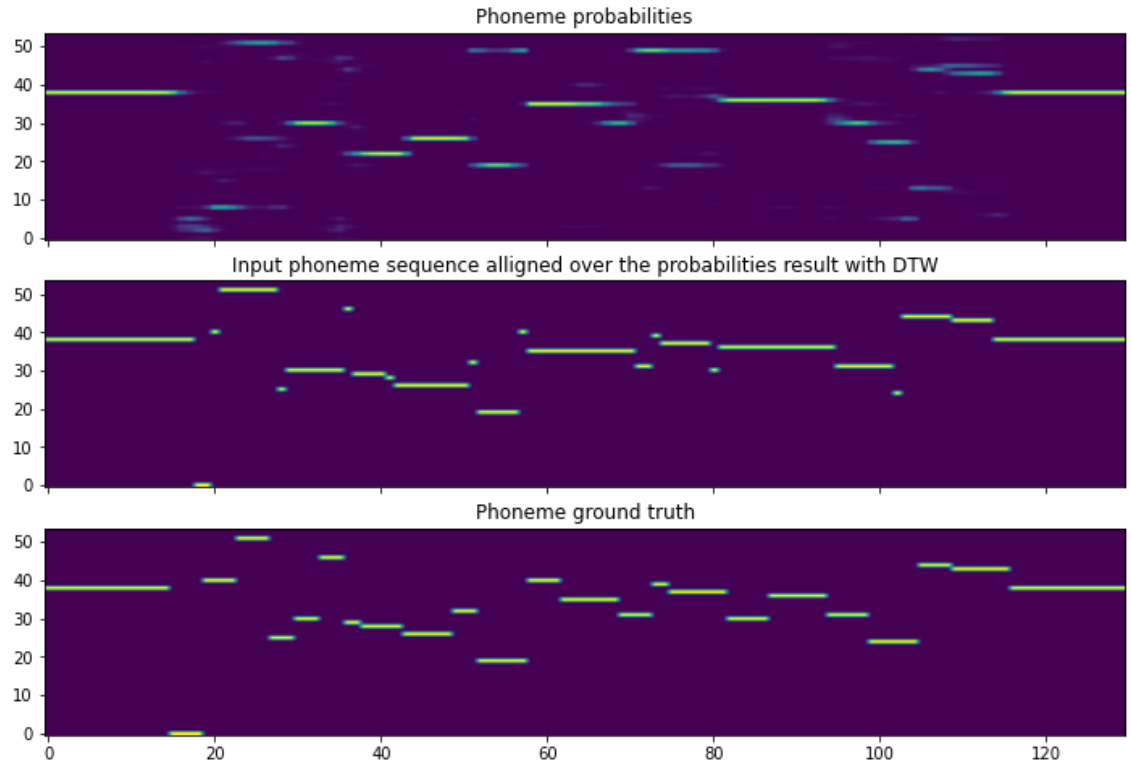

Figure 18 The improvements from post-processing the result probabilities with the transcription phoneme sequence making sequence tagging output compliant with requirements 1, 2 and 3.

As the requirements 1 to 3 are met the phoneme border agreements can be calculated on the alignment. The results are presented in Table 7. The 20ms agreement for this model is 82.6%. None of the boundary agreement percentages exceeded the set milestones.

Table 7 Initial phoneme boundary agreement percentages for sequence tagging.

Milliseconds	<5	<10	<15	<20	<25	<30	<35	<40	<45	<50
Agreement (%)	36.2	62.0	75.2	82.4	86.6	89.3	91.2	92.5	93.6	94.4
Milliseconds	<55	<60	<65	<70	<75	<80	<85	<90	<95	<100
Agreement (%)	95.0	95.5	95.9	96.3	96.6	96.9	97.1	97.3	97.5	97.7

6.2 Adding label transcription context with attention

One of the shortcomings of the proposed sequence tagging model is that it fails to incorporate the phoneme label transcription into the prediction. Previously, the label transcriptions are only considered at the post-processing step and not training. It stands to reason that the model could make more informed predictions if it could peek at the label transcription. Whether it is to get an idea about the correct order of phonemes or know what phonemes even expect in the audio input.

The method for combining the two variable-length input sequences, the audio and label transcription encodings, is to use the attention mechanism. It works by calculating the attention scores for the audio on the label transcription and then effectively taking the weighted average of the phoneme label encodings each audio frame. This produces a sequence of the same length and dimensionality as the audio input. The example of producing an audio enriched label transcription encoding is seen in Figure 19. Instead of calling this network a Seq-2-Seq network, it would be more appropriate to call it Seq^n-2-Seq or n-Seq-2-Seq.

In Figure 19, the attention scores in plot 3 were calculated for each one-hot encoded phoneme labels on plot 2 over the audio on plot 1. Note that the visualization of features shown on plots 1 and 2 are illustrative and do not reflect the actual intermediate encoding used by the attention mechanism. The third phoneme's attention scores over the audio frames are highlighted in plot 4. The multiplication between attention weights and audio encoder features can be seen on plot 5 and the final audio enriched phoneme vector on plot 6.

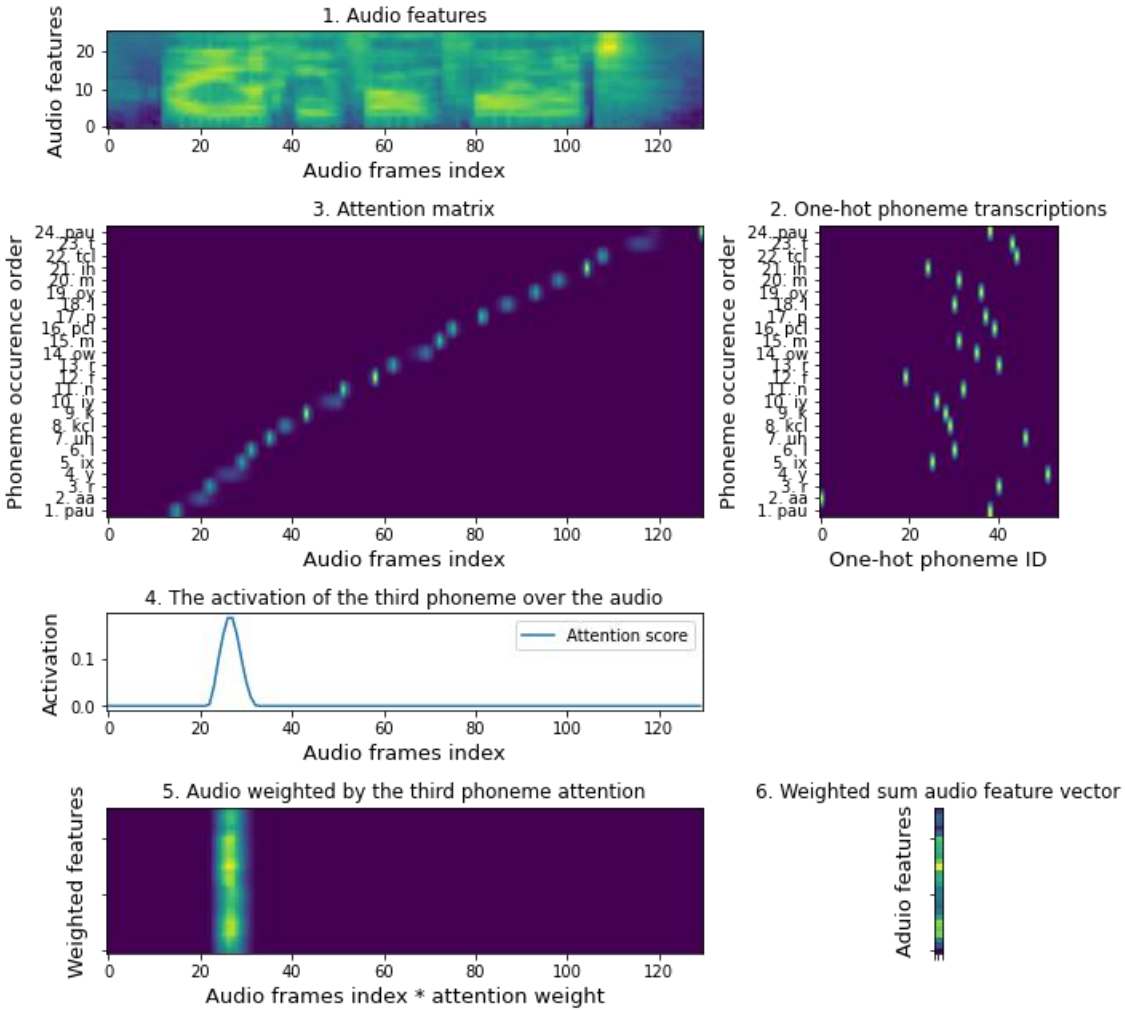

Figure 19 The process to gather the audio context for the third phoneme.

The proposed model architecture, as shown in Figure 20, produces the combination of the input sequences in two stages. Once for enriching audio with the phoneme sequence context. Then once more for combining a phoneme enriched audio encoding with an alternative audio encoding. This allowed the model to have multiple sequence combination possibilities, some of which were tested and made available as a configuration option on the model. The model could be trained in stages for each output configuration with cross entropy loss. The first stage allowed the first audio encoder to be trained and evaluated. The second stage made use of the previously trained encodings of the first encoder and combined them with phoneme sequence context. The last decoder combined the phoneme enriched audio encodings with pristine audio encodings to increase the adaptability of the model even more. The encoder and decoder blocks are further expanded in paragraph 5.1.

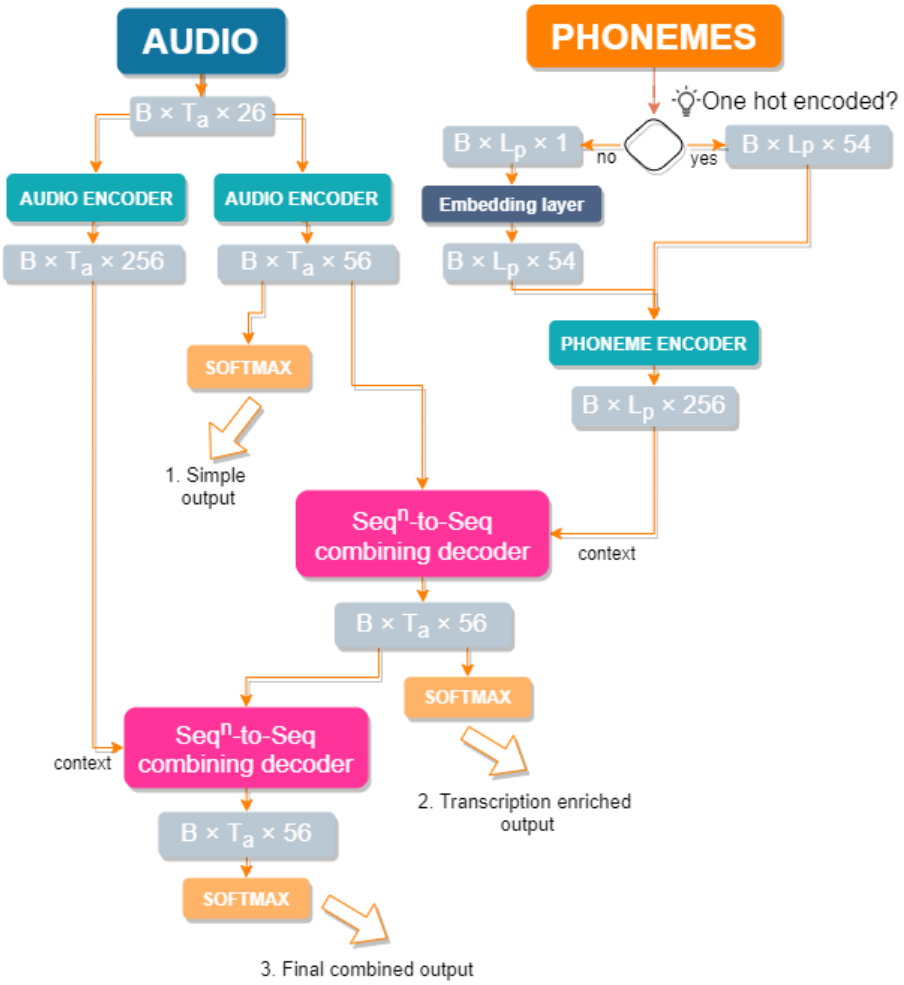

Figure 20 Enhanced sequence tagging model architecture which combines information from audio and phoneme context and allows multiple sections to output and be trained individually.

With these architecture changes, the final border agreement evaluation results in Table 8 show that the model has exceeded the requirement for the first milestone. There are significant improvements when compared to the previous results in Table 7. The mean border error was 8.57ms with a maximum difference of 851.44ms.

Table 8 Context improved sequence tagging phoneme boundary agreement percentages.

Milliseconds	<5	<10	<15	<20	<25	<30	<35	<40	<45	<50
Agreement (%)	47.4	76.7	88.1	93.0	95.6	97.1	98.0	98.6	99.0	99.2
Milliseconds	<55	<60	<65	<70	<75	<80	<85	<90	<95	<100
Agreement (%)	99.4	99.5	99.6	99.7	99.8	99.8	99.8	99.8	99.8	99.8

6.3 Window step length manipulation

The impact of the chosen audio transformation window step size for pre-processing will impact the accuracy of the model significantly. This is partly because the border itself is reported in the resolution of WAV file sampling, but the transformed audio input is significantly shortened by producing a vector only for every 15ms segments. As a result, the sequence tagging model limited by the transformation window step size, as it must output the labels with the same discrete intervals.

By reducing the step size, also the discrete intervals become shorter. With the tighter coverage, the shorter segments have a higher chance to be closer to the actual border. Alternatively, by increasing the step size, the model can benefit from faster training with less granular audio. Fewer frames also allow the use of bigger batch sizes as of the lower memory footprint. The trained model progress can easily be transferred between datasets with different step sizes as input features are produced the same way, nevertheless.

The following comparison chart in Figure 21 illustrates the benefits. The default step size of the audio transformation is 10ms which allowed batch-processing 64 inputs at the same time with the current hardware's memory limits. The following tests were done using 5ms, 10ms and 15ms as the step size. The 5ms step size increased the total count of discrete temporal audio feature vectors when compared 10ms, which led to the reduced batch size of 16 to meet the same memory constraints. The increased step size of 15ms over 10ms produced less discrete temporal audio feature vectors and allowed the batching 128 inputs within the same memory constraints.

In the first plot in Figure 21, the measurements were taken after transfer training the phoneme segmentation model which was previously trained on the 10ms step size transformed audio input. Notably, when training on any of the step sizes the model reached peak accuracy within 10 epochs, regardless of which step size the model was trained on before. The plot compares the boundary agreement percentages, which are the key metrics for model accuracy as explained in paragraph 5.4, between the three augmentations. In the second plot in Figure 21, the emphasis was on the total time for training over an epoch. The results were produced by taking the average epoch training time over 20 epochs.

Plot 1 exemplifies the benefit of using smaller audio transformation step size for sub 20ms agreement boundaries. There was a marginal improvement with 5ms step size but a significant decrease of accuracy when increasing the step size to 15ms. That said, the results on Plot 2 show a remarkable reduction of training time per epoch using 15ms step size when compared to 10ms and 5ms counterparts. It can be concluded that using larger audio transformation window step size and in effect, less discrete temporal audio feature vectors is a viable strategy for early model testing.

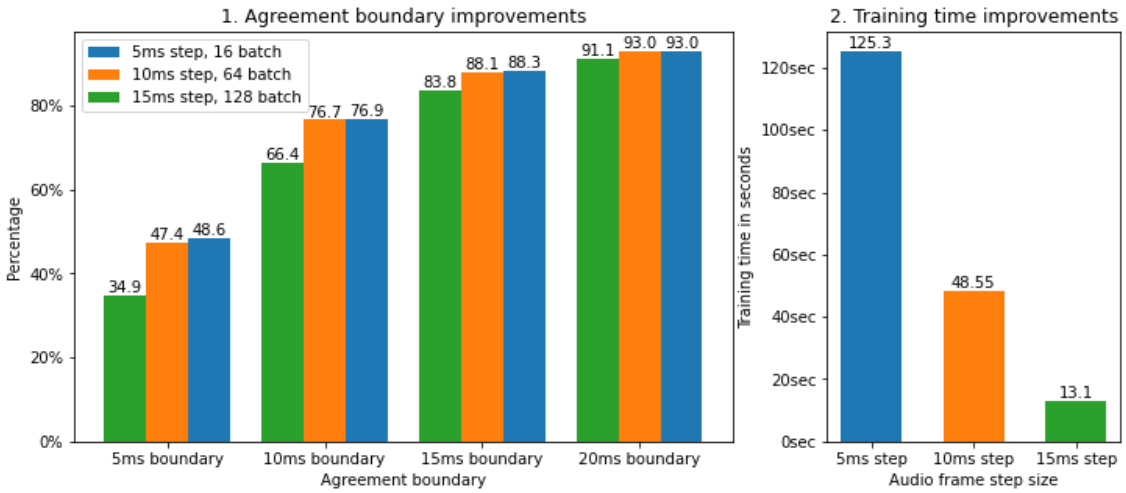

Figure 21 Audio window step size augmentation improvements for accuracy and performance.

6.4 Results

The model has met all requirements and produced 93.06% agreement within 20ms which exceeds the simple HMM 89% agreement and human 93% agreement. The results are presented in Table 9 in comparison to baseline 1 and 2. The model has exceeded the scores of only some baselines. The model has achieved better agreements than baseline 1 on almost all 5 to 45ms boundaries. It has not exceeded any of the baseline 2 results. The mean difference from the ground truth border is 7.50ms with a maximum of 758.12ms.

Table 9 Sequence tagging agreement improvements compared to baseline results.

	Baseline 1	**Baseline 2**	**Sequence tagging**
< 5ms	48.28		**48.90**
< 10ms	**79.30**	77.53	77.37
< 15ms	**89.49**		88.53
< 20ms	93.36	**93.92**	93.06
< 25ms	95.38		**95.42**
< 30ms	96.74	**97.43**	96.82
< 35ms	97.61		**97.72**
< 40ms	98.22	**98.78**	98.28
< 45ms	98.62		**98.67**
< 50ms	98.92	**99.35**	98.89
< 55ms	99.13		**99.07**
< 60ms	**99.32**		99.20
< 65ms	**99.45**		99.31
< 70ms	**99.57**		99.39
< 75ms	**99.64**		99.44
< 80ms	**99.70**		99.48
< 85ms	**99.75**		99.53
< 90ms	**99.78**		99.58
< 95ms	**99.81**		99.62
< 100ms	**99.83**		99.65

As shown in Figure 22, the final sequence tagging model has produced a much cleaner detection probability sequence than the baseline shown in Figure 18. For the comparison, the same audio segment was used for both figures to produce the phoneme occurrence probabilities. The improvement can be attributed to improved phoneme predictions. Not only are the occurrence streaks similar in duration but the predictions also have significantly fewer misclassifications.

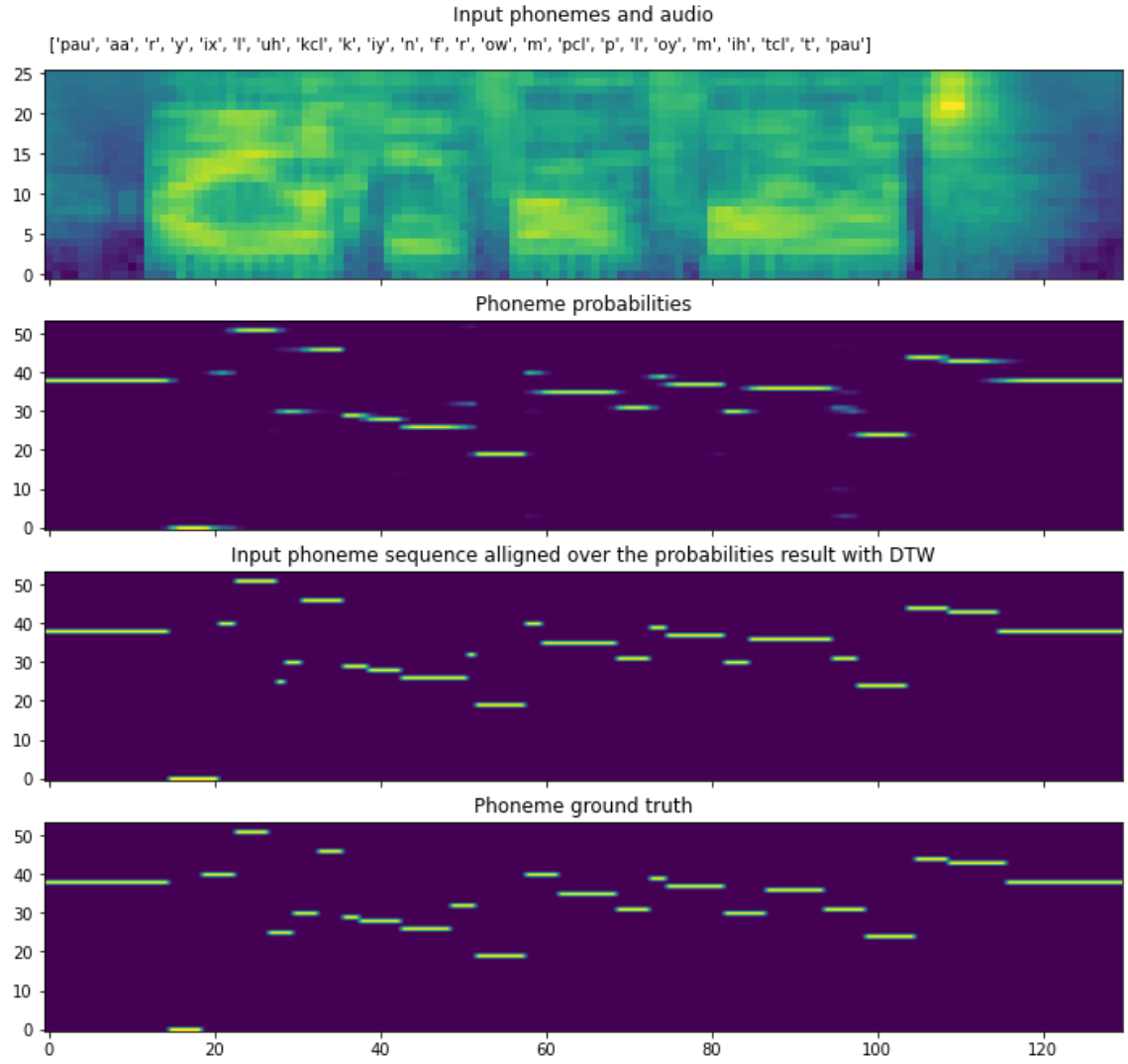

Figure 22 Improved sequence tagging probabilities for the previous example shown in Figure 18.

7 Experiments with duration prediction

The main goal of this thesis is to predict the boundary timestamps for the input phoneme labels. This leads to the wonder, whether it is better to focus more on the label transcription and instead use the audio-only as context. The transcription phoneme count is directly related to the boundary count when excluding the last end-of-file border, which makes the use of it satisfy the first requirement. Instead of using the model as a classifier, it is possible to use it for regression, predicting the duration for each phoneme. The boundary timestamps follow from this by taking the cumulative sum of durations. Furthermore, when only allowing non-negative durations this would also satisfy the third requirement.

The proposed model architecture, as shown in Figure 23, produces the combination of the input sequences in two separate stages. Once for enriching phoneme sequence with audio context and once more for combining the phoneme sequence with an alternative phoneme sequence encoding. This allowed the model to have multiple sequence combination possibilities similarly to Figure 20 which were available as a configuration option on the model and were used for training. The model could be trained in stages for each output configuration with MSE loss. The first stage allowed the firs encoder to be trained, the second and third stage allowed the previously learned encoding to be combined with context and then trained and the final stage concatenated the combined sequences into a single output. The model outputs a single duration value for each of the phonemes which are rectified with a ReLU to only allow positive duration values. The encoder and decoder blocks are further expanded in paragraph 5.1.

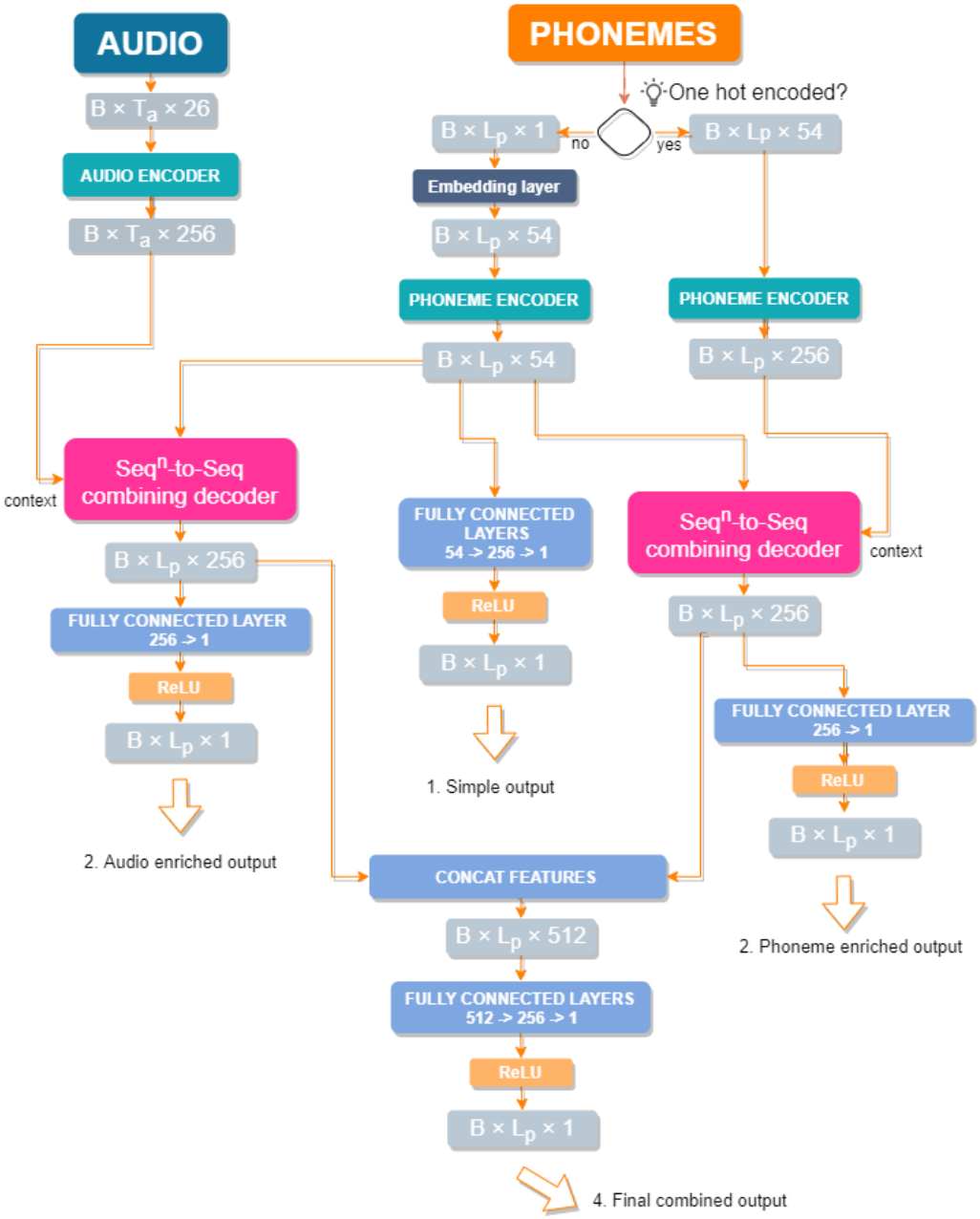

Figure 23 Duration prediction model architecture which combines information from audio and phoneme context and allows multiple sections to output and be trained individually.

The results for duration prediction and border agreement are shown in Table 10. The predicted durations are on average off by 19.59ms with a maximum of 1616.81ms. The duration model did not yield great results for boundary agreements as the errors for border positions are cumulative from the duration mistakes. Each deviation of the previously predicted duration would shift the subsequent borders by that amount.

Table 10 The errors for the duration and border predictions for the duration model.

	Duration agreement (%)	**Border agreement (%)**
< 5ms	20.37	2.68
< 10ms	39.06	5.23
< 15ms	54.51	7.58
< 20ms	66.53	9.69
< 25ms	75.40	11.78
< 30ms	81.89	13.65
< 35ms	86.43	15.46
< 40ms	89.76	17.16
< 45ms	92.07	18.74
< 50ms	93.74	20.29
< 55ms	95.00	21.74
< 60ms	96.00	23.13
< 65ms	96.69	24.50
< 70ms	97.24	25.82
< 75ms	97.65	27.18
< 80ms	97.98	28.41
< 85ms	98.27	29.58
< 90ms	98.47	30.80
< 95ms	98.67	31.88
< 100ms	98.82	33.02

7.1 Duration scaled label sequence alignment for sequence tagging

Given that the duration prediction by itself is not very useful, an alternative way is to instead use it to improve the sequence tagging model's post-processing alignment. This is achieved by using the predicted duration to give the input transcription label sequence scale before aligning them on the sequence tagging output. The easiest method is to iterate the durations and for each 10ms that the duration covers a corresponding one-hot encoded phoneme vector is added into the end of the sequence. In other words, if the predicted duration for the first phoneme is 90ms then 9 one-hot encoded vectors are added to the sequence. The result is a pseudo sequence tagging output mapping with

similar sequence length, as seen in plot 3 in Figure 24, provided the durations are normalized to add up to the audio segment total duration. Doing this forces the post processing alignment to avoid needlessly long or short phoneme sequences as it has access to the recommended phoneme duration ratios.

A visual example of how well the duration scaled labels can match the ground truth is shown in Figure 24 below. The chosen audio segment is visualized on plot 1. The predicted and actual duration are compared in plot 2. Indeed, the predictions are off by about 20ms on average as reported before. The third plot shows each phoneme label repeated with respect to the predicted duration which results in a rough estimate of the desired ground truth phoneme occurrence sequence as seen on the last plot.

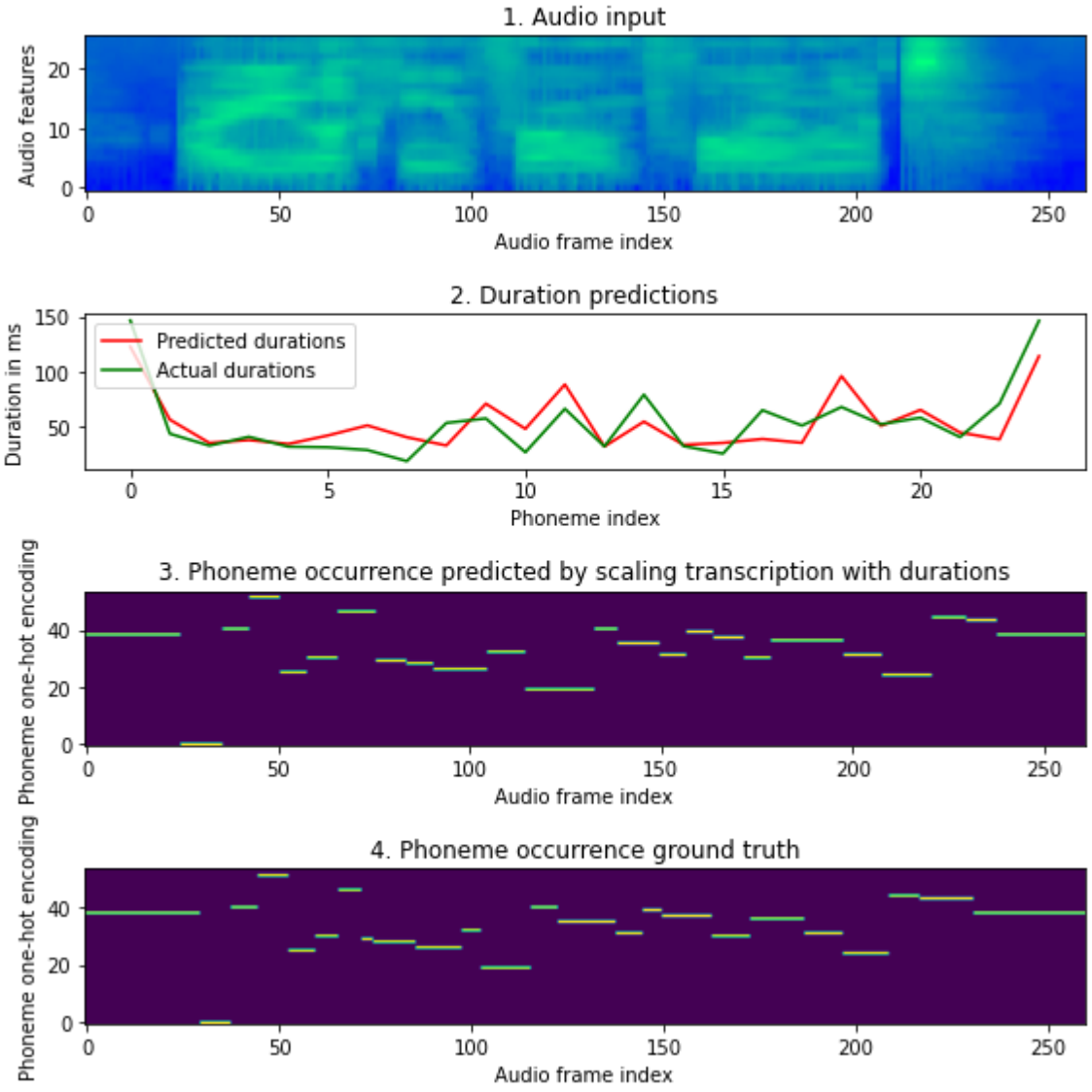

Figure 24 Predicted phoneme durations used to produce a phoneme occurrence graph by scaling the transcription phoneme labels with their duration.

7.2 Results

The duration model satisfied all three requirements but did not produce an acceptable segmentation result. Nevertheless, the additional knowledge that the duration scaled label transcription brought into the sequence tagging model's post-processing step improved the scores significantly. The boundary agreement results are shown in Table 11. The combined model has achieved better agreements than baseline 1 on all boundaries except 10ms and 15ms. It has not exceeded any of the baseline 2 results. The mean difference from the ground truth border is 6.98ms with a maximum deviation of 715ms.

Table 11 Duration enhanced sequence tagging agreement improvements compared to previous results.

	Baseline 1	**Baseline 2**	**Sequence tagging**	**Sequence tagging + durations**
< 5ms	48.28		48.90	**49.30**
< 10ms	**79.30**	77.53	77.37	77.92
< 15ms	**89.49**		88.53	89.03
< 20ms	93.36	**93.92**	93.06	93.57
< 25ms	95.38		95.42	**95.87**
< 30ms	96.74	**97.43**	96.82	97.27
< 35ms	97.61		97.72	**98.11**
< 40ms	98.22	**98.78**	98.28	98.66
< 45ms	98.62		98.67	**99.03**
< 50ms	98.92	**99.35**	98.89	99.24
< 55ms	99.13		99.07	**99.41**
< 60ms	99.32		99.20	**99.53**
< 65ms	99.45		99.31	**99.60**
< 70ms	99.57		99.39	**99.67**
< 75ms	99.64		99.44	**99.71**
< 80ms	99.70		99.48	**99.75**
< 85ms	99.75		99.53	**99.78**
< 90ms	99.78		99.58	**99.81**
< 95ms	99.81		99.62	**99.83**
< 100ms	99.83		99.65	**99.86**

8 Experiments with Soft Pointer Networks

Soft Pointer Networks is a new phoneme segmentation architecture proposed in this thesis. It is widely inspired by the attention mechanism and shares similarities to Pointer Networks.

The first requirement posed was that there must be a one-to-one correspondence between found borders and transcription borders. It stands to reason that maybe instead of finding phoneme probabilities and then using alignment to post-process it into an acceptable state it makes more sense to map the phonemes with their position on the audio segment directly.

One such mechanism to transform a phoneme query over an audio target into a probability distribution is the attention mechanism. As shown on Figure 25, the main idea is to calculate the attention weights (activation scores) shown in the second plot between the phoneme label transcription and audio encodings, but skipping the step to calculate the weighted context over the audio encodings. Instead, the resulting weights are used directly to vote for the border position for the given phoneme. As seen on the last plot, the attention score distribution centre of each of the given five phonemes resides around the ground truth border.

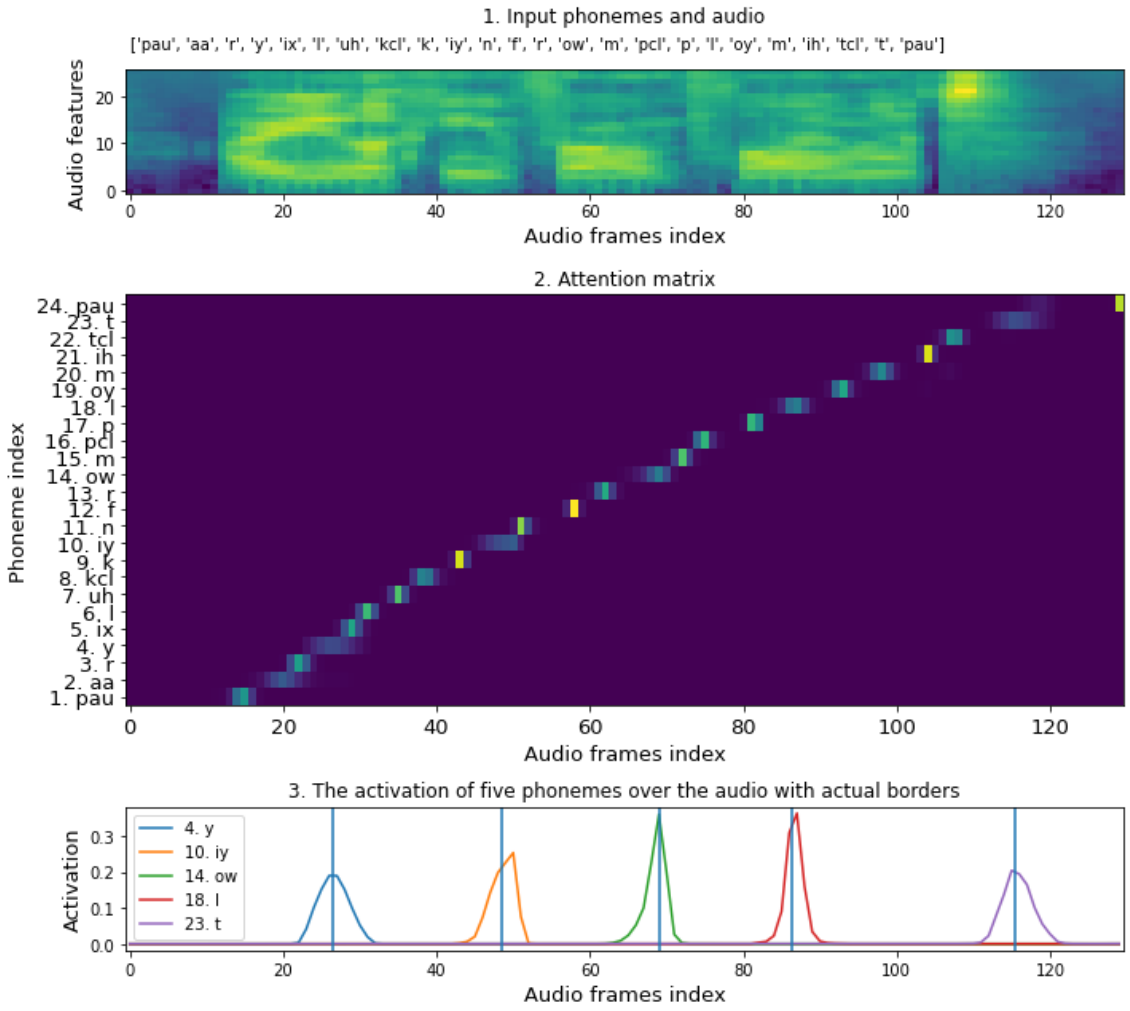

Figure 25 Example of attention scores activation distribution mean being around the actual border. (illustrated by a vertical line)

The key-value-query concepts come from retrieval systems, such as search engines. In the case of searching for a document in a files system, the text written to the search box would be the query which is then compared to the known filenames as the keys in the filesystem database and the content returned from those files would be the values. In phoneme segmentation, the label transcription would be the query to which it is possible to use the audio as the key database and the index gradient as a substitute value database.

8.1 Training targets and model architecture

The main goal of this model is to express the position of the phoneme border. To do so there are 2 options to train against:

1. The attention weights
    a. The raw attention activation matrix using one-hot encoded positions as the ground truth
2. The position gradients
    a. The position encoding vector which holds positional information as described in the original attention paper
    b. The position indexes as scalar values

The proposed model architecture, as shown in Figure 26, produces results for both options. The final output can be index, positional encoding or the attention score matrix. Each output mode is matched with a different loss function. The attention weights could be trained with MSE loss, the position encodings with Cosine loss and the position indexes with MSE loss or SmoothL1Loss. When compared to the previous models in Figure 20 and Figure 23 the model has no decoders and is generally very lightweight. Furthermore, both audio and phoneme inputs are considered equally important.

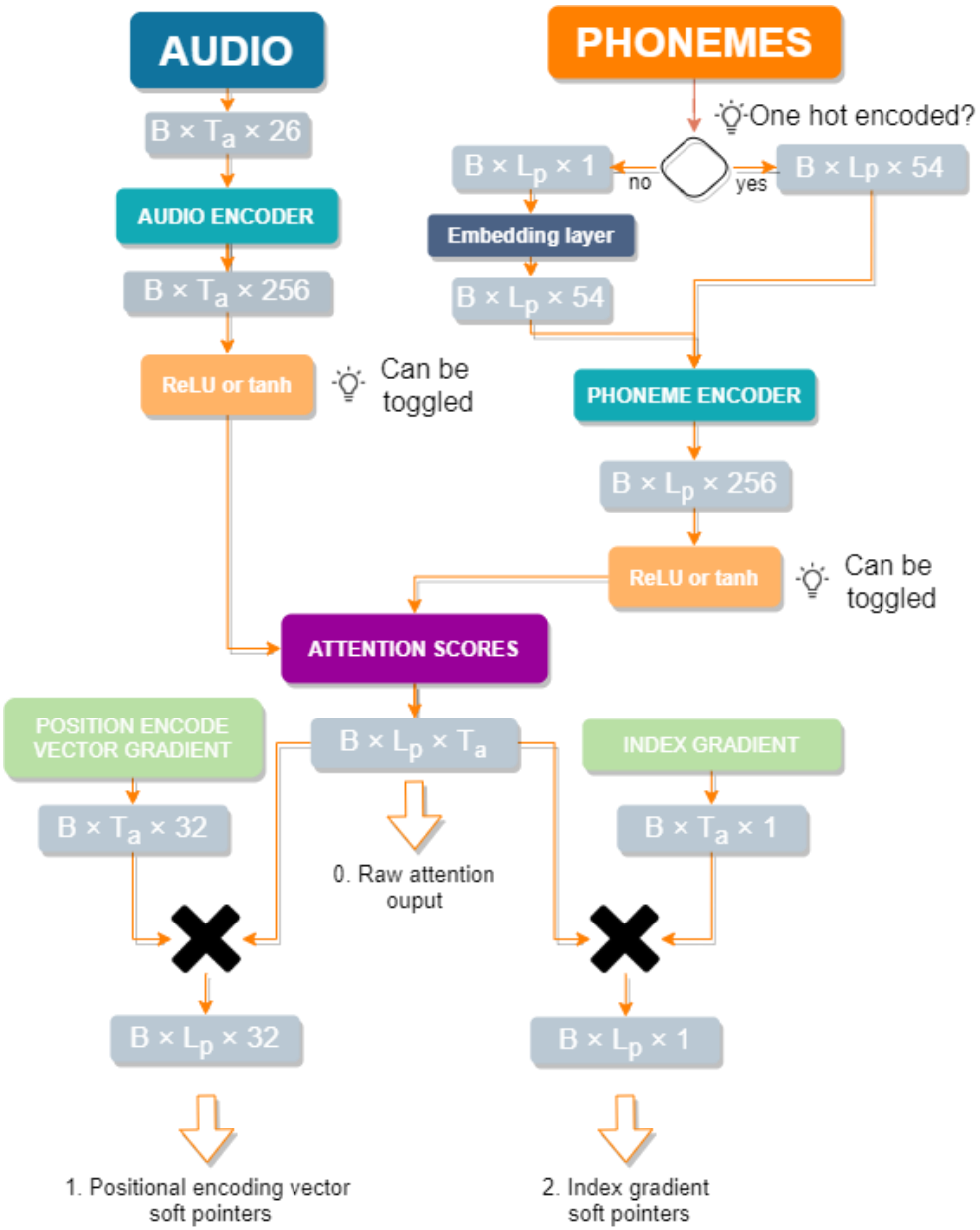

Figure 26 Soft Pointer Network model architecture which combines information from audio and phoneme context and allows multiple modes of output: index, position vector and attention score matrix.

8.2 First option: the attention weights

The first option is straightforward given that the model already produces an attention score matrix, but it was not as easily pipelined within the existing system that was used for the previous models. That is because the two axes of the attention score matrix, sequence and feature length, can vary in size as opposed to the previous solutions where the output only changed in the temporal axis. The problem is illustrated in Figure 27 where when comparing the individual attention score matrices from within a single batch on the four plots, the masked area, shown in a blue tint, varies significantly. This

also means that batch processing would require significantly more effort to mask and account for this.

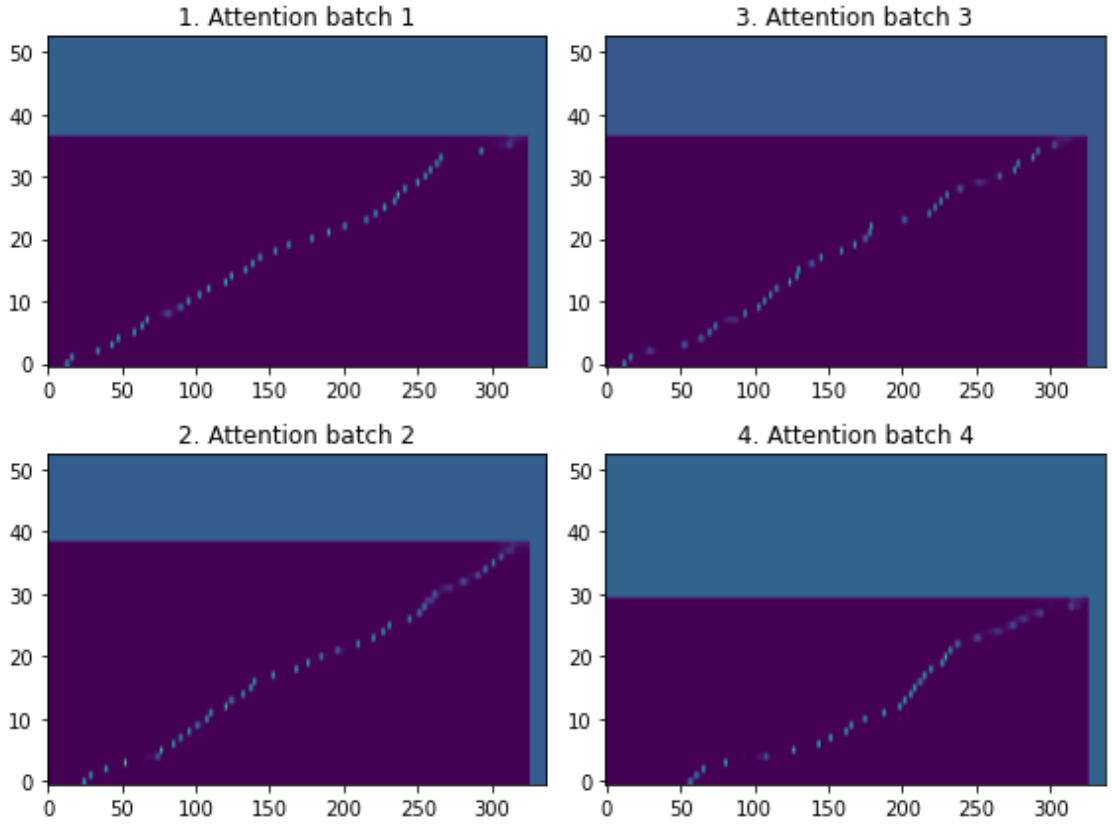

Figure 27 Four examples from a single batch of attention scores, where there is a high variation in masked areas in two dimensions.

The second problem is that the data preparation step must generate an example ground truth distribution as the target to motivate the model to learn also from almost correct positions as seen in Figure 28. The attention scores decay as a function of distance from each of the actual borders which are depicted as vertical lines on the figure. This shows that neighbouring frames are not independent, rather they each transition temporally from each other. As there is a clear incentive to have this gradual decay of attention around the predicted border, to guide the model smoothly between sequential positions, it is evident that the one-hot encoded position without this might be even counterproductive.

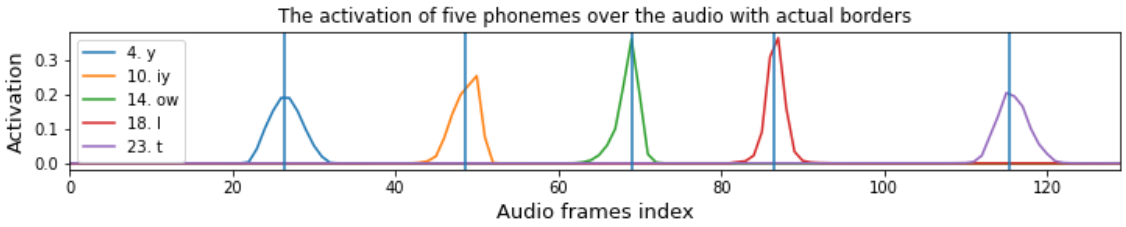

Figure 28 Attention score blips for an audio segment with different distribution shapes.

At first, each of the high attention areas in the target example above seems to be insignificant blips, but they represent activation distributions which have two defining parameters, the mean and standard deviation. In order to generate a training target, the data preparation step would need to define the mean of the blip's distribution as the desired phoneme border, but to choose the correct standard deviation hyperparameter is non-trivial. There is no evident way to choose the hyperparameter σ (standard deviation) for the as the resulting blip shape would also change how the model behaves around the proximity of the border, making the characteristics of the whole system depend on it.

The final output of this model is produced by applying argmax operation on the attention scores to produce the final index. The result of the argmax is the location of the highest peak for the attention score which discards the mean location of the activation which might be more accurate. This makes the model unable to produce the position in terms of index fractions to place the border between two frames. Given these constraints, the use of this output target was not further investigated.

8.3 Second option: the position gradients

Reasonably, it would make more sense to train the model on the argmax result instead, but argmax operation is not differentiable. Nor does it have properties that would make use of the scores on neighbouring timesteps. In general, the operation to find the position of the maximum value in a sequence is not trivially differentiable. So, in order to use the attention scores to calculate a single position a better strategy is needed.

The goal of the second option is also to improve the situation by reducing the complexity of the training process. Instead of having the output be dependent on the audio sequence length, as the one-hot encoded position vectors in the first option, it

would make more sense to encode the positions with a constant size vector or scalar. That way the output and ground truth sequence tensors would only variate in sequence length, not feature dimensions. This compact constant size format has only one dimension to mask, which has an immediate benefit to batching when compared to the first option.

8.3.1 Differentiable soft pointers

Position can be expressed in many ways. It can be a number for one-dimensional data, such as sequences, or a vector for multidimensional data. The authors of the attention mechanism used in this paper propose the use of a multidimensional position encoding for sequences to model explicit temporal relations in sequences. For the purposes of this paper, a position gradient is a sequence of which elements are required to identifiably mapped to the desired output position.

The most natural way to output position encoding is to use the produced audio to phoneme label attention weights, as shown in Figure 19 in section 6.2, to blend a position gradient instead of the audio encoding to a context vector at each decoder step. In effect, this would take the weighted average of the most voted positions. This can be done using an efficient and simple matrix multiplication between the attention weight matrix and the position gradient.

As an example, the position calculation for the tenth phoneme in Figure 25 and Figure 28 is shown in Figure 29. The attention score for the tenth phoneme is shown as the orange line and the actual border is shown as a vertical green line. The red line is the position predicted by the argmax of the attention score and resides on the peak of the attention score plot. The weighted mean position is shown as a blue vertical line and resides in the centre of the distribution. Even though the actual border is not in the midpoint of the attention score distribution, it is more accurately approximated by the soft mean position than the hard argmax position.

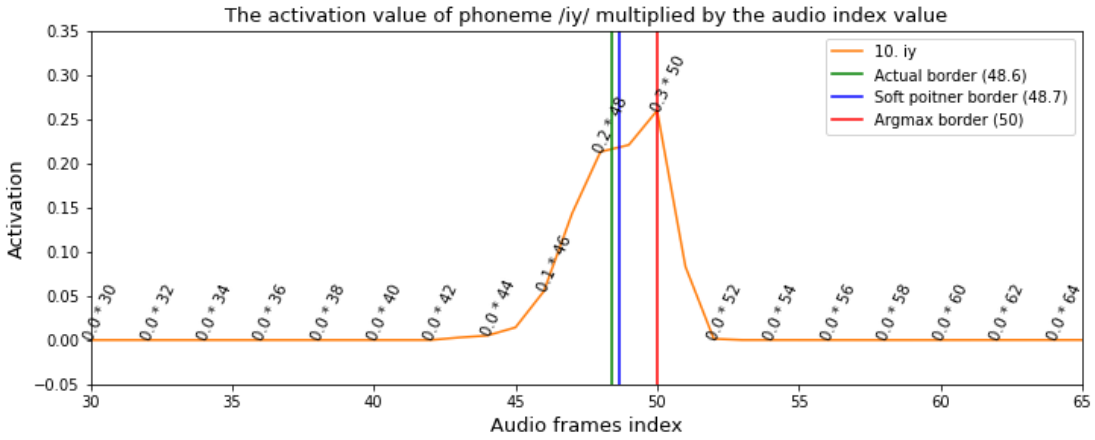

Figure 29 Benefits of using the weighted average of index gradient over argmax operation for interpreting attention weights.

The motivation for this approach comes from the observation, that the attention weights scores add up to 1 and their distribution would presumably have its mean around the desired border position. The previous example illustrates how the desired index can effortlessly be calculated by the weighted average of the position gradient generated from the frame indices. It can be seen there that the weighted mean position of the attention scores is better at estimating the position than the peak position. The fact that the attention scores add up to 1 warrants that the predicted position would always be bounded within the audio sequence. This can further be expressed as formula (15) where n is the length of the sequence, a_i the attention activation and C the final position index.

$$C_i = \sum_{j=1}^{n} a_i^j (j-1) \tag{15}$$

And the generic case for position vectors as formula (16) where V is the output vector and v_i is the i-th position encoding vector.

$$V_i = \sum_{j=1}^{n} a_i^j v_j \tag{16}$$

This is the best-case scenario, as seen from the previous example, which will result in a position with higher accuracy than the position gradient which consists of integers. From this, it can be concluded that it does not depend on the audio encoding step size. The model can express the final answer in terms of index fractions which can place the

border anywhere between two audio frames. This property of the model overcomes the issue of the sequence tagging model's dependence on the audio encoding granularity and circumvents the issues created by the argmax discrete index lookup.

8.3.2 Results

The initial model was trained using index gradients. The training of this model was done using a modified version of SmoothL1Loss. The final accuracy of this model was acquired after training the model without the positional encoding being added to the input sequences. The comparison of raw position and ground truth in Figure 30, suggests that the model performed exceptionally well. The actual border positions, depicted as solid vertical lines, are in proximity of the predicted border positions, depicted as dotted lines, and the attention score activations overlap minimally. There is also no border out of order in this example.

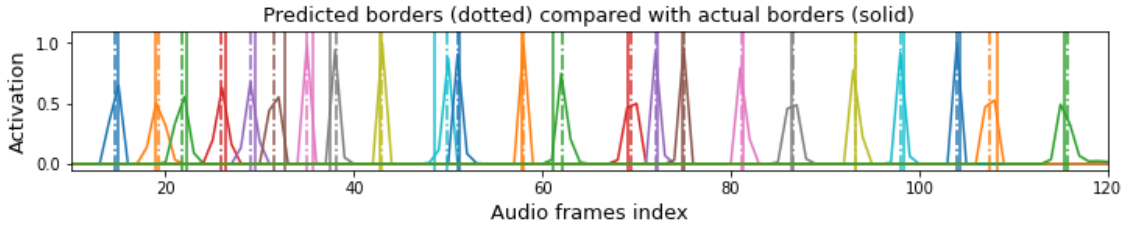

Figure 30 Predicted borders (dotted) compared with actual borders (solid).

The final positions were forced to satisfy requirement 3 by post-processing as described in the following section 8.4.1. Therefore, the final scores presented in Table 12 below satisfy all the requirements. The model has achieved better agreements than both baselines and enhanced sequence tagging model on boundaries less than 30ms. It has exceeded the main 20ms boundary agreement of baseline 2 results. The mean difference from the ground truth border is 8.30ms with a maximum of 748.36ms. There is clear evidence that this model has higher accuracy for sub 30ms boundaries than any other previous implementation.

Table 12 Initial Soft Pointer Network agreement improvements compared to previous results.

	Baseline 1	Baseline 2	Sequence tagging	Sequence tagging + durations	Soft Pointer Network
< 5ms	48.28		48.90	49.30	**57.28**
< 10ms	79.30	77.53	77.37	77.92	**82.07**
< 15ms	89.49		88.53	89.03	**90.56**
< 20ms	93.36	93.92	93.06	93.57	**94.00**
< 25ms	95.38		95.42	**95.87**	95.77
< 30ms	96.74	**97.43**	96.82	97.27	96.85
< 35ms	97.61		97.72	**98.11**	97.54
< 40ms	98.22	**98.78**	98.28	98.66	97.99
< 45ms	98.62		98.67	**99.03**	98.29
< 50ms	98.92	**99.35**	98.89	99.24	98.50
< 55ms	99.13		99.07	**99.41**	98.64
< 60ms	99.32		99.20	**99.53**	98.77
< 65ms	99.45		99.31	**99.60**	98.85
< 70ms	99.57		99.39	**99.67**	98.91
< 75ms	99.64		99.44	**99.71**	98.96
< 80ms	99.70		99.48	**99.75**	99.01
< 85ms	99.75		99.53	**99.78**	99.05
< 90ms	99.78		99.58	**99.81**	99.08
< 95ms	99.81		99.62	**99.83**	99.12
< 100ms	99.83		99.65	**99.86**	99.15

8.3.3 Position encoding

The main motivation for using the attention position encoding as the position gradient is for it to be later used on the second pass through as the replacement positional encoding as described in the reference attention implementation [35]. The repeating process of applying this method could potentially cause the system to converge faster to a solution.

The second benefit to output position encoding vectors is that they also describe the accuracy of the prediction. A well-formed position vector has activations only close to the intended position, but an ambiguous vector might lack the higher frequency features and blend smoothly over a larger neighbourhood of the position. This can be observed in Figure 9.

Training this kind of model requires special attention when choosing the loss function. Training the model with mean squared error (MSE loss) gives a mediocre result. A much better fit is cosine loss (cosine loss) as it is very similar to the matrix dot product step that the attention mechanism uses to produce the scores. Furthermore, observation during the testing suggested that using cosine loss also avoided unpredictable behaviour during training when compared to MSE loss.

8.3.4 Results

To calculate the border agreement percentiles requires the system to output millisecond timestamps in order to be scored. The final output used the intermediate attention weights and the index gradient instead of the positional encodings to output the final positions. The results of this model were trained using the previous index gradient models pre-trained weights as the starting point. There was no improvement in using the model in a recurrent fashion

As seen in Figure 31, the model succeeded in predicting position encodings. The actual position encoding on the first plot and the predicted position encoding on the second plot showed marginal difference as highlighted on the third plot.

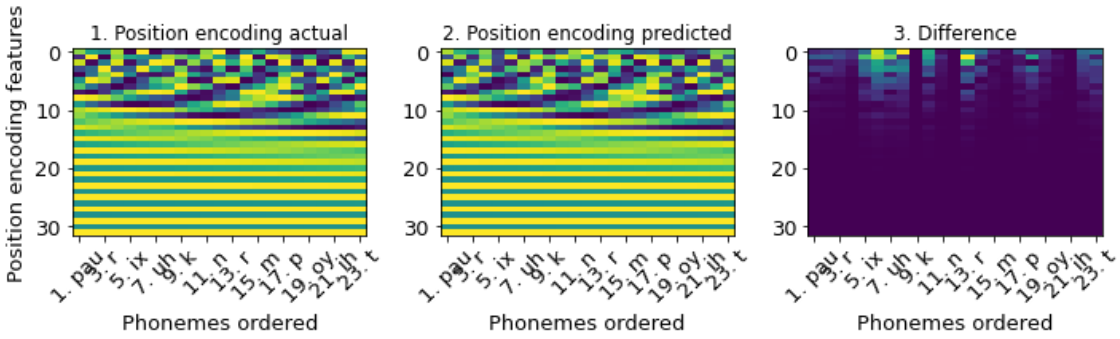

Figure 31 Example of a position encoding prediction.

The results of the positional encoding experiment are shown in Table 13. The model improved sub 15ms scores only marginally and performed slightly worse than the index gradient version on the rest of the boundaries. The model achieved the highest boundary agreement for 5ms. It can be concluded that the positional encodings as the position gradient did not improve the model's performance significantly.

Table 13 Soft Pointer Network trained with positional encodings output mode agreement improvements compared to previous index-based approach.

	Soft Pointer Network	**Soft Pointer Network + positional encodings**
< 5ms	57.28	**57.55**
< 10ms	82.07	82.20
< 15ms	90.56	90.49
< 20ms	94.00	93.92
< 25ms	95.77	95.69
< 30ms	96.85	96.75
< 35ms	97.54	97.41
< 40ms	97.99	97.86
< 45ms	98.29	98.15
< 50ms	98.50	98.34
< 55ms	98.64	98.48
< 60ms	98.77	98.60
< 65ms	98.85	98.68
< 70ms	98.91	98.74
< 75ms	98.96	98.80
< 80ms	99.01	98.85
< 85ms	99.05	98.88
< 90ms	99.08	98.91
< 95ms	99.12	98.94
< 100ms	99.15	98.97

Nevertheless, the experiment shows promising future applications. As shown before, the output position encoding vectors originate from a blend of discrete position vectors which would allow the system to output also positions between two coordinates. While the position encoding gradient is meant to represent one-dimensional coordinates with a multidimensional vector, the proposed model could very well work also with geometric coordinates. This shows that the proposed model could also be used to solve spatial problems where the network would have to output a position interpolated between their coordinates, for example, protein docking.

8.4 Monotonically growing

One of the issues that this method of segmentation creates is that the border indices might not be monotonically growing. The attention scores in Figure 32 illustrate how the model achieved a semi-good prediction of the phoneme positions given a corrupted audio input as seen in plot 1. The incomplete audio data lead the trained model to produce a fluctuating attention activation on plot 2 and is most visible on the final plot for the 14th phoneme. It becomes apparent that in situations like the presented, the choice of midpoint strategy and a more extreme case malformed input might cause displacement of the boundaries.

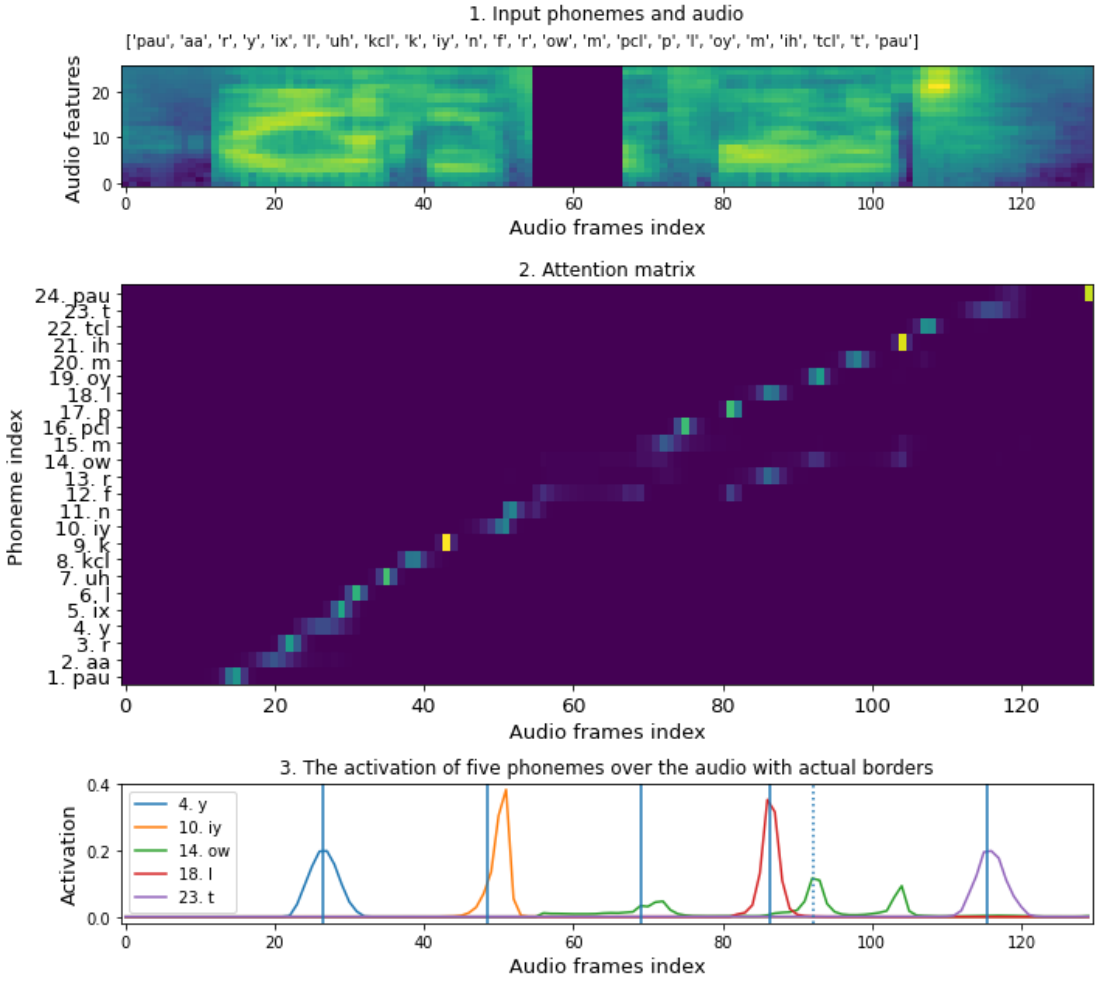

Figure 32 Interference to the data causing the border detection to be out of order.

Observation during the development suggested that the errors were systematic. The mistakes were independent and did not interfere with their neighbouring borders. The most prevalent issue was the mistakenly pointing to a previous or following border of the same phoneme, most likely mistaking whether it had already been handled. Which is why it can be seen that the model does not necessarily satisfy the requirement 3. The out of line border ordering could be fixed with:

- Post-processing
- Progressive masking

8.4.1 Post-processing

The goal of post-processing is to handle cases where the border is preceding the previous or following the next border. The resolution method to solve requirement 3 could be to:

1. Vanilla: Pick the midpoint of the previous and the following border.
2. Duration: Pick the midpoint of the previous and following border with the highest activation.
3. Activation: Place the new border relative to the previous or subsequent border by a predicted duration offset, but not before the previous border.
4. Combined: Average of the previous options.

The fallback if this does also not resolve to an in-order border is to use the previous border.

The results in Table 14 show that not fixing the ordering has the best boundary agreement, but as mentioned does not satisfy the third requirement. Picking the midpoint has the least precision loss when compared to just ignoring the issue. Using the highest activation turned out to be counterproductive as it was heavily biased towards the closest edge, which caused it to reside on the previous or the following border. Using a trained model to predict the duration of the phoneme to offset it relative to the previous one performed the worst. The combinations of options gave no measurable benefits over the previous options.

Table 14 Post-processing boundary agreement results.

	Nothing	Vanilla	Duration	Activation	Combined
< 5ms	56.32	56.31	56.14	56.31	56.31
< 10ms	82.07	82.06	81.81	82.06	82.06
< 15ms	90.99	90.98	90.71	90.97	90.98
< 20ms	94.64	94.62	94.35	94.61	94.62
< 25ms	96.63	96.60	96.32	96.59	96.60
< 30ms	97.78	97.75	97.46	97.74	97.75
< 35ms	98.46	98.43	98.13	98.41	98.43
< 40ms	98.88	98.85	98.55	98.83	98.85
< 45ms	99.13	99.11	98.80	99.08	99.11
< 50ms	99.30	99.28	98.97	99.25	99.28
< 55ms	99.44	99.42	99.11	99.39	99.42
< 60ms	99.51	99.49	99.18	99.46	99.49
< 65ms	99.58	99.57	99.25	99.53	99.57
< 70ms	99.62	99.61	99.29	99.57	99.61
< 75ms	99.65	99.64	99.32	99.61	99.64
< 80ms	99.67	99.66	99.35	99.63	99.66
< 85ms	99.69	99.68	99.37	99.65	99.68

8.4.2 Progressive masking

The weakness of the Soft Pointer model is reoccurring phonemes which makes the model prone to produce these out of order borders. Typically, these errors are not random, but duplications of previous borders. One way would have been to provide the previously attended indices as a feature vector in tandem with the raw audio data before encoding, but this leads to an untrainable big differentiation graph. Nevertheless, there is a clear need for relay previously attended positions during interference that is also feasibly trainable.

One of the key improvements to the model was to introduce a new masking strategy as conveniently the attention mechanism already supported masking the temporal

dimension. Instead of letting the encoder run out of memory, it was better to just omit already attended timesteps. As shown in the first plot in Figure 33, the mask could be generated using the previous phoneme's attention score cumulative sum with a threshold as a mask. As an additional tweak half of the features of the audio encodings, as shown on the second plot, could be multiplied by the un-thresholded cumulative attention scores to reduce the impact of this modification.

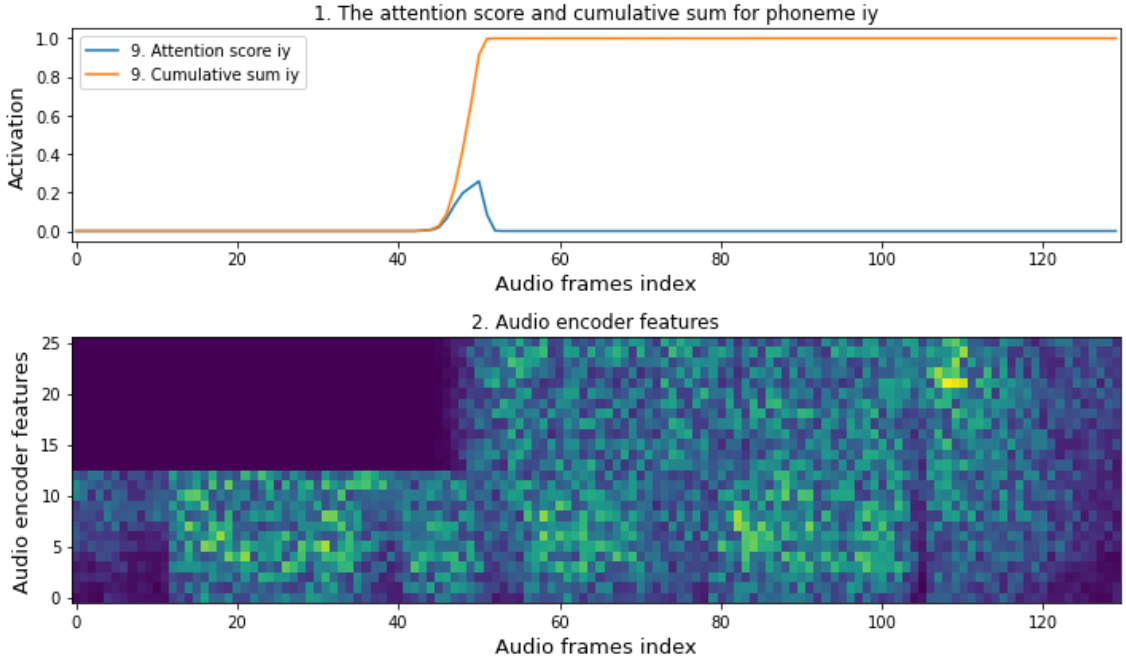

Figure 33 Example of cumulative attention score masking.

This leads to the model being more relaxed about specializing on finding a border as it now does not have to worry as much about re-evaluating the same border twice as much. Furthermore, the use of the softmax scores enables the model to express confidence in the masks. The results can be found in the next chapter.

The injected interdependence can be expressed using conditional probability in the following formula (17). Given the mask vector $b_{C_{i-1}}$, which depends on the previous predicted position C_{i-1}, the probability of C_i following the previous positions C_0 to C_{i-1} is equal the masked attention score with regards to the logit vector u_i.

$$p(C_i|C_0, \dots, C_{i-1}, P) = softmax(u_i * b_{C_{i-1}}) \tag{17}$$

9 Final Soft Pointer Network results

The final model was trained with spectrogram step size modulation, audio augmentations, different output target modes with their respective loss functions. The position mode training was done using a modified version of SmoothL1Loss. This was possible as most of the toolchain was modular and the model was built to be configurable on the fly for each output target. This was also motivated by the intention to force the model to adapt and generalize to avoid overfitting while the training progress being compatible with each model design iteration. The final model could be adapted to multiple audio transformation step sizes in about 4 epochs, which further exemplifies the superior granularity independence of this model over the sequence tagging model. The final output was post-processed with the help of predicted duration offsets to explicitly satisfy the third requirement.

In Table 15 the phoneme average absolute error shows that some phonemes that are less represented in the dataset, such as “/jh/” and “/ax-h/”, performed the worst. The errors are visualized in-depth in appendix 1. This might be a possible area to investigate for future improvements.

Table 15 Phoneme average absolute error.

oy	5.28ms	ao	6.30ms	ux	6.34ms	w	6.37ms	z	6.41ms
ng	6.42ms	r	6.45ms	hh	6.46ms	ch	6.48ms	sh	6.57ms
pcl	6.59ms	m	6.60ms	dcl	6.67ms	hv	6.67ms	iy	6.69ms
aa	6.71ms	v	6.74ms	gcl	6.76ms	l	6.77ms	ax	6.79ms
nx	6.79ms	th	6.82ms	t	6.84ms	bcl	6.85ms	b	6.86ms
tcl	6.86ms	ae	6.94ms	ah	7.01ms	ow	7.02ms	f	7.02ms
uw	7.04ms	g	7.04ms	uh	7.06ms	pau	7.09ms	s	7.10ms
k	7.13ms	aw	7.14ms	ix	7.20ms	y	7.21ms	p	7.26ms
n	7.27ms	ey	7.30ms	kcl	7.32ms	d	7.34ms	dh	7.37ms
er	7.38ms	eh	7.39ms	zh	7.43ms	dx	7.43ms	ih	7.64ms
ay	7.82ms	axr	7.86ms	jh	9.67ms	ax-h	11.68ms		

The final evaluation Table 16 presents that the Soft Pointer Network model has achieved better or equal agreements than baselines on all boundaries less than 70ms. With these findings, the proposed models have exceeded almost all the baselines with the sequence tagging model having higher accuracy above 70ms. The mean difference from the ground truth border is 7.01ms with a maximum of 780.91ms. The main 20ms boundary agreement percentage has been exceeded by 0.69%.

Table 16 Final scores for the Soft Pointer Network model compared to previous results.

	Baseline 1	**Baseline 2**	**Sequence tagging**	**Sequence tagging + durations**	**Soft Pointer Network**	**Soft Pointer Network + enhancements**
< 5ms	48.28		48.90	49.30	**57.28**	55.97
< 10ms	79.30	77.53	77.37	77.92	**82.07**	81.93
< 15ms	89.49		88.53	89.03	90.56	**90.88**
< 20ms	93.36	93.92	93.06	93.57	94.00	**94.61**
< 25ms	95.38		95.42	95.87	95.77	**96.62**
< 30ms	96.74	97.43	96.82	97.27	96.85	**97.79**
< 35ms	97.61		97.72	98.11	97.54	**98.48**
< 40ms	98.22	98.78	98.28	98.66	97.99	**98.87**
< 45ms	98.62		98.67	99.03	98.29	**99.15**
< 50ms	98.92	**99.35**	98.89	99.24	98.50	**99.35**
< 55ms	99.13		99.07	99.41	98.64	**99.48**
< 60ms	99.32		99.20	99.53	98.77	**99.57**
< 65ms	99.45		99.31	99.60	98.85	**99.65**
< 70ms	99.57		99.39	**99.67**	98.91	**99.67**
< 75ms	99.64		99.44	**99.71**	98.96	99.70
< 80ms	99.70		99.48	**99.75**	99.01	99.72
< 85ms	99.75		99.53	**99.78**	99.05	99.74
< 90ms	99.78		99.58	**99.81**	99.08	99.76
< 95ms	99.81		99.62	**99.83**	99.12	99.77
< 100ms	99.83		99.65	**99.86**	99.15	99.79

10 Differences and improvements over Pointer Networks

The proposed gradient lookup architecture came as a natural result of the earlier models. During the development, it was found out that this type of network had already been published under the name Pointer Networks [37]. Nevertheless, the model proposed in this paper offers significant improvements and adaptations for segmentation problems.

Pointer Networks are optimized to find and retrieve elements from an unordered sequence or more accurately, a set of values. Ptr-Nets do not make use of the fact that the audio input database is already ordered to encode positional information in the current audio segmentation task in fact, it disregards it entirely by encoding position targets as one-hot vectors. It is strictly a value retrieval solution. In a sense, it is more like a position classification model rather than a regression model.

The key difference between Pointer Network and the Soft Pointer Network proposed in this paper is:

- The usage of a second input sequence (the phoneme labels) as the query.
- The usage of a position gradient as the value database to enforce temporal relations and allow blended positions

The motivation for using the phoneme transcription as a query has been discussed already in paragraphs 7 and 8. The key takeaway is that it motivates the model to use all the available data for this specific use case in the training process.

The more distinguishing feature is the use of the position gradient. This method offers a far more appropriate approach to segmentation tasks. Encoding the position as a gradient transforms this from a classification problem to a regression one.

The key benefits are:

- Compact representation
- Natural representation
- Differentiable positions

By collapsing the attention score vector, it changes from a variable-length vector into a constant size one. The compact sequence of fixed size positions is substantially more effortless to batch-process as shown in paragraph 8.

With the output being a scalar, it is possible to output positions that are between the inputs of the original sequence as presented in Figure 29. With these enhancements, the model can output phoneme boundaries more naturally.

Ptr-Net defines the target as a one-hot encoded position vector over the whole input length. The loss is also calculated as though it would be a classification task and the final position index is taken by argmax-ing the scores. This disregards the false positives that are off only one to two frames. In the case of an unordered set of inputs, this is desirable, but for audio segmentation, the neighbouring frames are temporally linked. Yet another possibility that this fails to consider is that the desired position might be in between two frames. One option to combat this would be to define the Ptr-Net one-hot encoded target with a steady decline of activation around the desired position instead, softening the position. But that would also introduce an extra hyperparameter to tune the falloff distribution shape and in effect affect the model behaviour.

The proposed network overcomes this by not training directly on the one-hot encoded softmax positions, but on the weighted average positions which were calculated using the attention scores as the weights and the input sequence positions as values. There is no need to define a distribution to also reward positions close to the actual target as it is taken care of by the methods for regression models. Positions close to the desired target would have a smaller loss than positions further away, the same way as it would be with any other regression model. Most importantly, the position returned from this model is differentiable as opposed to the argmax index produced by Ptr-Net.

The proposed model can produce a sequence index when required to retrieve an element or a fractional index when dealing with temporal data. It is also shown that it is possible to use this model to produce an interpolated geometric coordinate, which would be beneficial for solving spatial problems, such as protein docking. Overall, this model is well adapted to solve a distinctly different set of problems.

11 Future work and applications

The Soft Pointer Network model achieved great results for the phoneme segmentation task, but there are possible ways to improve the current model. The most important area is the training process. Most of the time in this thesis was spent in model design which left less time to explore areas such as loss functions and training methods like teacher enforcing.

The myriad of different training targets and possible loss functions presented in this thesis show that there is good reason to explore appropriate loss functions. One of the main areas to improve in this regard would have been to use class weight-based approaches to combat the unbalanced phoneme representation in the dataset. Another possible approach would have been to train a model specifically for each phoneme and use the well-performing phonemes to pre-segment the dataset for more focused lookups for less common phoneme borders. In general, the results could have been improved by the use of more domain-specific knowledge from linguistics.

While the initial design of the Soft Pointer Network model did not use previous outputs in subsequent predictions the proposed progressive masking idea showed the importance of it. When introducing such dependence of previous results, it is important to train the model in a way that would be resistant to early mistakes in the training process for the output to not waste training time. One such method would be to use teacher enforcing to enable the model to learn more effectively over the whole training sequence.

The progressive masking process could be improved by being able to define a better rolling window for the detection area per phoneme. A better strategy to only focus on a relevant timeframe would enable the model to be less prone to re-evaluating the same phoneme twice.

The final improvement would be to use the raw waveform data as the input. While such a frame-level model would require more training as there are more frames with less high-level data to process there could be possible accuracy improvements as the model could define the audio features itself instead of relying on having all the required features present in the Mel-spectrogram representation.

The future goal is to apply the findings of this thesis on the Estonian speech corpora to evaluate its usability and provide it as a fully built out tool for the Estonian language research community. There is further work needed to be done to integrate this solution with their workflow and dataset.

12 Summary

The goal of this thesis was to research and propose a better end-to-end deep learning phoneme segmentation system, which would outperform the baseline solutions, and as a result, three possible neural network architectures were presented. All presented models satisfied the three requirements and produced well-formed outputs.

As the best result of this thesis the novel segmentation model architecture, Soft Pointer Networks was proposed for phoneme boundary detection. The system used Seq-2-Seq models, attention mechanism, and key-value-query retrieval concept. The proposed system used audio segments and phoneme label transcriptions to predict the position of each phoneme in the audio segment directly.

The main feature of this model is that it can learn differentiable positions or indices for queries using a position gradient without losing temporal information. The model was insensitive to the audio transformation step size. It was shown that the model has also a possible application in solving spatial problems where the network would have to output a position interpolated between the coordinates of the input set.

Sequence tagging based systems were more fault tolerant as there is a high dependence on the surrounding phonemes in the phoneme label alignment step, but the Soft Pointer Network achieved better results on sub 70ms boundaries. This difference was caused by the model's reduced constraint on interdependence which allows some predictions to reside well outside the expected region. This was mitigated with post-processing steps.

The proposed Soft Pointer Network system achieved 94.61% boundary agreement within 20ms compared to the manual segmentation on the TIMIT corpus which is a 0.69% improvement over the baseline results. It performed better or equally good in all boundaries less and equal to 80ms than when compared to the baseline systems.

The findings in this thesis show that end-to-end systems are viable options for phoneme segmentation tasks and possibly improve future language research for phoneticians.

References

[1] E. Meister and L. Meister, “Automaatse segmentimise hindamine,” *Mäetagused,* vol. 68, pp. 145-160, 2017.

[2] F.-E. Kirsi, “Soft Pointer Networks,” 2020. [Online]. Available: https://github.com/vegetablejuiceftw/soft-pointer-networks.

[3] J.-W. Kuo, H.-Y. Lo and H.-M. Wang, “Improved HMM/SVM methods for automatic phoneme segmentation.,” in *INTERSPEECH*, 2007.

[4] J. A. Gómez, E. Sanchis and M. J. Castro-Bleda, “Automatic speech segmentation based on acoustical clustering,” *SSPR&SPR'10 Proceedings of the 2010 joint IAPR international conference on Structural, syntactic, and statistical pattern recognition,* pp. 540-548, 2010.

[5] J.-P. Hosom, “Speaker-independent phoneme alignment using transition-dependent states,” *Speech Communication,* vol. 51, no. 4, pp. 352-368, 2009.

[6] E. Meister and L. Meister, “Production of Estonian vowels by Finnish speakers,” *Eesti ja soome-ugri keeleteaduse ajakiri. Journal of Estonian and Finno-Ugric Linguistics,* vol. 10, no. 1, pp. 129-143, 2019.

[7] E. Meister and L. Meister, “Aktsendikorpus ja võõrkeele aktsendi uurimine,” *Keel ja Kirjandus,* pp. 696-714, 2012.

[8] E. Meister and L. Meister, “Development and use of the Estonian L2 corpus,” *Book of Extended Abstracts. Workshop on Phonetic Learner Corpora : 12 August 2015 in Glasgow (Satellite workshop of the 18th International Congress of Phonetic Sciences),* pp. 45-47, 12 August 2015.

[9] E. L. Asu and P. Lippus, “Acoustic correlates of secondary stress in Estonian,” in *9th International Conference on Speech Prosody 2018*, 2018.

[10] M. A. Pitt, K. Johnson, E. Hume, S. F. Kiesling and W. D. Raymond, “The Buckeye corpus of conversational speech: labeling conventions and a test of transcriber reliability,” *Speech Communication,* vol. 45, no. 1, pp. 89-95, 2005.

[11] M. Talimets, “End-to-end speech recognition for Estonian,” *Digital Collection of TalTech Library,* 2018.

[12] The Editors of Encyclopaedia Britannica, “Phoneme,” Encyclopædia Britannica, inc., 2009.

[13] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, 2012.

[14] T. Sainath, R. J. Weiss, K. Wilson, A. W. Senior and O. Vinyals, “Learning the Speech Front-end with Raw Waveform CLDNNs,” in *INTERSPEECH*, 2015.

[15] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” *arXiv preprint arXiv:1609.03499,* 2016.

[16] B. Logan, “Mel frequency cepstral coefficients for music modeling,” in *Proc. of ISMIR 2000*, 2000.

[17] M. Zetlin, “Alexa, Siri, Google and Cortana Answer 800 Questions in IQ Comparison Test,” December 2018. [Online]. Available: https://www.inc.com/minda-zetlin/alexa-siri-google-cortana-loup-ventures-smart-speaker-iq-comparison-test.html.

[18] D. Amos, “The Ultimate Guide To Speech Recognition With Python,” April 2018. [Online]. Available: https://realpython.com/python-speech-recognition/.

[19] A. Stolcke, N. Ryant, V. Mitra, J. Yuan, W. Wang and M. Liberman, “Highly accurate phonetic segmentation using boundary correction models and system fusion,” in *ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[20] D. Wang, X. Wang and S. Lv, “An Overview of End-to-End Automatic Speech Recognition,” *Symmetry*, vol. 11, no. 8, p. 1018, 2019.

[21] A. Graves and N. Jaitly, “Towards End-To-End Speech Recognition with Recurrent Neural Networks,” in *Proceedings of The 31st International Conference on Machine Learning*, 2014.

[22] B. Shillingford, Y. M. Assael, M. W. Hoffman, T. Paine, C. Hughes, U. Prabhu, H. Liao, H. Sak, K. Rao, L. Bennett, M. Mulville, M. Denil, B. Coppin, B. Laurie, A. W. Senior and N. d. Freitas, “Large-Scale Visual Speech Recognition,” in *Interspeech 2019*, 2019.

[23] C. Szegedy, A. Toshev and D. Erhan, “Deep Neural Networks for Object Detection,” in *Advances in Neural Information Processing Systems 26*, 2013.

[24] Y. Bengio, Learning Deep Architectures for AI, 2009.

[25] C. Weng, D. Yu, S. Watanabe and B. H. F. Juang, “Recurrent deep neural networks for robust speech recognition,” in *ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[26] C. Olah, “Understanding LSTM Networks,” 27 August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed January 2020].

[27] A. Graves, A.-r. Mohamed and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[28] A. Graves, N. Jaitly and A.-r. Mohamed, “Hybrid speech recognition with Deep Bidirectional LSTM,” in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[29] F. Zuppichini, “Residual Networks: Implementing ResNet in Pytorch,” July 2019. [Online]. Available: https://towardsdatascience.com/residual-network-implementing-resnet-a7da63c7b278.

[30] D. Yarats, J. Gehring and M. Auli, “A novel approach to neural machine translation,” May 2017. [Online]. Available: https://engineering.fb.com/ml-applications/a-novel-approach-to-neural-machine-translation/.

[31] J. Gehring, M. Auli, D. Grangier, D. Yarats and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017.

[32] A. Hannun, “Sequence Modeling with CTC,” *Distill*, vol. 2, no. 11, 2017.

[33] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang and E. Keogh, “Querying and mining of time series data: experimental comparison of representations and distance measures,” *very large data bases*, vol. 1, no. 2, pp. 1542-1552, 2008.

[34] Y. Li, H. Chen and Z. Wu, “Dynamic Time Warping Distance Method for Similarity Test of Multipoint Ground Motion Field,” *Mathematical Problems in Engineering,* vol. 2010, pp. 1-12, 2010.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, “Attention is All You Need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 2017.

[36] D. Bahdanau, K. Cho and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” in *ICLR 2015 : International Conference on Learning Representations 2015,* 2015.

[37] O. Vinyals, M. Fortunato and N. Jaitly, “Pointer networks,” in *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2,* 2015.

[38] P. B. Ramteke and S. G. Koolagudi, “Phoneme boundary detection from speech: A rule based approach,” *Speech Communication,* vol. 107, pp. 1-17, 2019.

[39] P. Mizera and P. Pollak, “Automatic Phonetic Segmentation and Pronunciation Detection with Various Approaches of Acoustic Modeling,” in *International Conference on Speech and Computer*, 2018.

[40] J. Matoušek and M. Klíma, “Automatic Phonetic Segmentation Using the Kaldi Toolkit,” in *International Conference on Text, Speech, and Dialogue,* 2017.

[41] J. Franke, M. Müller, F. Hamlaoui, S. Stüker and A. Waibel, “Phoneme Boundary Detection using Deep Bidirectional LSTMs.,” in *ITG Symposium on Speech Communication,* 2016.

[42] V. Khanagha, K. Daoudi, O. Pont and H. Yahia, “Phonetic segmentation of speech signal using local singularity analysis,” *Digital Signal Processing,* vol. 35, pp. 86-94, 2014.

[43] A. Rendel, A. Sorin, R. Hoory and A. Breen, “Towards automatic phonetic segmentation for TTS,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2012.

[44] J. Yuan, N. Ryant, M. Liberman, A. Stolcke, V. Mitra and W. Wang, “Automatic phonetic segmentation using boundary models.,” in *INTERSPEECH*, 2013.

[45] 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 CD-ROM {TIMIT} | NIST,” *NIST Interagency/Internal Report (NISTIR) - 4930,* 1993.

[46] R. A. Cole and J.-P. Hosom, “Automatic time alignment of phonemes using acoustic-phonetic information,” *PhDT,* p. 2035, 2000.

Appendix 1 – Phoneme error plots

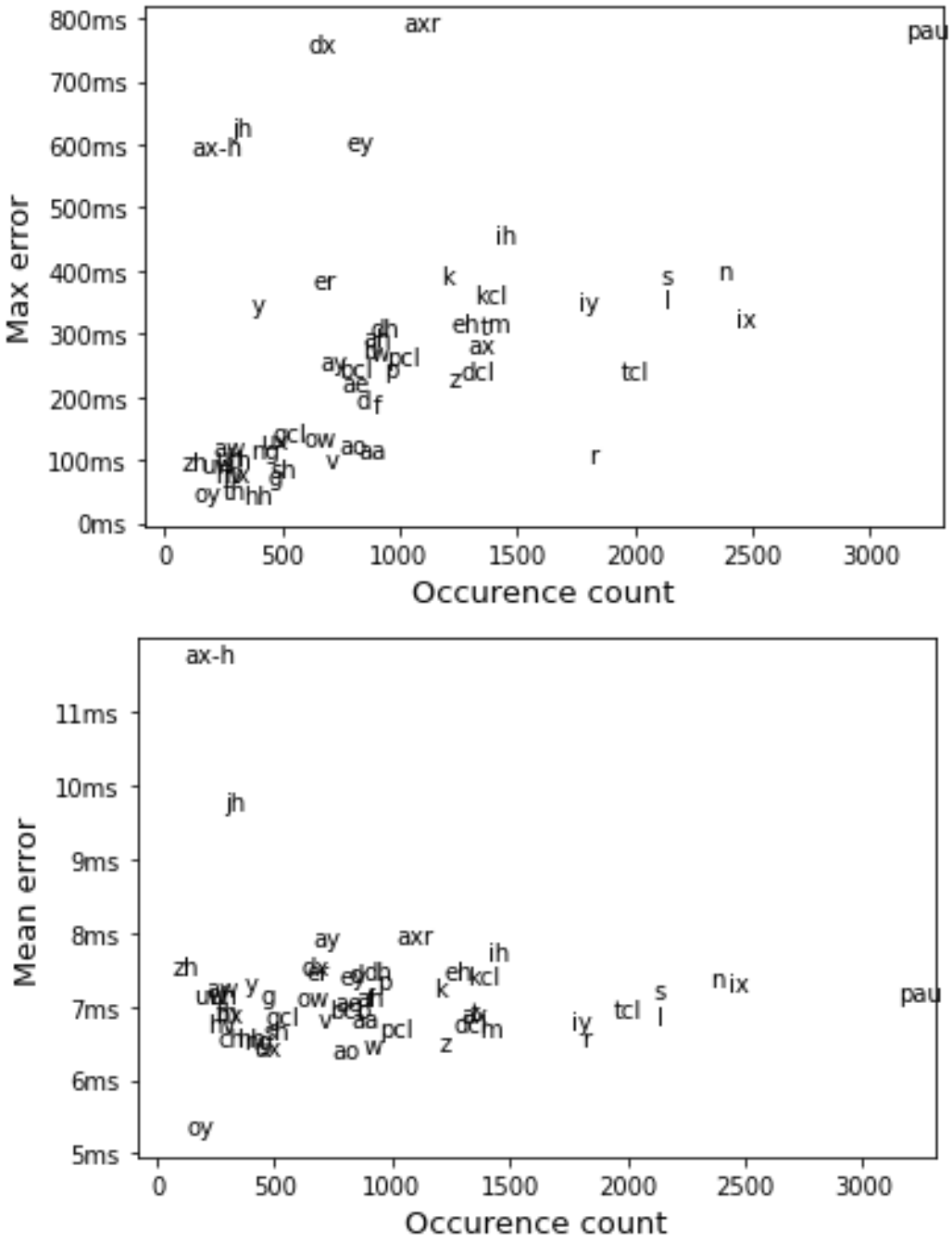

Figure 34 Phoneme error plots.