Duygu Dogan

Methods for Analyzing Attention Heads in Transformer-Based Predictive Models of Speech Representation Learning

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Prof. Okko Räsänen
Prof. Tuomas Virtanen
October 2021
ABSTRACT

Duygu Dogan: Methods for Analyzing Attention Heads in Transformer-Based Predictive Models of Speech Representation Learning
Master of Science Thesis
Tampere University
Data Engineering and Machine Learning
October 2021

Transformer has recently become one of the most popular deep learning models often utilized for processing sequential data. At the core of the Transformer, a mechanism called attention lies. The attention mechanism aims to learn the dependencies between different items in a sequence. Furthermore, Transformer applies multiple parallel attentions, called multi-head attention, in order to incorporate different types of dependencies behind the data. As Transformer became widely common, much literature has been devoted to the investigation of attention heads in natural language processing tasks. However, how they work in speech processing tasks is still under-explored.

The present study analyzes multiple methodologies to analyze the attention heads in speech representation learning tasks. For this purpose, two different Transformer-based predictive coding models with different learning strategies, Autoregressive Predictive Coding and Contrastive Predictive Coding, are used. Furthermore, attentions are grouped into explainable categories by using temporal analysis along with correlation and linear regression methods with respect to the known characteristic of speech. Additionally, the contributions of individual heads to the performance on phoneme classification tasks is evaluated and analyzed by their designated categories using the aforementioned methods.

The results of correlation and linear regression analyses show that individual attention heads have different functionalities and learn from varying speech features. Combining with temporal analysis, the findings further indicate that the heads learning from phonetic features tend to accumulate their attention in neighbor frames more consistently. On the other hand, the heads learning from other acoustic features spread their attention in temporally further past. In addition, the analyses indicate that instead of utilizing all heads, only a subset of heads can be used without seriously affecting the performance of the phoneme classification tasks. Although the choice of the subset is mainly related to the temporal behavior of the heads, it varies depending on the learning strategy of the model. As the model tries to predict further in the future, the best subset of heads consists of the ones that concentrate their attention on the more recent past.

Keywords: speech representation learning, predictive coding, transformer, self-attention, interpretability

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.
PREFACE

This thesis is the final step of my MSc. studies. I would like to thank everyone who has supported and motivated me throughout this journey.

My first debt of gratitude undoubtedly goes to my supervisors Prof. Okko Räsänen and María Andrea Cruz Blandón. They have always inspired me with a tremendous amount of new ideas. Without their support and constant guidance, I would not have been able to complete this thesis. I must also thank all the members of the Speech and Cognition Research Group for creating such a friendly environment and for all the insightful discussions. I would also like to thank Prof. Tuomas Virtanen for reviewing this thesis.

I would like to thank my friends in Finland; Atakan, Mert, and Pragya, for making this journey unforgettable with all the wonderful times we shared together. I would like to extend my sincere thanks to my friends in Turkey; Gaye, Oguz, and Altug, for always being there for me despite the thousands of miles between us.

I owe my sincere thanks to my dearest friend, Mehmet Killioglu, not only for his endless support and friendship but also for all the advice he gave, even under the stress of his own thesis.

Last but not least, my deepest gratitude is for my parents Sengul Dogan and Kemal Dogan, and my brother, Deniz Dogan. Their unwavering love, support, and sacrifice have always been a source of motivation for me. Without them, I would not be the person I am today.

Tampere, 12th October 2021

Duygu Dogan
5.1.2 Analysis of the heads on the speech features . . . . . . . . . . 46
5.2 Phoneme classification results . . . . . . . . . . . . . . . . . . . . 49
5.3 Discussion of the results . . . . . . . . . . . . . . . . . . . . . . . 53
6. Conclusion and Future Work . . . . . . . . . . . . . . . . . . . . . 56
References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
Appendix A: Distribution of the sounds in attention heads . . . . . . . . 64
Appendix B: Confusion matrices for heads . . . . . . . . . . . . . . . . 66
LIST OF FIGURES

2.1 Illustration of artificial neural network. ........................................... 4
2.2 Structure of a perceptron. ............................................................... 5
2.3 Representation of activation functions. ............................................ 8
2.4 Visual representation of 2D convolution in the convolutional layer. ...... 9
2.5 Visual representation of max pooling operation. ............................... 10
2.6 Illustration of a simple recurrent neural network. ........................... 11
2.7 Transformer architecture. ............................................................... 12
2.8 Scaled dot-product attention. ......................................................... 13
2.9 Illustration on attention mask. ....................................................... 15
2.10 Multi-head attention. ................................................................... 15
2.11 Illustration on positional embeddings of Transformer. .................... 16
2.12 Architecture of autoregressive predictive coding. ........................... 19
2.13 Architecture of contrastive predictive coding. ............................... 19
2.14 Diagram of articulators. ................................................................. 21
2.15 A 25 ms long speech segment and its Hamming-windowed version. ... 23
2.16 The distribution of English vowels on F1 and F2 frequencies. .......... 25

3.1 An overview of the head analysis system in the data pre-processing and training and analysis phases. ................................................................. 28
3.2 Architecture of the modified Transformer layer in predictive coding models for speech representation learning task. ................................. 30
3.3 Attention maps of heads visualized on the same 2 s long utterance. .... 32
3.4 Masking the attention maps. ............................................................ 33
3.5 Computing the cumulative attention. ............................................... 33
3.6 Forming the new attention matrix $A_{vowel}$. .................................. 34

4.1 Epoch-loss learning curves. ............................................................. 41

5.1 Attention weights visualized on the same 2 s long utterance for CPC and APC models. ............................................................. 44
5.2 Attention-distance plots on a 2 s long utterance. ............................ 45
5.3 Correlation coefficients between speech features and standard cumulative sum. ............................................................ 47
5.4 Correlation coefficients between phoneme identities and masked cumulative sum. ................................................................. 48
5.5 Linear regression between speech features and standard cumulative sum. ................................................................................ 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6 Phoneme classification accuracy with forward selection.</td>
<td>53</td>
</tr>
<tr>
<td>A.1 Distribution of the sounds for each head in the cumulative attention for CPC model</td>
<td>64</td>
</tr>
<tr>
<td>A.2 Distribution of the sounds for each head in the cumulative attention for APC model</td>
<td>65</td>
</tr>
<tr>
<td>B.1 The normalized confusion matrices for CPC heads.</td>
<td>66</td>
</tr>
<tr>
<td>B.2 The normalized confusion matrices for APC heads.</td>
<td>67</td>
</tr>
</tbody>
</table>
LIST OF TABLES

2.1 English phonetic alphabet using ARPAbet and IPA phonetic symbols. . . . 22
3.1 Properties and purposes of the selected acoustic features. . . . . . . . . . . 31
4.1 Data subsets in LibriSpeech. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
5.1 The mean distance and standard deviation of the heads across the samples. 46
5.2 LibriSpeech phoneme classification results. . . . . . . . . . . . . . . . . . . . . . . 50
5.3 Accuracy results of individual heads on phoneme classification. . . . . . . 50
5.4 Accuracy results by pruning heads on phoneme classification. . . . . . . . 51
LIST OF SYMBOLS AND ABBREVIATIONS

\( F_0 \) \quad \text{Fundamental frequency}

\( r \) \quad \text{Pearson’s correlation coefficient}

ANN \quad \text{Artificial Neural Network}

APC \quad \text{Autoregressive Predictive Coding}

ASR \quad \text{Automatic Speech Recognition}

BERT \quad \text{Bidirectional Encoder Representations from Transformer}

CNN \quad \text{Convolutional Neural Network}

CPC \quad \text{Contrastive Predictive Coding}

DCT \quad \text{Discrete Cosine Transform}

DFT \quad \text{Discrete Fourier Transform}

DNN \quad \text{Deep Neural Network}

GPT \quad \text{Generative Pre-trained Transformer}

GRU \quad \text{Gated Recurrent Unit}

IPA \quad \text{International Phonetic Alphabet}

LPC \quad \text{Linear Predictive Coding}

LSTM \quad \text{Long Short-Term Memory}

MFCC \quad \text{Mel-Frequency Cepstral Coefficients}

MI \quad \text{Mutual Information}

MLP \quad \text{Multilayer Perceptron}

MoA \quad \text{Manner of Articulation}

MSE \quad \text{Mean Squared Error}

NCE \quad \text{Noise-Contrastive Estimation}

NLP \quad \text{Natural Language Processing}

NN \quad \text{Neural Network}

ReLU \quad \text{Rectified Linear Unit}

RNN \quad \text{Recurrent Neural Network}

SL \quad \text{Supervised Learning}
STE  Short-Time Energy
UL   Unsupervised Learning
WER  Word Error Rate
1. INTRODUCTION

The main purpose of speech is communication through language [1]. Beyond its linguistic message, a speech signal also carries a vast amount of information about the speaker, such as native language, emotional state, and attitude [1], [2]. This richness of information that can be conveyed and perceived from speech has created a demand for continuous development on various types of applications in speech technology. Throughout history, speech technology has evolved from the invention of the telephone in 1881 to deep learning based approaches.

Deep learning models have become widely popular for solving challenging tasks in various fields of applications including speech technology (e.g. Apple’s Siri, Amazon’s Alexa). These models nowadays work accurately and their applications have a significant effect on individuals and further society. However, since their mechanism is not explicitly explainable to human cognition, they often remain as black-boxes making the reasoning of the system hard to be interpreted and understood [3]. Therefore, they bring along a new problem of ensuring the transparency of the model and understanding how it works. From the beginning, interpretation of machine learning models has always been a necessity for applications used in high-risk environments to eliminate undesirable biases and guarantee reliability. However, as deep learning models develop further with the current trend, the need for interpreting these models requires additional attention in various applications [4]. Besides human curiosity and learning driving this need [5], it has become essential for debugging and thus, improving the performance and impact of these models [6]–[8].

Recently, Transformer [9] with self-attention has become one of the most popular and ground-breaking deep learning models. Transformer was first introduced for machine translation tasks in 2017 and has become expeditiously widespread especially within Natural Language Processing (NLP) community. Arguably, the most well-known and influential examples of these models are Bidirectional Encoder Representations from Transformer (BERT) [10] and Generative Pre-trained Transformer series (GPT-2 [11], GPT-3 [12]). Due to the vast popularity of Transformers, much literature has been dedicated to its investigation (e.g. [13], [14], [15]). However, its applications in the field of speech have been slower in comparison. Consequently, they are yet to be explored for speech processing tasks.
Although Transformer consists of stack of different neural network layers, a mechanism called attention underlies its core structure. Therefore, attention mechanism is especially of interest while investigating Transformer architecture. In the broadest terms, attention reflects the dependencies between different items in a sequence. Furthermore, applications of Transformer commonly use multiple parallel attentions, called multi-head attention, with the aim of exploiting different dependencies. Hence, in the applications of Transformer in speech processing tasks, multiple attention heads are expected to learn to focus on different acoustic features in the speech signal.

1.1 Research questions of the thesis

The main research goal of this thesis is to develop a methodology to analyze the attention mechanism of Transformer-based speech representation learning models. In this thesis we aim to tackle the following research questions:

1. Do the attention heads learn to focus on different types of speech features?
2. How is the temporal behavior of the heads related to the speech features they are learning to focus on?
3. Can the heads be grouped by the speech feature they are specialized on both quantitatively and visually? If so, which are the most important ones for specific downstream tasks?

1.2 Structure of the thesis

This thesis is organized as follows. Chapter 2 outlines the main concepts of the present study. Chapter 3 explains the methodology used in the experiments. Chapter 4 describes the experiments conducted, followed by a review and discussion on the results of these experiments in Chapter 5. Finally, Chapter 6 concludes the main findings and gives a discussion on the possible future studies.
2. THEORETICAL BACKGROUND

This chapter describes the general description of the main concepts contributing to the thesis work. First, Section 2.1 introduces the fundamental machine learning and neural network concepts. Then, Section 2.2 dives into a specific deep learning model, Transformer. Next, Section 2.3 gives an overview of the predictive coding for speech representation learning mainly focusing on two specific models. Finally, Section 2.4 describes the basics of speech characteristics and processing.

2.1 Machine learning and artificial neural networks

Machine learning is a method used to recognize patterns in raw data [16], [17]. These raw data are often known as training data. The main objective of the machine learning algorithms is to build a model to perform specific tasks. However, it should be noted that instead of explicitly developing these models, they are trained with a large amount of data to identify patterns and make predictions or decisions accordingly. During this training process, the models optimize an objective function. The objective function can be in different types, one of the most commonly used of which is loss function. Loss functions define the discrepancy between the predicted and true outputs. Machine learning algorithms aim to minimize this discrepancy.

One of the most common algorithms used in the field of Machine Learning is called Artificial Neural Network (ANN), or simply Neural Network (NN). ANNs are inspired by the human biology and resembles the connections between human neurons. In the broadest terms, ANN can be defined as stack of small processing units called neurons in different levels called layers and the connections between them. Figure 2.1 illustrates a simple ANN architecture.

Deep learning, nowadays the most popular machine learning approach, is based on ANNs. Specifically, Deep Neural Networks (DNNs) are the subtypes of ANNs that have more than one hidden layer or introduce a certain level of complexity in the connections of the neurons. Universal approximation theorem states that an ANN with one single layer is, in principle, enough to model any non-linear representation [17]. However, the layer of shallow ANNs gets massively large yielding to difficulties in training in order to reach the representational power of deep networks [16]. Due to this computational efficiency of
Section 2.1.1 presents the basic principles of artificial neurons. Then, Section 2.1.2 outlines the most commonly used deep learning architecture, multilayer perceptrons. Next, Sections 2.1.3 and 2.1.4 respectively, describe the training process of the neural networks and most well-known activation functions. Finally, Sections 2.1.5 and 2.1.6 present particular deep learning models, convolutional and recurrent neural networks, respectively.

2.1.1 Artificial neurons

An artificial neuron is the main component of ANNs. A biological neuron, at its core, gets activated and passes along the received information depending on the sum of the signals it receives. As inspired by human brain, artificial neurons also mimic this behavior and receive the input signal as the weighted sum of the signals computed in the neurons at its preceding layers. During this process, a bias term is often introduced to the system in order to introduce a mean shift. In the end, generated output from the neurons is passed along an activation function to compute the final output [16].

One simple yet important ANN is the perceptron [18], which is regarded as a pioneering model presented by Frank Rosenblatt in 1958. A perceptron takes \( n \) inputs and makes a binary decision depending on the weighted sum of the inputs. Since a perceptron is used for binary classification tasks, it utilizes unit step function as an activation function. Figure 2.2 shows the architecture of a perceptron and the calculation of the output can be seen on Equation 2.1.

\[
y(x) = \begin{cases} 
1, & b + \sum_{i=1}^{n} w_i x_i > 0 \\
0, & b + \sum_{i=1}^{n} w_i x_i \leq 0
\end{cases}
\]  

Figure 2.1. Illustration of artificial neural network. Circular nodes represent neurons.
where \( w_i \) represents the weights given to each of the inputs \( x_i \) in a given input sequence \( x = \{x_1, x_2, x_3, ..., x_n\} \).

\[ y = \sum w_1 x_1 + \sum w_2 x_2 + \ldots + \sum w_n x_n \]

\( b \)

\( \sum \)

\( \) inputs

\( \) weights

\( \) Activation function

\( \) Figure 2.2. Structure of a perceptron.

2.1.2 Multilayer perceptrons

As simple as a single layer perceptron is, it works well only for linearly separable data \cite{ref17}. This limitation is eliminated by using hidden layers as shown in Figure 2.1. This architecture is often known as Multilayer Perceptron (MLP) and they are the most commonly used deep neural networks \cite{ref19}.

In MLPs, each neuron in a layer has an influence on every neuron in the following layer, making the network fully connected. Additionally, each layer can only take inputs from its preceding layer. Therefore, they are also referred to as Feedforward Neural Networks (FFN).

The basic computation inside of a perceptron underlies the FFN algorithm as well. Unlike a single layer perceptron, however, the computational steps apply through each upcoming layer in a sequential manner. Outputs produced in a neuron are used as input to its connected neurons. The outputs of hidden layers are known as hidden representations.

For more efficient computation, weight vectors \( w \) are typically packed into a single matrix \( W_{mn} \). Each element \( W_{mn}(j, i) \) in the matrix corresponds to the weight value of the connection between \( i \)th neuron in the layer \( m \) and \( j \)th neuron in the layer \( n \). Furthermore, for a simplified notation, a dummy neuron with value 1 can be added to the input vector \( x \) to replace the bias term \( b \). Finally, in an FFN with \( n \) layers (including input, hidden, and output layers), the hidden representation \( h_m \) of hidden layer \( m \in \{2, ..., n - 1\} \) and the final output \( \hat{y} \) are calculated as:

\[
\begin{align*}
    h_m &= \alpha_{m-1} I_m (W_{m-1} I_m x) \\
    \hat{y} &= \alpha_{n-1} I_n (W_{n-1} I_n h_{n-1})
\end{align*}
\] (2.2)
where $\alpha_{l_{m-1}l_m}$ is the activation function used in the connections between the input ($l_1$) or $m-1$th hidden layer and the $m$th hidden ($l_m$) layer and $\alpha_{l_{n-1}l_n}$ is between the last hidden ($l_{n-1}$) and output ($l_n$) layers.

### 2.1.3 Training of neural networks

The training process of a neural network can be seen as a mathematical optimization problem. The goal of training is to find optimized weights that minimize discrepancies between the predicted and desired output of the network. This discrepancy is calculated by a loss function $L$. There are different loss functions that will be covered in the later chapters. In order to explain the basics of the training process, this chapter only introduces one of the simple, yet most common loss functions, mean square error (MSE) loss:

$$L_{MSE} = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

(2.3)

where $N$ is the number of samples and $y_n$ and $\hat{y}_n$ are the desired and predicted outputs, respectively.

As $\hat{y}$ is defined in Equation 2.2, the loss function $L$ depends on the model parameters $W$ and $b$ in the case of $b$ not being packed into $W$. Hence, by utilizing the loss function, the weights along the network are adjusted using a training algorithm. Backpropagation (BP), which is the most common training algorithm, refers to computing the gradient of the loss function by propagating backward through the network from output to input layer. Here, the gradients define the partial derivatives of the loss function with respect to the model parameters, and they are calculated by using an optimization method called gradient descent. The gradient descent method repeatedly updates the given weights by moving along the negative direction of the gradient with small steps. These small steps are gradients multiplied by a pre-defined parameter called learning rate $\eta$. By doing so, the gradient descent method aims to find the local minimum of the loss function. The updating is done for each complete training pass called epoch $\tau$. For a weight $w_{i,l}^\tau$ of the $i$th neuron between layers $l$ and $l-1$ in epoch $\tau$, the updated weight for the next epoch is defined as follows:

$$w_{i,l}^{\tau+1} = w_{i,l}^\tau - \eta \frac{\partial L}{\partial w}$$

(2.4)

Instead of computing individual gradients for each weight in the network, the backpropagation algorithm computes the gradient of the loss function with respect to all parameters along the network by using the chain rule. During the training process, the weights can be randomly initialized, or previously learned pre-trained parameters can be utilized. In the latter case, the whole pre-trained network or only a specific part of it can be utilized. Additionally, the representations learned in the hidden layers of the pre-trained model can be
used as input features [20]. Furthermore, different layers within a model as well as of different models can share their parameters resulting in reducing the number of parameters and learning more robust representations.

In supervised learning (SL), the model learns the mapping between input \((x)\) and output \((\hat{y})\) pairs based on the provided output labels. On the other hand, in unsupervised learning (UL), the given data is unlabeled and the model learns the patterns on its own without any corresponding output label e.g. by clustering or grouping the data. In self-supervised learning, similar to UL, the data is unlabeled. However, in contrast to UL, the self-supervised models generate a supervisory signal from the data to compare predicted \((\hat{y})\) and desired \((y)\) outputs.

### 2.1.4 Activation functions

Activation functions define the transformation applied to outputs of individual artificial neurons. They often introduce a non-linear mapping between input and output signals and thus provide the model with a better capability to learn non-linear relations. Even though the choice of activation function depends on the task, the most commonly used ones are sigmoid, softmax, and rectified linear unit functions.

**Sigmoid function** is the differentiable version of the step function and the generated output inside of the sigmoid function is bounded between 0 and 1. Hence, it is widely used for binary classification problems and formulated as follows:

\[
\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}
\]

**Softmax function** normalizes the generated output over the output classes in a way that the summation of the components equals 1 as follows:

\[
\hat{y}_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.6}
\]

where \(\hat{y}_j\) is the probability distribution for each class \(j\). After the output classes are transformed into probability distributions, the predicted output is determined as the most likely class \(c\):

\[
c = \text{argmax}_{j}(\hat{y}_j) \tag{2.7}
\]

Softmax operation is commonly preferred as an output layer activation function in multi-class classification problems.
Rectified linear unit (ReLU) is a piece-wise linear function and is expressed as:

\[ \text{ReLU}(x) = \max(0, x) \]  

(2.8)

where \( x \) is the input to ReLU. ReLU maps the negative values to zero and keeps the positive values as they are. Therefore, they preserve many of the desirable properties of linear activation functions [16]. For instance, the models that use linear functions are easier to train than those that use nonlinear functions. Additionally, the positive values are unbounded which prevents the function from saturation and thus converges faster when the gradients are computed during backpropagation. Due to this simplicity and stability of ReLU, it is the most popular activation function [21].

Figure 2.3 visualizes the representation of sigmoid and ReLU activation functions.

\[ f(x) \]

\[ \sigma(x) \]

\[ \text{ReLU}(x) \]

Figure 2.3. Representation of activation functions, \( f(x) \). Rectified linear unit (ReLU) in red, sigmoid (\( \sigma \)) in blue.

2.1.5 Convolutional neural networks

Convolutional neural networks (CNNs) were first introduced for recognizing handwritten digits [22]. However, they have been shown to be powerful for any temporal data (e.g. in NLP tasks [23], [24], [25], [26], [27]). CNNs use convolution operation in one or more of its layers, as the name signifies, and thus they take advantage of spatial or temporal pattern in data. The main advantage of CNNs over FFNs is that CNN can capture the local patterns with far fewer number of parameters [28]. Additionally, they are translation invariant by being able to detect the patterns regardless of their position. A typical CNN-based model is composed of the following layers; convolutional, pooling, and fully connected layers. In the remaining of this section, the role of each layer is presented.
First, in a convolutional layer, a kernel with a predefined size slides through the input data and performs convolution operation at each position. This operation can also be regarded as a summation of element-wise multiplication between the input regions and the kernel. Figure 2.4 visualizes a 2D convolution operation. The produced output is called an activation map [29]. In the convolutional layer, kernels are the learned parameters and are used to recognize patterns by exploiting the locality of the input data. Typically, in visual object detection tasks, a kernel can be as simple as an edge detector at the foremost layers, carrying the information about lines and circles. Yet, depending on the application it is used for, the deeper the network is, the more complex these kernels get, e.g. having a receptive field of the size of the image as it is accumulated through previous layers. In
addition, the convolutional layers can use a non-linear activation similar to FFNs.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>9</th>
<th>8</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

max pooling

<table>
<thead>
<tr>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 2.5. Visual representation of max pooling operation with 2x2 window. The largest value within each window in the activation map is the output of the corresponding window.

Second, in the pooling layer, a pooling function operates on the activation map. The function performs down-sampling by replacing certain regions of the output with the value derived from its local neighbors. By doing so, it does not only reduce the number of parameters but also helps the activation maps to be invariant to small changes of the input positioning [16], [29]. Figure 2.5 shows the computation process of one of the most popular pooling functions, max pooling.

Finally, CNN-based models commonly use a regular fully-connected layer after the convolutional layers in order to map the features learned by the last CNN layers to an output. For instance, for a multiclass classification task, the output layer uses a softmax function and produces probability distribution across the classes.

2.1.6 Recurrent neural networks

Recurrent neural network (RNN) [30] is another deep learning architecture and contains cyclic connections between its neurons. RNNs are commonly preferred for sequence processing tasks due to their ability to capture long-term dependencies [16].

As can be seen in Figure 2.6, the main distinctive pattern of RNN architecture is the usage of hidden states in the network. Different from the hidden layer, these hidden states are used to store earlier outputs of the neurons and then fed back to their corresponding neurons as input during the next processing step. Consequently, RNNs cannot be trained with the standard backpropagation algorithm. Instead, they are trained by a modified algorithm called backpropagation through time. Different from the simple backpropagation algorithm, backpropagation through time introduces an additional time dimension. It unfolds the network in time and calculates the loss in each time step according to the previous time steps. Total loss of the network is the summation of the losses calculated for each time step.

Furthermore, RNNs have the ability to produce an output at each time step. In some
cases though, e.g. text classification task in [31], only the network output from the last time step is utilized in further processing rather than the intermediate outputs.

In addition, most RNNs are also immune to variable input length. However, in practice, the training process of RNNs is very difficult, especially for longer sequences. As the backpropagation algorithm is propagated through time, repeated multiplication with small weights leads to extremely small gradients [32]. This problem is known as the vanishing gradient problem [33]. To address this issue, more complex RNN architectures, long short-term memory (LSTM) [34] and gated recurrent unit (GRU) [35], have been introduced. Besides the hidden states, both LSTM and GRU architectures introduce gate parameters to control the information flow. The idea behind the gate components in both architectures is to separate the information coming from the past by its relevance to the present or future processing step. Therefore, by forgetting the information that is not relevant anymore, LSTM and GRU overcome the vanishing gradient problem. Additionally, by keeping the information that is useful for future context, LSTM and GRU manage to capture long dependencies.

2.2 Transformer and attention

Transformer [9] is a deep learning model that was originally developed for machine translation tasks and has been proven to be quite powerful for sequence processing e.g., in [10] and [12]. Its main advantage over RNNs is that Transformer computes each input item independently, and thus allows parallel computation and constant path length between any two temporal positions in a sequence.

Figure 2.7 illustrates the original architecture of Transformer that is developed for machine translation tasks. As can be seen in the figure, it consists of encoder and decoder sides.
3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

where \(\text{Sublayer}(x)\) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position $i$ can depend only on the known outputs at positions less than $i$.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Figure 2.7. Transformer architecture adapted from [9]. The left block is the encoder side and the right block is the decoder side. $N$ stands for the number of encoder and decoder layers. In the case of machine translation, the inputs and outputs correspond to the input sequence of the source and target languages, respectively.

Composed of a stack of simple linear and feedforward network layers, along with attention mechanisms. Before explaining the overall architecture, this section first describes the key component of Transformer: multi-head attention layers that use attention functions.

A simple attention function can be defined as a mapping function that assigns importance weights between each item in two sequences. On the other hand, self-attention, sometimes referred to as intra-attention, computes the weights for a single sequence based on itself. Consequently, self-attention reflects how much one item in a sequence is related to the other items including itself in the same sequence, and computes the representation of the sequence based on the learned dependencies. Even though several popular attention functions are defined so far (e.g., [36], [37]), one of the most common ones, proposed with Transformer, is scaled dot-product attention [9] shown in Figure 2.8.

Computation of a single scaled dot-product self-attention output can be divided into six different parts:
Figure 2.8. Scaled dot-product attention. Adapted from [9]. Q, K, and V defines the query, key, and value matrices, respectively.

1. Generate three vectors, query $q_i$, key $k_i$, and value $v_i$ for input $x_i$, by using the respective weights $W^Q$, $W^K$, and $W^V$:

$$
q_i = W^Q x_i \\
k_i = W^K x_i \\
v_i = W^V x_i
$$

(2.9)

Query vector denotes the current focus of attention. Key vector defines the preceding items in the input sequence to compare them to the query. Lastly, value vector is used to compute the final representation as the weighted sum of query-key mappings.

2. Compute similarity score values by performing dot product of the query vector with the key vector of the items that are being compared to the current focus of attention:

$$
score(x_i, x_j) = q_i \cdot k_j
$$

(2.10)

3. In order to prevent the dot product values from getting larger and causing small gradients, divide the score values by the square root of the dimension of the key vectors, $\sqrt{d_k}$:

$$
score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}
$$

(2.11)

4. Pass the score values through the softmax function:

$$
score(x_i, x_j) = softmax\left(\frac{q_i \cdot k_j}{\sqrt{d_k}}\right)
$$

(2.12)

Softmax transforms the score values into probability distribution and the values lie...
between 0 and 1. Consequently, it ensures that the resultant values stay bounded. The score values represent how related two items \( x_i \) and \( x_j \) are.

5. Multiply value vectors by their corresponding softmax scores:

\[
c_{ij} = v_j * \text{score}(x_i, x_j)
\]  

(2.13)

The weighting operation amplifies the values of the items that are more related or similar with the current focus of attention whilst drowning out the irrelevant ones.

6. Take sum of the weighted values and produce the output for item \( x_i \):

\[
z_i = \sum_j c_{ij}
\]  

(2.14)

Instead of producing a single output at the given time step as described above, Transformer computes this entire process at once by exploiting matrix operations. First, it forms query (\( Q \)), key (\( K \)), and value (\( V \)) matrices by packing the input sequence together into a matrix \( X \) and rearranges the Equation 2.9 as follows:

\[
Q = W^Q X \\
K = W^K X \\
V = W^V X
\]  

(2.15)

Then, it simultaneously computes the self-attention across the entire sequence using the following equation:

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]  

(2.16)

Throughout this process, the score values passing through the softmax operation include the comparisons of query values to the key values that are in the future of the current item. This results in an undesirable effect during prediction tasks where the aim is to predict the upcoming item. In order to overcome this issue and allow the model to learn from its past, a mask passes on the score values during attention computation. The mask sets the score values of future items to \(-\infty\) before the softmax operation and consequently the corresponding values go out from the softmax as zeroed out. Figure 2.9 shows an example of this process.

Moreover, instead of using a single self-attention layer, Transformer uses multiple parallel self-attention layers whose outputs are called heads. In this multi-head self-attention architecture, each head utilizes its own query, key, and value matrices and therefore allows the model to capture the information from different representation sub-spaces [9]. In multi-head self-attention layers, the core process of self-attention computation remains
Figure 2.9. Illustration on attention mask where the sequence length is 3. The green cells $s_{ij}$ of the matrix on the left is the score values extracted on Equation 2.11 for each $q_i$ and $k_j$. Matrix in the middle is an attention mask to mask the values of the gray cells and pass the values of the orange cells. The matrix on the left is the output of softmax operation that will be multiplied by value matrix.

the same while the dimensions of the query, key, and value matrices, $d_q$, $d_k$, $d_v$, of each head $i$, are defined as follows:

$$d_q = d_k = d_v = \frac{d_{model}}{h}$$  \hspace{1cm} \text{(2.17)}

where $d_{model}$ represents the dimensionality of the overall output of multi-head attention layer. Next, the self-attention function is performed in parallel on each query, key, and value matrices and the heads are obtained. Finally, the heads are concatenated, yielding the $d_{model}$ dimensional output and multiplied by an additional weight matrix $W^O$:

$$head_i = Attention(W^Q_i X, W^K_i X, W^V_i X)$$

$$MultiHead(Q, K, V) = W^O Concat(head_1, ..., head_h)$$  \hspace{1cm} \text{(2.18)}

Figure 2.10 depicts the architecture of multi-head self-attention.

Figure 2.10. Multi-head attention with $h$ number of attention layers. Adapted from [9].

Even though self-attention architectures are mainly utilized for processing sequential data, they only manage to add context of the input items to the model and disregard their order.
in the sequence by processing them in parallel rather than sequentially. Transformer
overcomes this problem by adding positional encodings to the input embeddings. The
original Transformer architecture uses sinusoidal functions with varying frequencies as
positional encodings, that are independent of the sequence length:

\[
P E_{(\text{pos}, 2i)} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)
\]

\[
P E_{(\text{pos}, 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)
\]

where \(\text{pos}\) represents the position of the input in the given sequence and \(i\) is the in-
dex of the positional embedding dimension \(d_{\text{model}}\). Figure 2.11 illustrates an example of
positional embedding curves.

\[\text{positional embeddings}\]

![positional embeddings](image)

\[\text{Figure 2.11. Illustration on positional embeddings of Transformer where } d_{\text{model}} \text{ is 256 and sequence length is 200.}\]

As can be observed from the Figure 2.11, the difference between positional embeddings
of the input items temporally far apart from each other starts emerging already in the low
frequencies whereas for nearby words it remains similar up until the far higher frequen-
cies. Thereby, the usage of \(i\)-dimensional positional representation allows the sinusoidal
embeddings to capture the relative positional information in a sequence.

Transformer uses these positional embeddings by adding them into input embeddings and
produces the encoded representations of a given input sequence, which then contains the
context and position of each input item. In machine translation tasks, the encoded rep-
resentation that is fed into the encoder side consists of the sequence of words in source
language while it is in target language on the decoder side. Both input and positional
embeddings in encoder and decoder sides produce the same dimension as \(d_{\text{model}}\). Due
to Transformer being proposed for machine translation tasks, the architectures of original encoder and decoder blocks shown in Figure 2.7 are further developed for sequence to sequence mapping. Hence, hereafter, the usage of encoder and decoder blocks will be explained based on the case of machine translation task.

On the encoder side, the encoded representation of source sequence first goes into the multi-head self-attention layer, which will be referred as encoder self-attention (Figure 2.7). The encoder self-attention layer generates the attention scores of each item and adds this contextual information into encoded representations. The generated output is then fed to a feed-forward network that is applied to each position of the input items independently:

$$ FFN(x) = ReLU(xW_1)W_2 $$

where $x$ is the output of the multi-head self-attention layer. After stacking $N$ number of encoders, the output of the last encoder block is mapped into key and value matrices to be used in the decoder.

On the decoder side, the encoded representation of the target sequence is first fed into a masked multi-head self-attention layer, and it will be referred as masked decoder self-attention layer. In contrast to the encoder self-attention layer, decoder self-attention applies a mask in its self-attention operation to predict the upcoming word of the target sequence. After the context information of the target sequence is added to the representations, the output is passed to the last multi-head attention layer in Transformer, encoder-decoder attention. Unlike the self-attention layers, encoder-decoder attention comprises input connections from different sequences. As can be seen in the Figure 2.7, key and value come from the encoder side and query from the decoder side. Therefore, encoder-decoder attention produces a contextualized target representation that is computed as the similarity of each target word with the source words. Lastly, decoder feeds this representation into feed-forward network similar to the one in encoder side in Equation 2.20. In both encoder and decoder sides, output of each attention and feed-forward layers goes into layer normalization before passed to the next step. Finally, by using softmax operation, output of the $N$th decoder block is converted into word probabilities of the target sequence.

### 2.3 Predictive coding in speech representation learning

The concept of predictive coding is so far used in various fields. Predictive coding in signal transmission is one of the fundamental data compression techniques whose core idea is to transmit the difference between the true and predicted messages rather than the message itself [38], [39]. In neurocognition, several studies [40–43] show that the brain acts as a predictive function while learning from sensory signals during both interoceptive
and exteroceptive process. Additionally, Ylinen et al. suggested that predictive coding is beneficial in early language acquisition [44].

Predictive coding in the field of machine learning is a method of predicting future or missing information in an unsupervised manner, and it does not necessarily correspond to encoding of temporal differences as in the signal coding theory. Since these predictive coding models generate a supervisory signal from the provided data, they are also commonly referred to as self-supervised models. Predictive coding methods are mainly utilized for representation (feature) learning tasks. As the performance of most machine learning algorithms is strongly dependent on the input features, feeding good representations of the data into the algorithm makes the learning task easier. Therefore, it is not unusual to extract more advantageous representations with an unsupervised approach and use the extracted representation for downstream tasks. This technique is especially of use when unlabeled data is substantial but labeled data is insufficient to efficiently train these kinds of models without good representations. It is worth noting that the aforementioned feedforward neural networks in Section 2.1.2 can also be seen as a particular form of representational learning by sharing components and producing more useful, powerful, and distributed representations in its deep layers [16].

One of the most successful and famous works utilizing predictive coding is developed by Mikolov et al. [45], [46]. The method, commonly known as word2vec, either predicts the current word by looking at its surroundings or the surrounding words from the current word to learn word representations. In NLP tasks, it has become widely common to use word representations learned by word2vec architecture as input features of the model rather than traditional features, such as one-hot encoding e.g., [47], [48].

Furthermore, predictive coding methods have recently enjoyed increased adoption in the speech community. Arguably, the two most popular examples in speech representation learning tasks are autoregressive predictive coding (APC) [49] and contrastive predictive coding (CPC) [50]. Both models aim to learn speech representations by predicting many steps ahead in the future, and thus infer more global structures of speech rather than the local information. However, they differ from each other in terms of the loss function and consequently the information they try to encode in prediction.

APC architecture as illustrated in Figure 2.12 consists of three blocks: a PreNet block with an arbitrary number of fully-connected layers, followed by an RNN-based autoregressive block producing latent representations $z$, and a PostNet block of a 1D convolution layer at the end, which performs the prediction task. In the APC model, the aim is to predict the future accurately based on the history by using MAE, also known as L1 loss. The MAE
loss is defined as follows:

$$L_{APC} = \sum_{i=1}^{N-n} |x_{i+n} - \hat{y}_{i+n}|$$  \hspace{1cm} (2.21)

where $N$ is the number of frames (timesteps), $n$ is the number of steps to look ahead in the future and $x = \{x_1, x_2, ..., x_N\}$ and $\hat{y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_N\}$ are the input and predicted sequences, respectively. MAE aims to predict the labels directly.

CPC architecture is visualized in Figure 2.13 and includes two blocks: a feature encoder block mapping the input features to compact latent representations and an autoregressive layer summarizing the history into context latent representations to make predictions.
Originally, the encoder block comprises convolutional layers in [50]. However, its adaptations replace them with fully-connected layers (e.g., [49], [51]). CPC uses InfoNCE loss, where the aim is to maximize the mutual information between the target and predicted futures by discriminating the target future from the negative ones that are randomly sampled from the noise distribution. First, instead of encoding the high dimensional data distribution directly, it defines a density ratio $f$ by using a log-bilinear model:

$$f_k(x_{t+k}, c_t) = \exp(z_t^T W_k c_t)$$  \hspace{1cm} (2.22)

where $k$ is the prediction step. Then, the InfoNCE loss corresponds to the cross-entropy loss of the density ratio:

$$L_{CPC} = -\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}$$  \hspace{1cm} (2.23)

Contrarily from APC, CPC predicts the future latent features $z$ instead of input features $x$ by maximizing the mutual information between input and latent features.

### 2.4 Characteristics of speech signals

This section presents the main characteristics of speech. First, Section 2.4.1 gives an overview of how speech is generated. Then, Section 2.4.2 provides information on the phonetic representation of speech. Finally, Sections 2.4.3 and 2.4.4 define the basics of speech processing and spectral and acoustic features carried in speech signals, respectively.

#### 2.4.1 Speech production

Speech production refers to the process of converting the linguistic message into a speech signal [1]. The generation of speech at the physical level is modulation of airflow, controlled by changing the shape of the acoustic path by the process called articulation. The vocal organs laying on this acoustic path are known as articulators and can be seen in Figure 2.14.

In English, almost all sounds are produced by an airflow starting from the lungs, passing through the articulators, and finally outwards from the nose or mouth. Typically, lungs act as an energy source. Larynx follows the lungs and encloses the vocal folds along with the gap in between, called glottis. The sound source of voiced speech is the airflow pattern generated by the vocal folds. When the air is forced from the lungs, vocal folds vibrate and glottis starts periodically opening and closing resulting in voiced sound. Subsequently, it pushes the air to the vocal tract where pharynx and nasal and oral cavities are situated.
at. Glottis gets its widest during breathing. The shape of the vocal tract determines the sound by modifying the vibration of the vocal folds. Thus, to a certain extent, vocal tracts can be seen as waveguides.

2.4.2 Phonetic representation of speech

The speech signal is a continuous, acoustic waveform. Yet, in verbal communication, speech can be discretely represented by a finite set of symbols called the phonemes, combining together with words and further into sentences [1], [2]. Phonemes are the smallest linguistic units of speech and may change the meaning of a word. Additionally, in contrast to the word and sentence level description of speech, phonemes have no direct meaning and they consist of a smaller number of symbols. Phonemes are specific to an individual language and the number of phonemes mostly vary between 32 and 64 depending on the language [1]. In English there are about 40 phonemes.

The phonetic alphabet is usually classified into two main categories as vowels and consonants. Furthermore, consonants can be grouped in terms of voicing, place, and manner of articulation. All vowels are voiced sounds and generated from glottal source. On the other hand, consonants can be derived from different sound sources. Voicing refers to whether the speech sound is produced with vocal folds vibration or not [2]. The excitation signal for unvoiced sounds is determined by the airflow made at a constriction in the vocal tract [52]. For instance, consonants such as [z] or [v] are voiced whereas [f] and [s] are voiceless. Place of articulation defines the point where the airflow is restricted in the acoustic path such as dental consonant [t] where the constriction occurs between
the teeth. Manner of articulation (MoA) describes the configuration between the speech organs during speech production. Table 2.1 shows the English phonetic alphabet where the consonants are classified in terms of their manner of articulation.

Nasal consonants are produced by modifying the sounds in the oral cavity but releasing them through the nasal cavity. Plosives, also known as stops, occur when there is a complete blockage in the vocal tract. Plosive pairs such as [p] and [b] or [k] and [g] are distinguishable from each other by voicing. Fricatives are generated by creating a turbulence with a narrow constriction in the vocal tract. Combining plosives and fricatives, affricates are generated. Affricate consonants are plosives followed by friction. Lastly, approximants can be seen as a combination of vowels and fricatives. All approximants are voiced and the manner of the articulators in approximant generation resembles the vowel production [2]. Yet, the constriction in the vocal tract allows an air turbulence during the generation of approximants.

Table 2.1. English phonetic alphabet using ARPAbet and IPA phonetic symbols.

<table>
<thead>
<tr>
<th>Class</th>
<th>ARPAbet</th>
<th>IPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vowels</td>
<td>AA AE AH AO AW a ae a c w</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AY EH ER EY IY ay e er e i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OW OY UH UW o u u</td>
<td></td>
</tr>
<tr>
<td>Nasals</td>
<td>M N NG m n n</td>
<td></td>
</tr>
<tr>
<td>Plosives</td>
<td>B D G K P T b d g k p t</td>
<td></td>
</tr>
<tr>
<td>Fricatives</td>
<td>DH F HH S d f h s</td>
<td></td>
</tr>
<tr>
<td>Affricates</td>
<td>SH TH V Z ZH ʃ ə v z ʒ</td>
<td></td>
</tr>
<tr>
<td>Approximants</td>
<td>R L W Y r l w y</td>
<td></td>
</tr>
</tbody>
</table>

2.4.3 Pre-processing of speech

Due to the constant change in glottis and vocal tract during speech generation, speech signals are non-stationary and time variant [2], [52]. Hence, the most common initial step of speech processing is to split time-varying digital speech signals into short segments, also called speech frames. By doing so, each speech frame is assumed to be stationary [53]. Consequently, the properties of the signal do not change over the same segment. This process is known as windowing and speech segment $x$ in time-step $n$ is defined as follows:

$$x[n] = s[n]w[n]$$ (2.24)

https://www.speech.cs.cmu.edu/cgi-bin/cmudict

https://www.internationalphoneticassociation.org
where $s$ is the full waveform of speech and $w$ is the windowing function.

During the windowing process, splitting the signal results in abrupt cuts at the temporal borders and creates discontinuities between sequential speech segments. Consequently, this effect results in a phenomena called spectral leakage that creates artifacts in the frequency domain. In order to overcome this problem, smooth windowing functions are used e.g., Hamming or Hanning functions. As these smooth functions go to zero at the temporal borders, they introduce tapering to the windowed segments when multiplied (Equation 2.24). In consequence, the aforementioned discontinuities become negligible. Figure 2.15 illustrates a 25 ms long segment of both original and Hamming-windowed speech signal along with the Hamming function. As can be observed from the figure, the function modifies the signal to introduce smoothing. This modification results in information loss in the signal, especially near the temporal borders. The most common practice to furthermore counter this effect is to overlap the adjacent speech segments with a defined frame.
shift.

Even though short-time speech analysis is essential for accurate computations in speech analysis tasks, the choice of window size remains as a trade-off. Short windows are required to ensure that the analyzed speech segments are stationary. On the other hand, the window should be long enough to provide enough data points to capture the properties of the speech signal. For instance, one window should be long enough to include at least one period of speech to obtain information about the fundamental frequency [2]. In speech processing tasks, typical window lengths are 20–30 ms with 10–20 ms frame shift [52].

2.4.4 Spectral and acoustic features

In the scope of short-time speech analysis, typical frame-level acoustic features are short-time energy (STE), linear predictive coding (LPC) coefficients, fundamental frequency ($F_0$), and mel-frequency cepstral coefficients (MFCC).

Energy is one of the main continuous acoustic features and is used in several speech recognition tasks. Energy defines the envelope of a speech signal. Besides its being easy to compute, it is one of the main indicators of the arousal dimension of the emotion of the speaker [54]. Moreover, energy is a useful speech feature to distinguish voiced and unvoiced segments. Speech energy tends to be higher in voiced sections than the unvoiced sections and almost zero in silent frames [1, 55]. STE is calculated as the square of the amplitude of the signal $s$ represented in a waveform in speech segment $n$:

$$E_n = \sum_{m} (s[m]w[n - m])^2 \tag{2.25}$$

where $m$ defines the time-step of the windowing function.

Short-time speech energy provides information about voiced, unvoiced, and silent segments. The STE values in the voiced components are higher than those in the unvoiced components. Additionally, STE is almost zero in the silent frames. It is a major feature to distinguish the phonemes mainly in terms of their voicing.

Fundamental frequency ($F_0$) is one of the main prosodic features of speech. $F_0$, physically, refers to the vibration in the vocal folds during speech production and strongly corresponds to the voiced signals [56]. The range of $F_0$ varies between 80–400 Hz in adult males and 120–800 Hz in adult females [52]. $F_0$ estimation is often referred as pitch detection. However, it should be noted that pitch is a subjective term that defines the perceived fundamental frequency [57]. $F_0$ estimation is not a trivial task as the $F_0$ of a speech signal varies over a wide range and the periodicity might be resulted from a background noise [52]. There are different approaches for pitch tracking, e.g. based on
inverse filtering in [58], autocorrelation in [59], and the combination of autocorrelation and spectral information in [60].

In general, a speech signal additionally consists energy at other frequencies besides $F_0$. These frequencies are known as harmonic frequencies. $F_0$ is mainly referred as the first harmonic and the rest of the harmonics are the integer multiplies of $F_0$. For instance, for a speech signal with $F_0$ of 200 Hz, the second harmonic occurs at 400 Hz. Once the harmonic frequencies are radiated from vocal folds, the vocal tract filters the harmonics and modifies their amplitudes by changing the shape of the articulators situated at the vocal tract [2], [61]. The increase in the frequency amplitudes is called resonance. These resonance frequencies of the vocal tract often known as formants [2], [52]. Formants are observed as the peak frequencies in the spectrum and numbered with the order of their corresponding frequencies, e.g. the formant with the lowest frequency is $F_1$. Formants are mainly used to represent vowel sounds as they do not provide enough information to discriminate consonant identities [52]. Principally, several formant frequencies occur at the vocal tract. However, first two formants, $F_1$ and $F_2$, are enough to distinguish the vowels [2]. Additionally, higher formants ($F_4$, $F_5$) are often hard to observe as they are not very clear on the spectrum by often overlapping in the $F_3$ range. $F_1\times F_2$ plane for English vowels can be seen on the Figure 2.16.

![Figure 2.16](image)

**Figure 2.16.** The distribution of English vowels on $F_1$ and $F_2$ frequencies. Obtained from [62].

Arguably, the most common method to estimate formant peaks and bandwidths is linear prediction coding (LPC). The method is especially of use in speech compression and modelling due to its ability to well represent the speech production process [1], [52]. LPC is based on the assumption that the consecutive samples of a speech signal are correlated rather than being independent [63]. Hence, the method aims to represent a speech
sample \( s(n) \) as a prediction \( \hat{s}(n) \) by exploiting the knowledge of the previous samples \[62\]. The prediction is calculated as:

\[
\hat{s}(n) = - \sum_{k=1}^{m} \alpha_k s(n - k)
\]  

(2.26)

where \( n \) is the index of the sample, \( \alpha_k \) are the linear prediction coefficients, and \( m \) is the order of the predictor. Furthermore, the prediction error \( e(n) \) is defined as follows:

\[
e(n) = s(n) - \hat{s}(n)
\]  

(2.27)

\[
e(n) = s(n) + \sum_{k=1}^{m} \alpha_k s(n - k)
\]  

(2.28)

Subsequently, the optimum linear prediction coefficients are defined as the ones that minimize the mean square prediction error:

\[
\alpha = \underset{\alpha}{\text{argmin}} \sum_n e^2(n)
\]  

(2.29)

There are different methods to estimate the optimum linear prediction coefficients. Perhaps, the two mainly used ones are covariance and autocorrelation methods that were defined by Markel et al. in \[64\]. Once the prediction coefficients are obtained, the coefficients are transformed into z-domain. The roots of the resultant prediction polynomial correspond to the formants.

Mel-frequency cepstrum coefficients (MFCCs) are nowadays one of the most common features used in speech, speaker, and acoustic recognition problems \[52\]. MFCC computation was first introduced in \[65\]. After the short-time speech segments are obtained, the process of MFCC extraction pursues the following steps. First, discrete Fourier transform (DFT) is applied on speech segments. Then, the Mel filters are multiplied with DFT bins. Mel-scaling \[66\] is used to model the perception of human hearing to provide a better resolution in the lower frequencies. Hz to Mel conversion is defined as:

\[
\text{mel} = 2595 \times \log_{10}(1 + \frac{f}{700})
\]  

(2.30)

where \( f \) is the frequency in Hz. After the conversion to Mel-scale, the obtained filters are logarithmized. Finally, the discrete cosine transform (DCT) is performed to remove redundant information by decorrelating the dimensions.

The first Mel-frequency cepstrum coefficient represents the signal energy \[63\]. The coefficients approximately up until 13 mainly define the phonetic components of the speech.
The higher-order coefficients reflect speaker characteristics, i.e. pitch. Therefore, the choice of the number of MFCCs depends on the speech analysis task. For instance, in automatic speech recognition (ASR) tasks, the common practice is to use the first 13 coefficients.
3. METHODS

The main research goal of the present study was to develop a system to group and define attention heads in terms of the temporal behaviors of the heads and their relation with known characteristics of speech. Figure 3.1 displays the general overview of this system in data pre-processing, training, and analysis phases.

In the data pre-processing phase of the proposed system (Figure 3.1a), MFCC features are extracted from the training data that contain speech utterances. Then, the extracted features are split into training samples with an invariant length. In the training phase of Transformer-based models, two different predictive models, APC and CPC, are trained. The models take MFCC features of training samples as the input features and comprise...
of a fully connected layer followed by a Transformer layer. In APC, a 1D convolution layer follows the Transformer layer. Both models aim to predict the speech representation of future frames by using a self-supervised objective function.

In the analysis phase (Figure 3.1b), the trained models are applied to a test set. Then, the attention weights are extracted from the Transformer layer for each test sample. Next, 2D attention weights are converted into 1D attention scores for each attention head. After that, acoustic and linguistic features are extracted from the speech signals and phonetic annotations of the test corpus, respectively. Finally, the attention heads are analysed and grouped both temporally and with respect to the extracted speech features.

The rest of this chapter details the main methodology used in this thesis. First, Section 3.1 introduces the Transformer-based models underlying the present study. Then, Section 3.2 defines the speech features that are analyzed. Next, Section 3.3 explains the procedures followed for attention score extraction. Finally, Section 3.5 explains the metrics utilized on the evaluation of the heads on phoneme classification task.

### 3.1 Predictive coding models

As described earlier in Section 2.3, APC tries to predict the future input features accurately, whereas CPC aims to predict future compact latent features by maximizing the mutual information between input and latent features. In the present study, both CPC- and APC-based models are used to explore the attention heads that are trained with different learning strategies of the models.

For both predictive coding models, the implementations provided in [51] are followed but the autoregressive layers are replaced with a Transformer layer. For CPC, contrarily from the original implementation in [50], the encoder block of convolutional layers are replaced with fully connected layers.

As explained in Section 2.2, the original Transformer architecture with both encoder and decoder stacks is developed for sequence-to-sequence learning, specifically for machine translation tasks. In order to adapt the Transformer layer in speech representation learning, Transformer architecture was modified. Figure 3.2 visualizes the modified Transformer layer that is used in CPC- and APC-based models in the present study. The new architecture resembles the encoder side of the original Transformer structure. However, the predictive coding models, by their nature, do not rely on the future during training. Therefore, the adapted Transformer architecture furthermore utilizes the attention mask from the decoder side. In the modified architecture, the input of the Transformer is the output of the encoder and prenet layers of CPC and APC, respectively.
Figure 3.2. Architecture of the modified Transformer layer in predictive coding models for speech representation learning task. Input represents the output of encoder and prenet layers of CPC and APC, respectively.

3.2 Speech feature extraction

In speech feature extraction, acoustic and linguistic features are extracted from the speech signal and phonetic annotations of the test set, respectively. For the linguistic features, first, phonemes are grouped in terms of their manner of articulation (MoA), yielding seven categories: vowels, nasals, plosives, fricatives, affricates, approximants, and silence. Then, binary MoAs and phoneme boundaries are defined.

For each MoA category \( s \), speech frames are binarily grouped as follows:

\[
f_s(x_n) = \begin{cases} 
1, & \text{MoA}(x_n) = s \\
0, & \text{otherwise} 
\end{cases}
\]

(3.1)

where \( x \) is the speech signal, \( n \) is the time instant of \( x \) indicating the frame number, and \( \text{MoA}(x_n) \) is the manner of articulation of the speech frame \( x_n \).

Furthermore, phoneme boundaries are specified as the frames whose manner of articu-
lation is different than one of its adjacent frames in a ±20 ms window range:

\[
f_{\text{bound}}(x_n) = \begin{cases} 
1, & \exists k \left[ MoA(x_n) \neq MoA(x_{n+k}) \right] \\
0, & \text{otherwise}
\end{cases}
\]  

(3.2)

where \( k \in \left[ -\frac{20}{w}, \frac{20}{w} \right] \) and \( w \) is the frame size in milliseconds for each test sample.

For the acoustic features, speech energy, fundamental frequency \( F_0 \), and first three formants \( F_1, F_2, \) and \( F_3 \) are extracted for each speech frame. The properties and purposes of the selected acoustic features can be seen in Table 3.1.

**Table 3.1.** Properties and purposes of the selected acoustic features. The purpose explains the intended goal in the analysis under the assumption that the variation of the attentions can be explainable by the corresponding speech feature. Temporal domain in (T) and frequency domain in (F).

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Domain</th>
<th>Properties</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short-time energy</td>
<td>T</td>
<td>High in voiced components, low in unvoiced components, and almost zero in silent frames.</td>
<td>Analysis of the effect of energy on the attention heads when classifying vowels, silence, and voiced and unvoiced consonants.</td>
</tr>
<tr>
<td>( F_0 )</td>
<td>F</td>
<td>Observed in the existence of voiced speech. A rise-drop pattern in ( F_0 ) corresponds to boundaries.</td>
<td>If the observed variability that is explainable with ( F_0 ) in the heads (i) is biased towards the low or high ( F_0 ) frames (ii) corresponds to voiced and unvoiced frames and phoneme boundaries</td>
</tr>
<tr>
<td>Formants</td>
<td>F</td>
<td>A major indicator to distinguish vowel identities.</td>
<td>Analysis of the effects of the formants if the heads on which the formants are visible can be utilized to discriminate vowels.</td>
</tr>
</tbody>
</table>

### 3.3 Attention score extraction

Once the models are trained, they are applied to a test set and attention maps are obtained for each test sample from the multi-head attention layer of the modified Transformer.
Figure 3.3 shows the attention maps $A^h_u$ for each head $h$, visualized on the same 2 s long utterance $u$. The 2D attention maps in the figure represent how much a frame at the time instant of query $q$ attends to the frame on the time instant of key $k$.

![Attention Maps](image)

**Figure 3.3.** Attention maps of heads visualized on the same 2 s long utterance. Query and key axes stand for the number of frames where the frame size is 10 ms.

As multi-head attention layer applies a mask on the future frames during training, each frame in a sample can only access its past. As a result, the last frames in one sample can access relatively longer past whilst the first frames only reveal the fairly local dependencies. In order to prevent this bias, MFCC input, acoustic, and linguistic features are extracted with an overlap between test samples. 50% overlap is chosen to compensate for the trade-off between the continuity of the speech frames and the biases of the heads during prediction due to masking. By using this approach, it is ensured that each speech frame (except the ones in the first half of the first sample) has access to a speech history that is at least half the length of the speech sample.

In order to exploit the aforementioned benefits of the overlap, overlapped frames are masked in each test sample. The mask passes the attention values only where the time instant of a query is larger than half of the input sequence length. After the masking, attention map $A^h_u$ for a head $h$ on a $T$ second long sample $u$ is modified as follows:

$$A^h_u[q, k] = \begin{cases} A^h_u[q, k], & q \geq \frac{T}{2} \\ 0, & \text{otherwise} \end{cases}$$  (3.3)

Additionally, Figure 3.4 depicts this process. It should be noted that the $A^h_u[q, k]$ is already equal to zero where $q < k$ due to the attention mask.

In order to measure the relation between the attentions and speech features, the 2D
 attentions are projected into 1D space. For this purpose, two different 1D attention scores were explored; cumulative attention and masked cumulative attention.

Cumulative attention is calculated as the total attention values given to each time instant of key $k$:

$$
\text{cumSum}_h^k[k] = \sum_{k=0}^{T} A_h^k[k] \tag{3.4}
$$

Instead of presenting the relation of each query-key pair individually, cumulative sum represents how much a speech frame is attended over one sample and thus reflects the overall attention distribution of the complete sequence. Figure 3.5 shows the mapping of attention maps into cumulative sum.

In order to reveal the phonetic relations between query and key, masked cumulative sum is explored. The masked cumulative sum is calculated as the total attention values given
to each \( k \) by the speech frame \( q \) that consists of a specific MoA category \( s \). First, for each MoA category \( s \), the values of the given attention map \( A^h_u \) are masked if the \( \text{MoA}(x_q) \) is not the same as \( s \):

\[
A^h_{u,s}[q, k] = \begin{cases} 
A^h_u[q, k], & \text{MoA}(x_q) = s \\
0, & \text{otherwise}
\end{cases}
\]  

(3.5)

Using \( A^h_{u,s} \), the cumulative sum is calculated for each MoA category \( s \). Masked cumulative attention reflects the accumulated attention that is given to each speech frame by the frames that consist of the same and specific MoA category. Therefore, in contrast to standard cumulative attention, masked cumulative attention distinguishes the attention scores by the phonetic features of the query. For instance, \( A^h_{u,\text{vowel}} \) consists of the attention weights of head \( h \) when the time instant of query corresponds to a vowel. Figure 3.6 shows the computation of \( A^h_{u,\text{vowel}} \). After \( A^h_{u,s} \) is defined, masked cumulative sum \( \text{cumSum}_{s}^h \) is calculated as standard cumulative sum (Equation 3.4) for each MoA category \( s \).

![Figure 3.6. Forming the new attention matrix \( A_{\text{vowel}} \).](image)

After both \( \text{cumSum}_{u,s}^h \) and \( \text{cumSum}_{u}^h \) are obtained, they are concatenated to define the attention score of heads for the whole test set with \( N \) number of samples:

\[
\text{cumSum}^h = \text{cumSum}_{u,0}^h \oplus \text{cumSum}_{u,1}^h \oplus \cdots \oplus \text{cumSum}_{u,N}^h
\]

\[
\text{cumSum}_{s}^h = \text{cumSum}_{u,0,s}^h \oplus \text{cumSum}_{u,1,s}^h \oplus \cdots \oplus \text{cumSum}_{u,N,s}^h
\]

(3.6)

### 3.4 Grouping the heads

Once the attention scores and speech features are extracted, attention heads are grouped by using three different methodologies:
1. Dependence analysis between speech features and attention scores
2. Linear regression analysis between speech features and attention scores
3. Defining the temporal behavior of the heads

In statistics, there are two main methodologies of measuring the dependence between two random variables: mutual information and Pearson and Spearman correlation coefficient. Mutual information (MI) indicates how much of the uncertainty of one random variable can be reduced by observing the other. MI can capture any relationships between the variables. Pearson’s correlation coefficient (PCC), on the other hand, measures the linear association between two variables, and, in contrast to MI, PCC delivers the information about the polarity of the relationship and thus provides a clearer interpretation [67].

In addition, PCC is computationally more efficient than MI. Due to the advantages of one over another, MI and PCC commonly provide complementary information on each other.

In the present work, dependence analysis was conducted by adopting Pearson’s correlation coefficient \( r \) that allows to further explain the direction of the relationship between the attention scores and the speech features. It measures the strength and direction of the linear relation between two variables \( x \) and \( y \) as follows:

\[
r(x, y) = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_i (x_i - \mu_x)^2 (y_i - \mu_y)^2}}
\]

(3.7)

where \( \mu_x \) and \( \mu_y \) are the mean values of variables \( x \) and \( y \), respectively. Pearson’s correlation is a commutative operation, leading to \( r(x, y) = r(y, x) \), and it varies between \(-1\) and \(1\). A high absolute correlation indicates an existence of a strong linear relationship between the variables. A negative correlation implies a negative association where the variables tend to change in opposite directions whilst variables change in the same direction in the case of a positive correlation. By using the equation above, the correlation coefficient between the speech features and both standard (\( \text{cumSum}^h \)) and masked (\( \text{cumSum}^h_s \)) cumulative sum for each head \( h \) and MoA category \( s \) is measured as follows:

\[
r_{hp}^h = r(\text{cumSum}^h, x_p)
\]

(3.8)

\[
r_{hp,s}^h = r(\text{cumSum}_{s}^h, x_p)
\]

(3.9)

where \( x_p \) is the extracted speech feature of the test signal \( x \) for each feature category \( p \) defined in Section 3.2. For instance, \( r_{\text{energy}}^1 \) corresponds to the correlation coefficient between energy and the cumulative sum of the first head. On the other hand, \( r_{\text{vowel,nasal}}^1 \) defines the correlation coefficient between binary represented nasals and the cumulative sum of the first head where only the attention values given from the frames that are composed of vowels are taken into account. For the acoustic features and phone
boundaries, the correlation coefficient measured against the cumulative sum in Equation 3.8 are reported to explain their effect on the overall change in the cumulative sum. For the MoA, the correlation coefficients against masked cumulative sum in Equation 3.9 are obtained to observe whether the relation between the MoA of two frames underlies the attention distribution of the heads. Furthermore, as the existence of one MoA category in a speech frame implies the absence of the others, it is reflected in the correlation coefficients. Hence, only the $r_{p,s}^h$ values where $p$ and $s$ correspond to same MoA category are reported.

Furthermore, in addition to the correlation analysis, a multiple linear regression model is applied between the speech features and attention scores. In contrast to correlation, linear regression analysis aims to estimate parameters in a linear equation and use the estimated parameters for the prediction of the outcome based on the given independent variables:

$$y = b_0 + \sum_{p=1}^{M} b_p x_p$$  \hspace{1cm} (3.10)

where $b_0$ is bias or often referred to as intercept and $b_i$ is the slope coefficient for each independent variable $x_i$ given $M$ variables. In our analysis, $x_i$ represents the speech features of the test data for each feature $p$ and $y$ is the cumulative sum for each head. Thereby, $b_{\text{energy}}^h$ defines the change in the cumulative sum of a head $h$ with speech energy. Both correlation and linear regression analysis define the behavior of the heads with respect to the concurrent speech features.

Moreover, the temporal behavior of the heads is analyzed independent of the speech features. For this purpose, each head $h$ is defined as an attention-distance function for each sample $u$ by using the following equation:

$$f_u^h[d] = \sum_{q=T/2}^{T} A_u^h[q, q-d]$$  \hspace{1cm} (3.11)

where $d \in [0, T/2]$ is the distance from the current frame. The resultant function $f$ defines the total attention given to the frame that is temporally $d$ steps behind the current frame of the test sample $u$. Then, the center of mass $j_u^h$ of the attention-distance functions for each sample $u$ is calculated as follows:

$$j_u^h = \arg\min_j \left[ \sum_{d=0}^{j} f_u^h[d] \geq \left( \sum f_u^h \right) / 2 \right]$$  \hspace{1cm} (3.12)

By using $j_u^h$, temporal behavior of the heads are defined as the mean distance $\mu_h$ and standard deviation $\sigma_h$ of the distances for each head $h$ across the samples:
where \( N \) is the total number of samples. The mean distance represents, on average, how many frames the attention concentration over a sample is behind the current frames. The standard deviation explains how spread-out the attention distances across the samples are. Thus, temporal behavior allows further explanation on the behavior of the heads. For instance, small mean distance indicates a learned phonetic information as the neighbor frames tend to have same sound. On the other hand, approximately zero standard deviation might suggest that these heads are better at capturing the positional information rather than individual speech features as the mean distance do not vary across different utterances.

3.5 Evaluation of the heads on phoneme discrimination

After extracting the head scores, the attention heads in both APC and CPC models are evaluated. Since phonemic information is one of the most important characteristics that distinguish two different utterances from each other [49], the evaluation is done by the phoneme classification performance of the models. In order to explore the Transformer layer of both APC and CPC, the representations learned in the Transformer layer were passed to a feed-forward network with softmax activation function that predicts the phoneme classes for each frame. Furthermore, forward and backward feature selection methods are adapted to assess the importance of each head as well as their combinations. For a model \( M \in \{\text{APC, CPC}\} \), \( M_{+h_i,h_j} \) represents the model when only the heads \( h_i \) and \( h_j \) are utilized and the other heads are pruned whereas \( M_{-h_i,h_j} \) is when \( h_i \) and \( h_j \) are subtracted from the model \( M \). Additionally, \( M_{\text{none}} \) corresponds to the model when none of the heads contribute to the model. In this case, the Transformer layer of \( M_{\text{none}} \) adds the positional embedding to the input and directly passes the resultant representation by skipping the multi-head self-attention layer (see Figure 3.2).

Forward selection starts with empty subset \( M_{\text{none}} \). Then, it reports the accuracy of \( M_{+h_i} \) for each individual head \( h_i \in [h_0, h_{m-1}] \) where \( m \) is the number of attention heads. Following this, it selects the head with the best accuracy, denoted as \( h_1 \), and generates the model \( M_{+h_1} \) to be combined with each remaining individual head. Next, it selects the next head that improves the accuracy of the new model \( M_{+h_1} \) the most, \( h_2 \), and forms the model \( M_{+h_1,h_2} \) for the next iteration. The algorithm iteratively continues this process until either the next selected head does not show any significant improvement to the accuracy.
or the full model $M$ is composed.

Backward selection, on the other hand, starts with the full model $M$ which includes all the attention heads. It then prunes each individual head and reports the accuracy of the model for each head, $M_{-h_i}$. Afterward, it discards the head that decreases the accuracy the least when pruned, denoted as $h^{-1}$. Next, by using the $M_{-h^{-1}}$, it again reports the accuracy of the remaining heads when removed and iteratively continues the process until either all heads that drop the accuracy are removed from the model or the empty subset $M_{none}$ is reached.
This chapter introduces the experiments conducted. First, Section 4.1 describes the datasets that are used in the experiments. Then, Section 4.2 explains the experimental setup of both training and analysis pipelines.

4.1 Datasets

This section provides an overview of the datasets used to train and analyze the models. Sections 4.1.1 and 4.1.2 describe the speech audio files and text transcriptions of the speech corpus, respectively.

4.1.1 LibriSpeech

LibriSpeech\[1\] is a corpus of audiobooks of 16kHz read English speech [68]. The entire collection is approximately 1000 hours and was derived from a larger corpus of unaligned audiobooks by performing a two-stage alignment and segmentation process. First, the two-stage alignment phase in [68] produced a subset of audio segments no longer than 35 seconds, then filtered out the utterances whose decoding are deviated from the transcript. Then, the authors in [68] performed the segmentation phase. In the segmentation phase, the resultant utterances that consisted of silence longer than 0.3 seconds were split for the training set. The utterances whose silent interval corresponded to a sentence break were split for test and development sets.

The authors in [68] partitioned the corpus into two subsets: “clean” and “other” (noisy) according to the speakers’ word error rate (WER) which defines the number of errors including additional, replaced, and omitted words divided by the total number of words. Speakers with lower WER were moved to clean and those with higher WER to noisy subsets. For the clean data, two training sets were defined with approximately 100 hours (train-clean-100) and 360 hours (train-clean-360). For noisy data, a single training set of approximately 500 hours (train-other-500) was formed. Test and development sets of both subsets are approximately 5 hours. Table 4.1 shows the detailed information of all subsets. In our experiments, training, validation, and test data were chosen as “train-

\[1\]https://www.openslr.org/12
clean-100", "dev-clean", and "test-clean" subsets, respectively.

<table>
<thead>
<tr>
<th>subset</th>
<th>hours</th>
<th>per-speaker minutes</th>
<th>female speakers</th>
<th>male speakers</th>
<th>total speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev-clean</td>
<td>5.4</td>
<td>8</td>
<td>20</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>test-clean</td>
<td>5.4</td>
<td>8</td>
<td>20</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>dev-other</td>
<td>5.3</td>
<td>10</td>
<td>16</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>test-other</td>
<td>5.1</td>
<td>10</td>
<td>17</td>
<td>16</td>
<td>33</td>
</tr>
<tr>
<td>train-clean-100</td>
<td>100.6</td>
<td>25</td>
<td>125</td>
<td>126</td>
<td>251</td>
</tr>
<tr>
<td>train-clean-360</td>
<td>363.6</td>
<td>25</td>
<td>439</td>
<td>482</td>
<td>921</td>
</tr>
<tr>
<td>train-clean-500</td>
<td>496.7</td>
<td>30</td>
<td>564</td>
<td>602</td>
<td>1166</td>
</tr>
</tbody>
</table>

4.1.2 LibriSpeech alignments

LibriSpeech alignments\(^2\) contain word- and phoneme-level alignments of the LibriSpeech corpus. Lugosh et al.\(^69\) obtained these alignments using Montreal Forced Aligner\(^70\) to pre-train a speech model for end-to-end speech language understanding. Forced alignment is a technique to synchronize text transcription of speech with the corresponding audio file using an automatic speech recognizer.

Every word and phoneme in the alignments in\(^69\) contain their start and end time in the corresponding sequence. In total, there are 73 distinct phonemes defined. Each vowel was additionally classified into three groups according to its word stress; no stress, primary stress, and secondary stress. Silence was defined into four groups depending on its position in the sequence or whether it represents silence or non-speech vocalizations such as breathing.

4.2 Experimental setup

This section presents the experimental setup of the methods explained in Chapter 3. First, Section 4.2.1 implements the predictive coding models and phoneme classifier. Then, Section 4.2.2 explains the setup used for the extraction of speech features on LibriSpeech data. In all the experiments, Python version 3.7 was used to develop the code. The models were implemented by using TensorFlow library version 2.4. Speech feature extractions were done by using Librosa and Praat libraries.

\(^2\)https://zenodo.org/record/2619474
4.2.1 Settings of the predictive coding models and phoneme classifier

For both APC and CPC models, 39 MFCC (13 static + ∆ + ∆∆) coefficients were chosen as input features where the MFCC coefficients are normalized to zero mean at utterance level. The features were extracted with a window length of 25 ms and a window shift of 10 ms. The data was split into 2 s samples with 50% overlap.

Both models followed the same Transformer setup; one Transformer layer with 8 attention heads. The dimensionality of the latent space (corresponding to $d_{\text{model}}$ presented in Section 2.2) was 256 and of the inner layers of feed-forward layer in Transformer layer was 512. 20% dropout was used between hidden layers.

![Learning curves](image.png)

**Figure 4.1.** Epoch-loss learning curves. Training and validation data are train-clean-100 and dev-clean subsets of LibriSpeech corpus, respectively.
For APC, first, PreNet consisted of 3 layers with 128 dimensions of fully connected layer with ReLu activations and 20% dropout in between, following one Transformer block, and PostNet with 1D convolutional layer with the kernel size of 1 at the end. The learning rate was set to $10^{-4}$ and the model predicted 5 frames ahead. For CPC, one fully connected layer with 256 dimensions with ReLu activation was followed by one Transformer block. For the loss calculation, 10 negative samples were drawn from the batch. The learning rate was set to $10^{-3}$ and the model predicted 12 frames ahead. Both models were trained with Adam optimizer and a batch size of 32. APC model was trained with 100 epochs whereas CPC model was trained with 40 epochs due to computational limitations, yet no underfit was observed. Figure 4.1 shows the learning curves of the models.

In order to allow the model to be analysed by the contribution made by each individual head as described in Section 3.5, a masking gate with the length of the number of attention heads was passed as input to the model.

For the phoneme classifier, the parameters of the trained models were frozen and one fully connected layer with softmax activation on top of the Transformer layer was added. The learning rate was set to $10^{-3}$ and the models were trained with SGD optimizer and a batch size of 32 for 15 epochs using categorical cross-entropy loss. The accuracy results of Transformer-based APC and CPC models were compared with three baselines; random initialization, MFCC features, and APC and CPC using autoregressive models. For APC and CPC with autoregressive layer, the implementation provided by [51] was followed and one RNN layer was used in order to replace one layer of Transformer.

4.2.2 Setup of extraction of speech features

For the linguistic features of the speech corpus, phoneme-level transcription of the dataset was used. In order to align the phoneme annotations with their corresponding frames in the input and acoustic features, phoneme-level intervals were designated into 10 ms long parts to which they belong. Additionally, 73 phonemes were merged into 40 classes by discarding the word stress of the vowels and the variety in the silent frames. Phonemes were defined as categorical variables. Furthermore, phonemes were classified based on their manner of articulation, resulting in 7 different MoAs: vowels, nasals, plosives, fricatives, affricates, approximants, and silence, as described in Section 3.2. Similar to individual phonemes, MoAs were also defined as categorical variables and further represented in binary forms. Additionally, phoneme boundaries were specified as the frames whose manner of articulation is different than of one of its adjacent frames in a ±20 ms range, which corresponds to ±2 frames in the current setup.

For the acoustic features speech energy, pitch, and first three formants were extracted for each speech frame, corresponding to 10ms.
5. RESULTS

In this chapter, the results of the experiments conducted are presented and discussed. First, the results on overall head behavior are reported in Section 5.1. Then, in Section 5.2, phoneme classification results are presented. Finally, the overall results are summed up and discussed in Section 5.3.

5.1 Overall head behavior

Figure 5.1 visualizes the attention weights on the same 2 s long utterance for APC and CPC models. The rest of this section reports the results of the overall head behavior on the LibriSpeech data for both CPC and APC models. Section 5.1.1 presents the temporal behavior of the heads. Section 5.1.2 analyzes the behavior of the heads with respect to the speech features.

(a) CPC model
Hereafter, the heads will be grouped by temporal, correlation, and regression analysis and according to the findings, they will be referred as:

(i) diagonal heads whose mean attention distance is below 5 frames and denoted as $(d)$
(ii) heads whose mean attention distance is above 5 frames with zero-deviation and denoted as $(p)$
(iii) acoustic heads whose cumulative sum has a relatively higher correlation with the acoustic features (excluding the phonetic features) and denoted as $(a)$

### 5.1.1 Temporal behavior of the heads

As can be seen in Figure 5.1, APC model produces heads with temporally less spread-out attention than CPC model. This property of APC is furthermore reflected in the results of temporal behavior analysis.

Figure 5.2 depicts the attention-distance plotted for a 2 s long utterance for APC and CPC models. In addition, Table 5.1 presents the results on the temporal behavior of the heads $M_h$ where $M \in \{\text{APC, CPC}\}$ and $h$ is the index of the attention head. As can be seen in the table and the figure, temporal mean distance and standard deviation of the APC heads stay within a narrower range than the CPC heads. Additionally, except for
APC heads produce approximately 0-deviation across the samples whereas only CPC\(_0\), CPC\(_1\), and CPC\(_5\) display the same behavior in the CPC model. The remaining CPC heads, CPC\(_2\), CPC\(_3\), CPC\(_4\), CPC\(_6\), and CPC\(_7\) still produce more deviation than the only non-zero deviated head of APC, APC\(_0\).

Figure 5.2. Attention-distance plots on a 2 s long utterance. The attention weights of the chosen utterance correspond to the Figure 5.1.

Furthermore, the results show that APC\(_1\), APC\(_2\), APC\(_6\), APC\(_7\), and CPC\(_0\) always concentrate their cumulative attention at the same distance across the test samples. Since the temporal behavior of these aforementioned heads does not change across different utterances, these heads with zero standard deviation might be influenced by positions
Table 5.1. The mean distance and standard deviation of the heads across the samples. Diagonal heads in (d), heads looking into further past with zero-deviation in (p).

<table>
<thead>
<tr>
<th>(a) CPC model</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>mean distance</td>
<td>deviation</td>
</tr>
<tr>
<td>CPC₀ (d)</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>CPC₁ (d)</td>
<td>2.23</td>
<td>0.44</td>
</tr>
<tr>
<td>CPC₂</td>
<td>33.27</td>
<td>12.7</td>
</tr>
<tr>
<td>CPC₃</td>
<td>42.43</td>
<td>12.18</td>
</tr>
<tr>
<td>CPC₄</td>
<td>28.8</td>
<td>13.11</td>
</tr>
<tr>
<td>CPC₅ (d)</td>
<td>2.65</td>
<td>0.95</td>
</tr>
<tr>
<td>CPC₆ (d)</td>
<td>4.96</td>
<td>5.39</td>
</tr>
<tr>
<td>CPC₇</td>
<td>39.93</td>
<td>7.42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b) APC model</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>mean distance</td>
<td>deviation</td>
</tr>
<tr>
<td>APC₀</td>
<td>24.87</td>
<td>3.96</td>
</tr>
<tr>
<td>APC₁ (d)</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>APC₂ (p)</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>APC₃ (d)</td>
<td>2</td>
<td>0.02</td>
</tr>
<tr>
<td>APC₄ (d)</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>APC₅ (p)</td>
<td>14</td>
<td>0.03</td>
</tr>
<tr>
<td>APC₆ (p)</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>APC₇ (p)</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

more than individual speech features. On the other hand, the heads CPC₂, CPC₃, CPC₄, CPC₇, and APC₀, which produce more spread attention over a sample (see Figure 5.1), not only aggregate their cumulative attention in the further past but also display temporally deviated behavior across the samples.

5.1.2 Analysis of the heads on the speech features

Figures 5.3 and 5.4 show the results for correlation analysis using the standard and masked cumulative sum, respectively. In addition, the distribution of the sounds for each head in the cumulative attention for both models is represented in Appendix A. The correlation coefficients between the standard cumulative attention scores and phoneme boundaries and acoustic features are shown in Figure 5.3. As can be seen in the figure, the heads of the CPC model display varying behavior between each other. On the other hand, the heads of the APC model are similar to each other. Additionally, diagonal heads in both models, such as CPC₀ and APC₁ do not display a linear relationship with the acoustic features. This suggests that these heads are learning to focus on different speech features than energy, pitch, or phone boundaries. On the other hand, APC₀, CPC₂, CPC₃, CPC₄, and CPC₇ demonstrate a weak to moderate correlation with the acoustic features (0.15 < |r| < 0.40). For instance, correlation value of –0.32 between energy and CPC₇ corresponds to a coefficient of determination ($R^2$) of 0.1. This implies that 10% of the variability of the cumulative sum can be explained with the speech energy and overall, CPC₇ tends to accumulate its attention in low energy frames. Contrarily, CPC₅ shows the opposite relation with energy and pitch, suggesting that its attention tends to be concentrated in high-energy frames. For the APC model, the only head that shows a considerable level of association with acoustic features is reported as APC₀ and its negative correlation with energy indicates that the overall attention of APC₀ gets larger on the frames with low en-
energy. Those heads that display a noticeable relation with acoustic features, APC₀, CPC₂, CPC₃, CPC₄, CPC₅, and CPC₇, will be referred to as acoustic heads. It should be recalled that these acoustic heads display both visually and quantitatively deviated and temporally spread-out attention. Additionally, except CPC₅, they tend to attend to long past with the mean distance above 20 frames. On the other hand, CPC₅ being the only diagonal and acoustic head suggests that it is learning to focus on local acoustic features to generate its representation.

Figure 5.3. Correlation coefficients between speech features and standard cumulative sum. Diagonal heads in (d), acoustic heads in (a), heads attending to past frames (more than 5 frames) with zero-deviation in (p).

Figure 5.4 shows correlation coefficients between masked cumulative sum and binary-represented manner of articulations. The figure reports the correlation values of $r_{p,s}$ only where the MoA category that is used to compute masked cumulative sum $s$ and the binary represented MoA $p$ are the same (see Section 3.4). For instance, for vowels, the correlation between binary represented vowels and the cumulative sum of the head that is given from vowels is reported. In this case, a high positive correlation between a MoA and attention score indicates that the frames tend to attend to the ones that consist of the same sounds.

As can be observed from Figure 5.4, all the heads in both models are sensitive to silent frames, suggesting that each head has the capacity to distinguish between speech and non-speech frames. On the other hand, phonetic selectivity of the CPC heads varies depending on the sound itself, whereas of the APC heads remain more stable regardless of the sound. For instance, the correlation coefficient between CPC₄ and MoA shows significant change depending on the MoA category (e.g. –0.073 with vowels and 0.28 with fricatives) whilst there is no such discernible variation in APC heads. For the CPC model,
Figure 5.4. Correlation coefficients between phoneme identities and masked cumulative sum. Diagonal heads in (d), acoustic heads in (a), heads attending to past frames (more than 5 frames) with zero-deviation in (p).

The high correlation between each MoA and attention scores of CPC_{0}, CPC_{1}, and CPC_{6} implies that each sound tends to attend to the same sound. Differently, CPC_{5} shows a high correlation only with approximants and vowels. This behavior supplements the findings of the positive association between CPC_{5} and energy since vowels and voiced consonants such as approximants carry higher energy than other sounds in a speech signal. In contrast, CPC_{7} shows a negative correlation with vowels, denoting that the frames consisting of vowel sound tend not to attend to past vowels. This behavior of CPC_{7} potentially implies that the attention concentration on low energy frames, which is shown in previous findings, is related to high energy frames choosing to focus on low energy frames. Differently from CPC, APC heads display similar patterns with each other, except APC_{0}. APC_{0}, an acoustic head, produces distributed correlation values depending on the MoA category similar to the other acoustic heads in the CPC model such as CPC_{5} and CPC_{7}. APC_{1}, APC_{3}, and APC_{4} can be easily grouped together by displaying a high correlation with each MoA category. On the other hand, attentions in APC_{2}, APC_{5}, APC_{6}, and APC_{7} do not show notable preference to the frames that are composed of the same sound as the current frame since their correlation values remain relatively weaker.

These results initially point out that the aforementioned heads that produce a high positive correlation with the manner of articulations might be better at distinguishing different sounds. Yet, it should be noticed that all of these heads, CPC_{0}, CPC_{1}, CPC_{6}, APC_{1}, APC_{3}, and APC_{7} have a mean distance of below 5 (Table 5.1). This indicates that the remarkably higher phonetic selectivity of these heads potentially is resulting from the accumulated attention in neighbor frames which correspond to the same sound as of the
current frame. Acoustic heads such as CPC5, CPC7, and APC0 are mostly related to speech energy yet this is furthermore reflected in their phoneme selectivity. Lastly, the low correlation values of APC2, APC5, APC6, and APC7 with both acoustic and phonetic features suggest that these approximately 0-deviated heads that attend to longer past (denoted as $z$) might be learning only from relative positions of the frames as the temporal behavior of these heads do not vary across different utterances.

Additional to the correlation analysis, regression results along with the $R^2$ values are presented in Figure 5.5. The results are obtained between the speech features and the standard cumulative sum. Supporting the findings of correlation and temporal behavior analysis, CPC heads can be grouped with different combinations of speech features.

On the other hand, the only acoustic head of APC, APC0, can be put into a different category than other APC heads. In CPC, the acoustic heads CPC5 and CPC7 produce relatively higher r-square values, indicating that 12% and 15% of the variability of the cumulative sum can be explained with the extracted speech features. On the other hand, results show that APC heads, including APC0, cannot be reliably modeled with linear regression. This might be arguably resulting from the 0-deviation in the APC heads. As 0-deviation indicates that the temporal behavior of the heads does not change across different utterances, these heads might be reflecting positional information rather than a varying acoustic or phonetic representation.

**Figure 5.5.** Linear regression between speech features and standard cumulative sum.

### 5.2 Phoneme classification results

The results for the phoneme classification experiments are presented in this section. First, the accuracy results of Transformer-based APC and CPC models are compared with three
baselines; random initialization, MFCC features, and APC and CPC using autoregressive models, and the results are shown in Table 5.2.

Table 5.2. LibriSpeech phoneme classification results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy [0–1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random initialization</td>
<td>0.03</td>
</tr>
<tr>
<td>MFCC features</td>
<td>0.41</td>
</tr>
<tr>
<td>APC-rnn</td>
<td>0.56</td>
</tr>
<tr>
<td>CPC-rnn</td>
<td>0.52</td>
</tr>
<tr>
<td>APC-transformer</td>
<td>0.48</td>
</tr>
<tr>
<td>CPC-transformer</td>
<td>0.54</td>
</tr>
</tbody>
</table>

The accuracy results of the phoneme classification task with the first iteration of forward and backward selection that is introduced in Section 3.5 are reported in Tables 5.3 and 5.4. Forward and backward selection represent iterative addition and subtraction of the individual heads in the model, respectively. For model $M$, $M_{\text{none}}$ corresponds to the model where all the heads are pruned. It should be recalled that the Transformer layer in $M_{\text{none}}$ adds the positional embedding to the input and directly passes the resultant representation to the feed-forward layer by skipping the multi-head self-attention (see Figure 3.2). $M_{-h_i,h_j}$, where $h_i$ and $h_j$ are the heads of model $M$, represents the model when $h_i$ and $h_j$ are pruned. $M_{+h_i,h_j}$ corresponds to using only $h_i$ and $h_j$, and pruning the others.

Table 5.3. Accuracy results of individual heads. $M_{\text{none}}$ is where all the heads are pruned. $M_{+h_i}$ is where only head $h_i$ is included in the model. Accuracy$_1$ represents the accuracy of phoneme classification (40 classes). Accuracy$_2$ represents the accuracy of MoA classification (7 classes). Best accuracy is reported in bold. (a), (d), and (p) correspond to the acoustic heads, diagonal heads, and the heads looking into further past, respectively.

(a) CPC model

<table>
<thead>
<tr>
<th>head</th>
<th>accuracy$_1$</th>
<th>accuracy$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPC$_\text{none}$</td>
<td>0.489</td>
<td>0.696</td>
</tr>
<tr>
<td>CPC$_{+0}$ (d)</td>
<td>0.526</td>
<td>0.729</td>
</tr>
<tr>
<td>CPC$_{+1}$ (d)</td>
<td>0.498</td>
<td>0.705</td>
</tr>
<tr>
<td>CPC$_{+2}$ (a)</td>
<td>0.491</td>
<td>0.697</td>
</tr>
<tr>
<td>CPC$_{+3}$ (a)</td>
<td>0.490</td>
<td>0.697</td>
</tr>
<tr>
<td>CPC$_{+4}$ (a)</td>
<td>0.489</td>
<td>0.696</td>
</tr>
<tr>
<td>CPC$_{+5}$ (a,d)</td>
<td>0.496</td>
<td>0.704</td>
</tr>
<tr>
<td>CPC$_{+6}$ (d)</td>
<td>0.469</td>
<td>0.687</td>
</tr>
<tr>
<td>CPC$_{+7}$ (a)</td>
<td>0.490</td>
<td>0.696</td>
</tr>
</tbody>
</table>

(b) APC model

<table>
<thead>
<tr>
<th>head</th>
<th>accuracy$_1$</th>
<th>accuracy$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>APC$_\text{none}$</td>
<td>0.235</td>
<td>0.460</td>
</tr>
<tr>
<td>APC$_{+0}$ (a)</td>
<td>0.221</td>
<td>0.458</td>
</tr>
<tr>
<td>APC$_{+1}$ (d)</td>
<td>0.351</td>
<td>0.580</td>
</tr>
<tr>
<td>APC$_{+2}$ (p)</td>
<td>0.254</td>
<td>0.449</td>
</tr>
<tr>
<td>APC$_{+3}$ (d)</td>
<td>0.143</td>
<td>0.395</td>
</tr>
<tr>
<td>APC$_{+4}$ (d)</td>
<td>0.137</td>
<td>0.284</td>
</tr>
<tr>
<td>APC$_{+5}$ (p)</td>
<td>0.265</td>
<td>0.499</td>
</tr>
<tr>
<td>APC$_{+6}$ (p)</td>
<td>0.288</td>
<td>0.477</td>
</tr>
<tr>
<td>APC$_{+7}$ (p)</td>
<td>0.301</td>
<td>0.525</td>
</tr>
</tbody>
</table>

As can be seen in Table 5.3, phoneme accuracy results of individual heads show more dis-
crepancy in APC than in CPC. Arguably, the diversity of the heads within the CPC model allow each head to learn from different features that are useful for phoneme discrimination from different aspects and thus balance the accuracy. In both models, the highest accuracy belongs to the diagonal heads with temporal zero deviation. Additionally, in both APC and CPC, there is at least one head whose model gives lower accuracy than the model with no heads. This suggests that some heads might be learning to focus on speech features that are harmful to phoneme classification and using only positional encoding might be better for phoneme classification task in those cases. However, surprisingly, those heads are reported as APC\(_3\), APC\(_4\), and CPC\(_6\), which are diagonal heads, showed to have better phonetic selectivity in correlation analysis. The correlation analysis is conducted with respect to the MoA categories instead of individual phonemes. Thus, these results initially pointed out that the aforementioned heads might be learning to distinguish the manner of articulations from each other but not the individual phonemes within the same MoA category. Yet, they cannot outperform the case when there is no head at all on the MoA classification task either.

**Table 5.4.** Accuracy results by pruning heads on phoneme classification. \(M_{-h_i}\) is where only head \(h_i\) is subtracted from the model. The lowest accuracy is reported in bold. (a), (d), and (p) correspond to the acoustic heads, diagonal heads, and the heads looking into further past, respectively.

<table>
<thead>
<tr>
<th></th>
<th>(a) CPC model</th>
<th></th>
<th>(b) APC model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>head</td>
<td>accuracy</td>
<td>head</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-0}) (d)</td>
<td><strong>0.489</strong></td>
<td>APC(_{-0}) (a)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-1}) (d)</td>
<td>0.538</td>
<td>APC(_{-1}) (d)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-2}) (a)</td>
<td>0.544</td>
<td>APC(_{-2}) (p)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-3}) (a)</td>
<td>0.544</td>
<td>APC(_{-3}) (d)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-4}) (a)</td>
<td>0.545</td>
<td>APC(_{-4}) (d)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-5}) (a,d)</td>
<td>0.536</td>
<td>APC(_{-5}) (p)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-6}) (d)</td>
<td>0.54</td>
<td>APC(_{-6}) (p)</td>
</tr>
<tr>
<td></td>
<td>CPC(_{-7}) (a)</td>
<td>0.544</td>
<td>APC(_{-7}) (p)</td>
</tr>
</tbody>
</table>

Furthermore, Table 5.4 reports the accuracy results of the heads when pruned. In contrast to the results shown in Table 5.3 in both models, the accuracy drops the most when diagonal heads are pruned, as also suggested in [71]. For CPC, accuracy dropped the most when CPC\(_0\) was pruned, followed by CPC\(_1\), CPC\(_5\), and CPC\(_6\), even though the results of those heads do not show a notable difference to be properly interpreted. In APC, the lowest accuracy is reported when APC\(_1\), APC\(_3\), and APC\(_4\) are pruned. These results support the previous findings of correlation analysis on phonemic selectivity of diagonal heads. However, as CPC\(_6\), APC\(_3\), and APC\(_4\) individually give the lowest accuracy, this might indicate that these heads carry crucial information on phoneme discrimination yet
are useful only when combined with others. The heads that accumulate their attention in the long past, CPC\textsubscript{2}, CPC\textsubscript{3}, CPC\textsubscript{4}, CPC\textsubscript{7}, APC\textsubscript{0}, and APC\textsubscript{5}, produce relatively moderate accuracy in both models (Table 5.3) but their contribution can also be compensated by the other heads (Table 5.4). Overall, CPC\textsubscript{0} and APC\textsubscript{1}, the heads that attend to their neighbors with 0-deviation, are reported as the heads that result in a significant drop when pruned as well as the highest accuracy when used individually.

Figure 5.6 displays the accuracy results with forward selection algorithm where the iteration starts with CPC\textsubscript{+0} and APC\textsubscript{+1}. For the APC model, the algorithm first respectively adds APC\textsubscript{7}, APC\textsubscript{6}, and APC\textsubscript{2} based on the accuracy of each subset of heads. Next, by using APC\textsubscript{+1,2,6,7}, it evaluates the performance of the model where the remaining heads are added. As can be seen in the figure, APC\textsubscript{5} brings on a negligible improvement whereas APC\textsubscript{0}, APC\textsubscript{3} and APC\textsubscript{4} result in a decrease in the accuracy. Therefore, APC\textsubscript{1}, APC\textsubscript{2}. APC\textsubscript{6} and APC\textsubscript{7} are reported as the heads that result in the most significant performance improvement. Even though these heads do not show similar behaviors in correlation and regression analysis, they differ from the rest of the APC heads by displaying 0-deviation in temporal analysis (Table 5.1).

For the CPC model, the forward selection method composes CPC\textsubscript{+0,1,5,6} whose performance converges to the original CPC model and none of the remaining heads improve the performance of CPC\textsubscript{+0,1,5,6} significantly. Even though the accuracy results of CPC heads are similar to each other, the heads that contribute to phoneme discrimination the most are CPC\textsubscript{0}, CPC\textsubscript{1}, CPC\textsubscript{5}, and CPC\textsubscript{6}. Besides CPC\textsubscript{0}, CPC\textsubscript{1}, and CPC\textsubscript{6} previously being reported as heads with high phonetic selectivity on correlation analyses, these heads along with CPC\textsubscript{5} differ from the rest by attending to its neighbors (less than 5 frames) whereas the other CPC heads tend to look at longer past (more than 25 frames).

For further inspection, confusion matrices for individual heads are presented in Appendix B and correspond to the phoneme classification results reported in Table 5.3. As can be observed, CPC heads do not demonstrate a remarkable variation between each other. On the other hand, APC heads noticeably differ from each other. In CPC, all heads have the capability of distinguishing MoA categories from each other, except affricates. As affricates are plosives followed by friction, they are mainly predicted as either plosives or fricatives. Additionally, in all the heads, approximants are sometimes classified as vowels. This potentially results from approximants being voiced consonants as described earlier in Section 2.4.2. APC heads furthermore follow the same prediction pattern for affricates and approximants. All APC heads, except APC\textsubscript{4}, classify approximants as vowels and affricates as plosives. APC\textsubscript{4} tends to mislabel each sound as a vowel. Additionally, in contrast to previous findings on correlation analysis, APC\textsubscript{3} and APC\textsubscript{4} cannot distinguish between speech and non-speech. This might be underlying their poor performance in phoneme discrimination task (Table 5.3).
Forward selection of APC.

(b) Forward selection of CPC.

Figure 5.6. Phone classification accuracy with forward selection. (a) APC and (b) CPC accuracy for selected heads. Individual heads are subsequently added to the models. Diagonal heads are denoted with (d), acoustic heads with (a), heads attending to past frames (more than 5 frames) with zero-deviation with (p).

5.3 Discussion of the results

Correlation, linear regression, and temporal behavior analysis methods were evaluated by comparing their results with the performance of the attention heads in the phoneme discrimination task for two different speech models. The performance of the attention
heads was assessed by adapting forward and backward feature selection algorithms.

First, it should be recalled that APC tries to accurately predict 5 frames ahead of input features. On the other hand, CPC first encodes the input features into latent representations and aims to predict 12 frames ahead of latent features instead of the input features. Therefore, CPC can capture more general structures of the speech signal and perform the prediction task over a longer time. This difference between the learning strategies of the two models is furthermore reflected in the temporal analysis. It was observed that the attention heads of the CPC model display more diverse behavior than those of the APC model. In addition, CPC consists of more heads attending to longer past (mean distance of 28 to 43 frames) than of APC. Yet, both models consist of acoustic, diagonal, and 0-deviated heads.

The acoustic heads were defined as the heads that have a high correlation with acoustic features (excluding the phonetic features). The results showed that the acoustic heads are more associated with speech energy rather than pitch or formants. Additionally, these acoustic heads display temporally more spread-out attention by mostly focusing on the longer past. Moreover, diagonal heads were defined as heads with a temporal mean distance of less than approximately 5 frames. The correlation and linear regression analysis showed that the diagonal heads have a stronger linear relation with phonetic information, as expected since neighbor frames tend to consist of the same sounds. However, it was also observed that a diagonal head not necessarily has to be learning from phonetic features and instead could be an acoustic head focusing on local acoustic structures (CPC).

Next, it was noted that subtracting the diagonal heads is the most harmful for phoneme classification tasks. Yet, these heads do not necessarily perform well when they are used as the only heads in the analysis. Moreover, in both APC and CPC, the models with no head outperformed at least one model that only used a diagonal head (see Table 5.3). The heads that are looking into longer past individually produce moderate accuracy but do not cause a noticeable performance drop when subtracted. During forward selection, it was reported that in both models the best combination of the heads can be explained with their temporal behavior. In APC, the heads that are position-wise more stable and looking into a relatively longer past form the best subset whereas, in CPC, the diagonal heads make the best combination. This difference can be related to the learning strategies of the two models. As CPC predicts a further future than APC, CPC tries to exploit the phonetic information in neighbor frames. On the other hand, APC can afford to benefit from the information in longer past as it performs phoneme prediction task in the nearer future.

Finally, both correlation and linear analyses were able to reflect the temporal behavior of the heads in terms of the distance that the heads tend to look at. However, their implications on the performance of the heads for specific phoneme identities were not necessarily accurate. This implies that a linear model might not be enough to capture
the prediction strategy of non-linear models. Furthermore, it should be noted that the cumulative sum reflects the overall behavior of the heads and might not be enough to reveal one-to-one relationships between the frames. Nevertheless, it was shown that analysis of temporal behavior is the most reliable strategy to be used as a predictor of head behavior in phoneme discrimination tasks. The mean distance of the heads provides the best grouping strategy for phoneme classification among the proposed methods and it is additionally supported with the standard deviation of the mean distances.
6. CONCLUSION AND FUTURE WORK

In this thesis, different methodologies to analyze the behaviors of attention heads in Transformer-based speech models were presented and evaluated. The evaluation was conducted by comparing the findings with the performance of the attention heads in the phoneme classification task. Correlation and linear regression analyses were utilized to describe the dependencies between attention heads and speech features. In addition, the temporal behavior of the attention heads was analyzed by the mean distance of attention concentration in each head along with the deviation of those distances across test samples. Furthermore, forward and backward feature selection algorithms were adapted to assess the contribution made by individual heads to the performance of the model on the phoneme classification task. In order to generalize the aforementioned methods, two different predictive coding models were used and the results were discussed by comparing the different learning strategies of these models.

The findings indicate that attention heads of Transformer-based models learn to focus on different types of speech features. The analysis of the temporal behavior shows that the heads learning from phonetic structures rely on the neighbor frames whereas those learning from other acoustic features mostly attend to longer past. Taken together, only a subset of heads was shown to be important for the phoneme classification task. In CPC, these important heads appear to be the diagonal ones that accumulate the total attention in neighbor frames. In APC, they are the heads focusing on temporally longer past with approximately zero deviation. This indicates that the importance of the heads on the phoneme classification task depends on the learning strategy of the model. It was additionally found out that the diagonal heads are the last to be pruned in backward selection in both models, as also suggested in [71]. However, the results showed that those diagonal heads carrying supplementary information are not always individually enough in the prediction task and can only be exploited when combined with others.

For further development, the analysis should benefit from higher-order models instead of linear regression. For instance, a polynomial regression model might be a better fit for capturing the dependencies between attention heads and speech features. Consequently, more explicit and interpretable functionalities might be assigned to the heads and the heads can be further grouped by different properties. Besides, a different head score function than cumulative sum might be required to inspect one-to-one relationships be-
tween the frames that attend to each other to better capture the underlying reason behind the association of a head with a specific speech feature. In order to further improve the evaluation of analysis methods, the findings should be compared with the performance of the heads in different downstream tasks, such as speaker recognition. Additionally, future research could continue to explore the effects of the prediction step of models on the selection of heads contributing to the performance of the model the most.
REFERENCES


APPENDIX A: DISTRIBUTION OF THE SOUNDS IN ATTENTION HEADS

**Figure A.1.** Distribution of the sounds for each head in the cumulative attention for CPC model.
Figure A.2. Distribution of the sounds for each head in the cumulative attention for APC model.
Figure B.1. The normalized confusion matrices for CPC heads.
Figure B.2. The normalized confusion matrices for APC heads.