The Role of Position Embeddings in Transformers for Automatic Speech Recognition

Xinying Wei

Master of Science
School of Informatics
University of Edinburgh
2022

Abstract

Position embeddings in transformers provide position information. However, as the class of a phone is mainly based on its features rather than the position, we hypothesize that position embeddings in transformers are only useful to a certain extent for phone classifiation. We conduct several experiments to explore the role of position embeddings in transformers for phone classification. We find that without position embeddings, transformers partially learn phone classification. However, position embeddings are important for transformers to distinguish stops from silence and separate other phones similar on voicing. They play a crucial role in distinguishing /z/ from /s/. We also find position embeddings improve the smoothness of the output, and the smoothness helps to avoid specific types of errors. Without position embeddings, transformers can still recognize neighbouring frames based on features. Due to the close relation between phone classification and automatic speech recognition (ASR), our conclusions validate the importance of position embeddings in transformers for ASR.

Declaration

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

(*Xinying Wei*)

Acknowledgements

I would like to thank my supervisor Hao Tang for spending a lot of time discussing experiment results with me. Thank him for his insightful comments that inspired me a lot. Thank him for his comprehensive guidance and help that made me successfully complete the project.

I would like to thank Gene-Ping Yang for encouraging me when I struggled, and thank Sung-Lin Yeh for his useful PyTorch tutorial.

I would also like to thank my friends and my family for their support and love along the way.

Table of Contents

Chapter 1

Introduction

Transformer, a deep learning architecture, has shown great performance in a wide range of tasks [6, 5, 8, 31], including automatic speech recognition (ASR) [41, 52, 19]. The transformer receives a sequence as input. A core mechanism in transformer architecture is self-attention. Self-attention captures relations between elements at different positions in the input sequence, allowing transformers to make use of contextual information. As the calculation of self-attention is invariant to the order of the input, transformers are usually provided with position information. Position information in transformers comes from position embeddings.

Previous studies have come up with plenty of ways to construct different position embeddings. Vaswani et al. [44] calculate position embeddings that contain absolute position using sinusoid functions. It is also common to learn position embeddings during training [9, 8, 21, 25]. Shaw et al. [36] proposed relative position embeddings, where the calculation of self-attention in transformers is changed. Subsequent studies [5, 14, 32, 50] make improvements in different aspects based on Shaw et al's work.

Despite having plenty of ways to construct position embeddings, inconsistencies in comprehension persist across different literatures. Position embeddings with various types are usually directly applied to different models without analysis. It remains unclear whether position embeddings are used properly, or how much they benefit models. As many proposed position embeddings are applied in a range of different tasks, it is also questionable if they are suitable for all downstream tasks. Some previous studies reported their findings on the importance of position embeddings. Gehring et al. [9] found position embeddings useless in convolutional neural networks for machine translation. Wang et al. [46] confirmed that position embeddings are necessary for transformers to do machine translation tasks. Haviv et al. [12] suggested that the

transformers trained with and without position embeddings have similar perplexities as language models. Likehomanenko et al. [23] found removing position embeddings in transformers for ASR tasks results in a severe reduction in performance. However, no studies are found to analyze the importance of position embeddings in transformers for phone classification.

Phone is the smallest unit to recognize speech information. Phone classification, a task that is closely related to ASR, classifies phones at each time frame in a given utterance. As the class of a phone mainly depends on features at its corresponding time frame, we hypothesize that compared to tasks where elements have complex dependencies with each other, phone classification could be relatively less sensitive to position embeddings.

This project aims to study the role of position embeddings in transformers in the context of phone classification. We focus on the sinusoid fixed absolute position embedding in [44] since it was the first position embedding used in the original transformer model [44] and also commonly used [1, 49, 11, 23, 29]. The main contributions of this project are as follows.

1. Our experiments confirm that transformers without position embeddings can partially learn phone classification. We further validate that transformers can partially learn phone classification without any order information.

2. We validate that position embeddings help distinguish stops from silence and differentiate phones similar on voicing. They are crucial to separate /z/ from /s/.

3. We validate that transformers without position embeddings identify neighbours of frames based on features.

4. We validate that position embeddings improve smoothness of the output. We also find that this property reduces specific types of errors.

5. We discover the changes that position embeddings make to self-attention as well as the patterns that are not changed.

The main structure of this thesis is as follows. The background knowledge of transformers, position embeddings and essential components of transformers is introduced in chapter 2. Previous work related to transformers without position embeddings is discussed in chapter 3. The transformer model built in this project is described in chapter 4. Experiment results are discussed in chapter 5. Chapter 6 concludes this project, points out the limitation of this project and indicates future work.

Chapter 2

Background

This chapter aims to remind readers of background knowledge of this project. Section 2.1 introduces the structure of transformers. Self-attention mechanism of the transformer is described in section 2.2. Section 2.3 discusses position embeddings in detail. Section 2.4 introduces the background knowledge of residual connections.

2.1 Transformer

The original transformer proposed in [44] comprises an encoder and a decoder. With the original encoder-decoder transformer model, we encode the input to a sequence of hidden states and decode the sequence of hidden states to outputs. As phone classification relies on hidden states, only the transformer encoder is required.

An encoder is formed by a sequence of encoder layers connected one after another (Figure 2.1). Each encoder layer consists of two sublayers. The first sublayer is a multi-head attention layer followed by layer normalization. The second sublayer contains two feedforward layers, followed by layer normalization as well. Residual connection is added in both two sublayers. Input embeddings are combined with position embeddings as the input of the transformer encoder. Input embeddings are vector sequences that represent input elements. They typically solely related to features of inputs themselves and do not carry any position information.

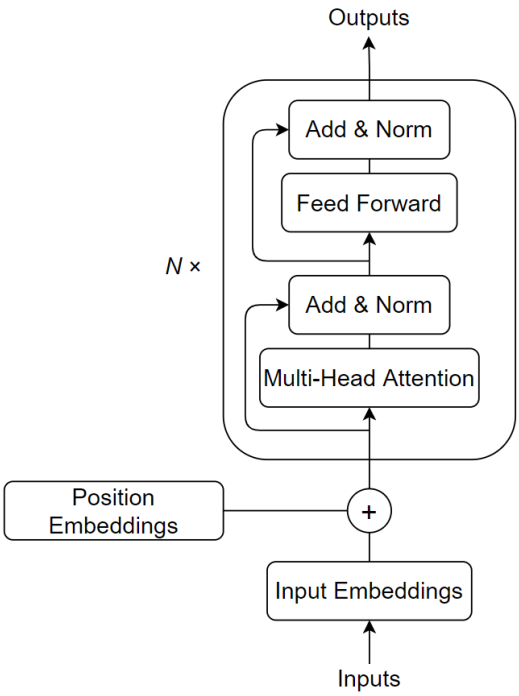


Figure 2.1: Encoder

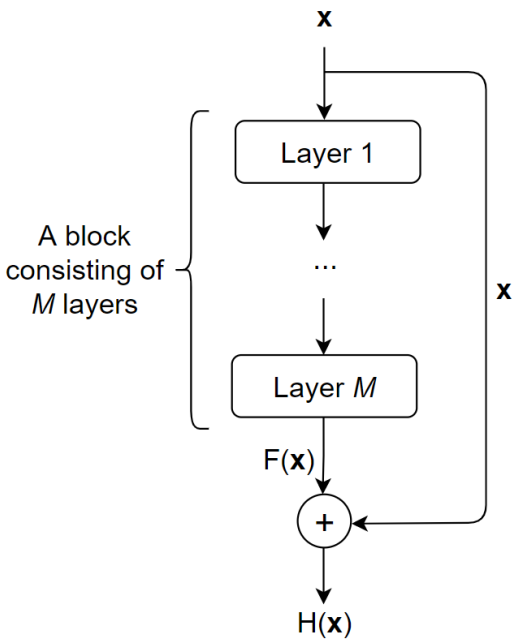


Figure 2.2: Residual Connection

2.2 Multi-Head Self-Attention

Self-attention calculates the attention that the input pays to itself. The attention mechanism receives queries, keys and values as input. It calculates the similarity between queries and keys to obtain the attention scores. The attention scores are applied to values to yield the final result. In self-attention, queries, keys, and values are derived from different linear projections of the same input. In multi-head self-attention, different heads focus on different features along the sequence. With H heads and d_{model} features where d_{model} is a multiple of H, multi-head self-attention evenly assigns d_{model}/H features to each head. The output of multi-head self-attention is obtained by concatenating the results of all heads and passing them through a linear layer.

Suppose the input is a sequence of vectors $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$, we have

$$\text{MultiHead}(\mathbf{X}^Q, \mathbf{X}^K, \mathbf{X}^V) = \text{Concat}(\text{head}_1, ..., \text{head}_H)W^O \tag{2.1}$$

where $\mathbf{X}^Q$, $\mathbf{X}^K$ and $\mathbf{X}^V$ come from the same input $\mathbf{X}$, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ is the projection matrix applied after concatenating heads.

Suppose the output of head_a is $\mathbf{y}_a = \mathbf{y}_{a,1}, ..., \mathbf{y}_{a,T}$. To calculate $\mathbf{y}_{a,i}$ where $i \in 1, ..., T$

we have

$$\alpha_{ij} = \frac{\exp \alpha'_{ij}}{\sum_u^T \exp \alpha'_{iu}} \qquad \alpha'_{ij} = \frac{\mathbf{x}_i^Q \mathbf{W}_a^Q (\mathbf{x}_j^K \mathbf{W}_a^K)^\top}{\sqrt{d_k}} \tag{2.2}$$

$$\mathbf{y}_{a,i} = \sum_{j=1}^{T} \alpha_{ij} (\mathbf{x}_j^V \mathbf{W}_a^V) \tag{2.3}$$

where α'_{ij} multiplies queries by keys and scales them. d_k is the dimension of queries and keys. α_{ij} applies softmax and becomes the attention weight between the query $\mathbf{x}_i^Q$ and the key $\mathbf{x}_j^K$. The results of attention multiplied by values for all $j \in 1, ..., T$ are added together to produce the final output $\mathbf{y}_{a,i}$ of head$_a$ at position i. $\mathbf{W}_a^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_a^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $\mathbf{W}_a^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are projection matrices of queries, keys and values of head$_a$, where d_v is the dimension of values.

As $\mathbf{y}_{a,i}$ is computed as the weighted sum of elements at all positions, it loses all order information during the calculation. This property of self-attention indicates the difficulty for transformers to utilize position information without explicit position embeddings.

2.3 Position Embeddings

Position embeddings are used to provide order information to models. Absolute position embeddings provide the absolute position of each element with regard to the whole input sequence. They are usually combined to input embeddings as part of the input to the model. Relative position embeddings indicate relative position between element pairs. They are usually added to queries, keys or values during the calculation of self-attention. To calculate the fixed absolute sinusoid position embedding [44] at position p, we have

$$\begin{aligned} \text{PE}(p, 2i) &= \sin(p/10000^{2i/d_{\text{model}}}) \\ \text{PE}(p, 2i+1) &= \cos(p/10000^{2i/d_{\text{model}}}) \end{aligned} \tag{2.4}$$

where d_{model} is the model dimension, i is the index that ranges from 0 to half of the model dimension, $2i$ and $2i+1$ refer to dimensions in position embeddings, and p is the position index. Hence, position embeddings in even dimensions dimensions are obtained using sine, whereas in odd dimensions they are obtained based on cosine. The divided term $10000^{2i/d_{\text{model}}}$ ensures that we compute the sine and cosine of position embeddings with different periods in different dimensions. It greatly mitigates the occurrence of repetition problems caused by the sinusoidal period.

There appear to be different ways of combining position embeddings and input embeddings. Some models add position embeddings to input embeddings elementwise

[48, 47, 50], while others concatenate them [38, 10, 16]. In this project, position embeddings are concatenated to input embeddings for the following reasons.

1. Concatenation, rather than addition, avoids mixing input and position features, which are two heterogeneous types of features [20].

2. Concatenation may be more appropriate for acoustic modelling because the input features are fixed speech features rather than learned embeddings [38].

3. Concatenation allows the model to learn position embeddings in a more direct way [16].

2.4 Residual Connection

Residual connection [13] which is added to a block of layers builds a path from the input to the output of the block directly. We define the input of a block of layers to be $\mathbf{x}$, and the desired output of a block of layers to be $\mathrm{H}(\mathbf{x})$. Residual connection creates a map from the desired output $\mathrm{H}(\mathbf{x})$ to $\mathrm{F}(\mathbf{x})$ and input $\mathbf{x}$:

$$\mathrm{H}(\mathbf{x}) := \mathrm{F}(\mathbf{x}) + \mathbf{x} \tag{2.5}$$

where $\mathrm{F}(\mathbf{x})$ is the original output of the block before reaching the point of the residual connection (Figure 2.2). Residual connection is often applied to mitigate the vanishing gradient problem since the model is expected to learn to fit $\mathrm{F}(\mathbf{x}) = 0$ and purely passes the input $\mathbf{x}$ directly to the next layer block. It makes stacking of encoder layers easier. Besides, by adding an additional input to the output of a layer block, the output is allowed to interact directly with the input. More information about the input is able to be kept across a deep network, including the features of the input as well as order information conveyed by position embeddings.

Chapter 3

Related Work

Some studies have suggested the necessity of position embeddings in transformers. Haviv et al. [12] and Likhomanenko et al. [23] show that the fixed absolute sinusoid position embeddings in [44] are of limited help for transformers on some tasks, including language modelling and image classification. Similar analysis on position embeddings also appears on ASR tasks. Zhang et al. [53] remove relative position embeddings from their convolution-augmented transformer model for ASR, finding it does not decrease the model's performance. Park et al. [29] build a convolution-based encoder for ASR. They find that concatenating position embeddings to the output of the convolutions helps the model achieve a lower word error rate. Results of Zhang et al. [53] and Park et al. [29] suggest that the size of the dataset may be a factor influencing the importance of position embeddings. However, Likhomanenko et al. [23] results suggest fixed absolute position embeddings in [44] could be vital in transformers for ASR.

Despite the above studies bringing us some insights, the problem of this project has not been thoroughly solved. Results of image classification [12] and language modelling [23] cannot be directly applied to phone classification. It has been found that object detection task and semantic segmentation task learn more position information than image classification task [18]. Thus, the role of position embeddings in different tasks may vary, and position embeddings could suggest different importance on phone classification and on other tasks. Results of Zhang et al. [53] and Park et al. [29] cannot be directly applied to all transformers since their models are convolution-based. As it has been shown that convolution layers can learn position information themselves [9, 18], their results provide limited help on the importance of position embeddings for the original transformer in [44]. Results of Likhomanenko et al. [23] indicate that position embeddings are needed for ASR. In spite of the close relation between phone

classification and ASR, the scope of this project is limited to phone classification. Above studies [12, 23, 53, 29] compare the performance of transformers with and without position embeddings without exploring the reasons behind it.

Nonetheless, Wang et al. [45] discuss the importance of position embeddings in a more convincing way. They discovered that removing position embeddings has a greater impact on some performance metrics than others, implying that position embeddings in BERT [6] may be partially necessary for language modelling. Wang et al. [45] propose three properties of position embeddings, namely monotonicity, translation invariance, and symmetry respectively. They introduce a term, proximity, to represent the closeness of position embeddings. Assume $\mathbf{e}_a$ represents the position embeddings at position $a \in \mathbb{N}$. An inner product $\phi(\mathbf{e}_a, \mathbf{e}_b)$ suggests the proximity between position embeddings $\mathbf{e}_a$ and $\mathbf{e}_b$. We have

$$|a-b| > |a-c| \Rightarrow \phi(\mathbf{e}_a, \mathbf{e}_b) < \phi(\mathbf{e}_a, \mathbf{e}_c) \tag{3.1}$$

Monotonicity: the proximity of position embeddings goes down as the distance between positions increases. Monotonicity may help the model distinguish between close and distant positions.

Translation invariance: the proximity between all position pairs with the same distance remains constant. It may enable the model to learn that each proximity corresponds to a fixed distance between two positions.

Symmetry: the proximity of two position embeddings $\mathbf{e}_a$ and $\mathbf{e}_b$ is the same as the proximity of two position embeddings $\mathbf{e}_b$ and $\mathbf{e}_a$. Symmetry eliminates the direction position information. It makes distinguishing between preceding and succeeding elements difficult for the model.

They discover that monotonicity and translation invariance benefit downstream tasks, while symmetry brings negative influence. They claim that position embeddings may partially support the three properties. It is shown that sinusoid position embeddings proposed in [44] satisfy translation invariance and symmetry but not monotonicity. It indicates that when we explore the role of the sinusoid position embeddings in [44], we also look at the role translation invariance.

According to their visualization of averaged attention weights, all position embeddings suggest some patterns of attention distribution correlated to monotonicity and translation invariance. BERT without position embeddings evenly attends to all tokens, instead of nearby tokens when position embeddings are included. It suggests that removing position embeddings harms self-attention and may related to lacking

monotonicity and tranlation invariance. Nevertheless, conclusions on language models may differ when the task is changed to phone classification. The results on the original transformer in [44] may suggest a difference on BERT as well.

Shim et al. in [37] verify the importance of position embeddings when decomposing the role of self-attention. They state that the main roles of self-attention in transformers for ASR task include phonetic and linguistic localization, where phonetic localization makes self-attention pay attention to traditional phonetic features across the whole utterance, and linguistic localization relates to frames nearby. They modify the self-attention matrix and increase the independence of the two roles during training. Their modified self-attention performs better than the transformer with the original self-attention and relative position embeddings. It implies that their modified self-attention lessens the need of relative position embeddings. Their findings may not suit the original transformer encoder with absolute position embeddings.

Chapter 4

Methodology

This chapter describes the methodology applied in this project. Section 4.1 defines the problem settings as well as the input and output of the model. The architecture of the transformer model is presented in section 4.2. Section 4.3 describes how the models are trained and tested. The dataset and feature extraction is introduced in section 4.4. Visualization approaches used in this project are introduced in Section 4.5.

4.1 Problem Settings

The input of the transformer is a sequence of Mel spectrogram features with 40 dimensions. Given an utterance containing T frames, the input $\mathbf{x} = \mathbf{x}_1, ..., \mathbf{x}_T$ is a matrix of size $T \times 40$. For each time frame i, we get a corresponding class c_i according to the output $\mathbf{y}_i$. Hence the output of the model $\mathbf{y} = \mathbf{y}_1, ..., \mathbf{y}_T$ and the classes $\mathbf{c} = c_1, ..., c_T$ have the same length to the input $\mathbf{x}$. Phones are classified in the scope of 42 classes. Thus $\mathbf{y}$ is of size $T \times 42$, and $\mathbf{c}$ is a vector of scalar indices of length T. Given a frame index $i \in [1, T]$, to obtain class c_i from the output $\mathbf{y}_i = y_{i,1}, ..., y_{i,42}$, we have

$$c_i = \underset{c_i \in [1,42]}{\text{argmax}}\ y_{i,c_i} \tag{4.1}$$

The classes of phones are shown in Table 4.1, where sil is silence, spn is spoken noise, and nsn is non-spoken noise.

4.2 Model Architecture

Our transformer model follows the structure proposed in [44]. To construct the input of the model, we first concatenate position embeddings (Equation 2.4) with dimension

aa	ae	ah	ao	aw	ay	b	ch	d	dh	eh	er	ey	f
g	hh	ih	iy	jh	k	l	m	n	ng	nsn	ow	oy	p
r	s	sh	sil	spn	t	th	uh	uw	v	w	y	z	zh

Table 4.1: 42 Phone Classes

Argument	Meaning	Value
d_{model}	Model dimension	512, 768
d_{ff}	Dimension of the feedforward sublayer	2048
N	Number of encoder layers	6
H	Number of heads	8
dropout	Dropout argument	0.1

Table 4.2: Model Arguments

d_{model} to the 40 features of the input. After a linear projection of $(d_{\text{model}} + 40) \times (d_{\text{model}} + 40)$, the concatenated embeddings are fed into the transformer encoder. When we build the transformer without position embeddings, position embeddings are padded with zero. In this way, position embeddings carry no information and the model's structure remains unchanged.

Arguments of the model are listed in Table 4.2. Aside from the original d_{model} in [44], 512, we also tried 768 since it is a common choice that achieves excellent performance in plenty of transformers [48, 8, 47, 20]. The choices of 6 layers, 8 heads, $d_{\text{ff}} = 2048$, and dropout $= 0.1$ follow [44]. It is also a common option to have 12 layers and 12 heads [8, 20, 32]. Our conclusions on transformers with 6 layers and 8 heads might apply to those with 12 heads and 12 layers as well.

4.3 Training and Testing

Models are trained using stochastic gradient descent (SGD) optimizer. SGD randomly picks an utterance from the input each time and applies back propagation to it. Given a loss function, back propagation calculates the gradient of the loss with respect to the weights of the model. It adjusts the weights based on the gradient in the direction of minimizing the loss. In this project, the training objective is to minimize the cross entropy loss since it is commonly used by classification tasks [24, 27, 26]. Cross

entropy loss calculates the cross entropy between the estimated distribution and the true distribution.

$$\mathbf{L} = -\sum_{i=1}^{42} t_i \log p_i \tag{4.2}$$

where 42 is the total number of phone classes. $\mathbf{p} = p_1, ..., p_{42}$ is the estimated distribution, where p_i is the probability predicted by the model that the phone belongs to class i. $\mathbf{t} = t_1, ..., t_{42}$ is the true distribution, indicating whether the target phone class is class i ($i \in 1, ..., 42$). We have $t_i = 1$ if the target phone class is class i, and $t_i = 0$ otherwise. The cross entropy loss for each phone is continuously ranging from 0 to 1.

During testing, we compute phone error rate (PER) to measure the performance of the transformer. PER calculates the minimum number of steps to transform the predicted phone sequence to the target phone sequence. PER is defined in Equation 4.3.

$$\text{PER} = \frac{I + D + S}{N} \tag{4.3}$$

where I, D, and S correspond to the number of insertions, the number of deletions, and the number of substitutions respectively. N is the total number of phones in the target sequence. Since the input and output of an utterance in phone classification are always the same length, PER for this specific task could be written as Equation 4.4.

$$\text{PER}_{\text{class}} = \frac{S}{N} \tag{4.4}$$

Phone error rate is commonly used in phone classification [43, 22, 33]. The same way to calculate word error rate is also commonly applied in ASR tasks [38, 41, 19]. It shows us a general ability of the model to predict correct class. PER is measured at test time rather than training time since it considers only the binary correctness of the prediction at each time frame, without worrying about how far the prediction at a frame is away from the target.

4.4 Data

Our experiments are conducted using Wall Street Journal (WSJ) dataset since it is commonly used and suggests excellent performance on plenty of ASR related tasks [7, 28, 40, 3]. The dataset with 37395 utterances is split to 9:1 as the training set and the dev set respectively. We train the model on the training set. When we test the model on the dev set, we test their performance on generalization or pick up the best epoch.

We may also test the model on the training set to check the model's performance on fitting the data during training.

Each utterance in the dataset is firstly sampled to obtain the waveform. The waveform is converted to the Short-Time Fourier Transform. It is then transformed to the log Mel spectrogram with 40 dimensions at each time frame. Hence, each time frame of the log Mel spectrogram contains 40 features. Each log Mel spectrogram feature is normalized to mean 0 and standard deviation 1. The time frame is sometimes called the frame in the following content for simplicity.

4.5 Visualization Approach

In addition to phone error rates, we also evaluate transformers according to confusion matrices and attention maps. We visualize the similarity of position embeddings at different positions to interpret our findings as well.

4.5.1 Confusion Matrix

Confusion matrix is a common evaluation approach for classification problems [35, 42, 39]. Confusion matrix visualizes the model's prediction at the class level. It suggests how often a class is predicted to another class. Each row represents predictions of a target label, while each column corresponds to a predicted label. An entry at the row of target label C_{target} and the column of predicted label C_{predict} suggests how often that C_{target} is classified to C_{predict}. To derive a confusion matrix, we firstly record the number of times that each C_{target} is classified to C_{predict}. Each value is divided by the summation of its rows to obtain the distribution with respect to the target class. Confusion matrices tell us which phone classes are easy for the model to recognise and which phone classes are difficult for the model to distinguish. By comparing confusion matrices of transformers with and without position embeddings, we can understand the influence of position embeddings on each phone class.

4.5.2 Attention Map

Attention maps are commonly used for transformers [45, 4, 37]. Attention map visualizes the attention weights of self-attention. It corresponds to the output of Equation 2.3. Given an utterance of length T, the attention weight matrix of a head is of size $T \times T$. Attention maps reflect the distribution of attention paid by each phone in the utterance.

They provide insights into how transformers identify relations between phones and how position embeddings alter it. Each head has an attention weight matrix. Besides showing the attention of each head separately [37], some work averages the attention matrices of all heads to look up the general distribution of attention [45, 4]. In this project, we do both of them.

4.5.3 Position Similarity

To better understand the influence that position embeddings bring to self-attention, we also visualize the similarity of the fixed absolute sinusoid position embeddings proposed by Vaswani et al. [44]. Previous work [48] visualizes the similarity of the position embeddings proposed by Vaswani et al. [44] with a maximum sequence length of 500 by cosine similarity. Cosine similarity of two vectors is obtained by dividing the dot product of two vectors by the product of Euclidean norms of their magnitude. Given two position embeddings $\mathbf{e}_a$ and $\mathbf{e}_b$, the cosine similarity between $\mathbf{e}_b$ and $\mathbf{e}_b$ is calculated by Equation 4.5.

$$\text{similarity}_{\cos}(\mathbf{e}_a, \mathbf{e}_b) = \frac{\mathbf{e}_a \cdot \mathbf{e}_b}{\|\mathbf{e}_a\| \|\mathbf{e}_b\|} \tag{4.5}$$

We claim that the similarity based on dot product could be more reasonable than cosine similarity since this is how attention is calculated (Equation 2.3). Hence, we calculate similarity of position embeddings $\mathbf{e}_a$ and $\mathbf{e}_b$ according to Equation 4.6.

$$\text{similarity}_{\text{dot}}(\mathbf{e}_a, \mathbf{e}_b) = \mathbf{e}_a \cdot \mathbf{e}_b \tag{4.6}$$

Besides looking at the similarity of position embeddings in a whole in [48], we also look at the similarity in detail. We visualize the similarity of position embeddings of length 3000 since the maximum length of our utterances is 2344. By visualizing the dot product similarity of position embeddings at different positions, we can know what similarity information of positions is supplied to self-attention.

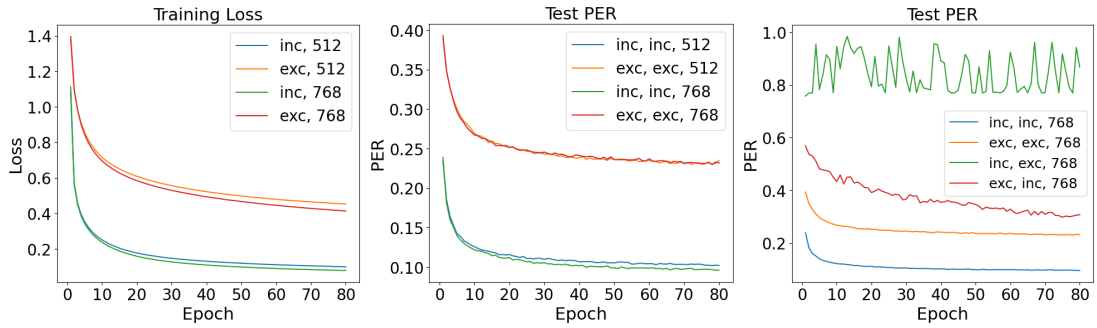
Chapter 5

Experiments

This chapter discusses experiments conducted in this project. Experiment settings are introduced in 5.1. Section 5.2 shows the training losses and test phone error rates of the models. To validate the hypothesis we proposed in section 5.2, we generate confusion matrices and present them in section 5.3. Section 5.4 shows attention maps of the models and discusses the patterns learned by models in detail. Two additional hypotheses are generated based on attention maps. They are validated in section 5.5 and 5.6. We further validate if the model learns to ignore the misplaced position embeddings in section 5.7. The similarity of the fixed absolute sinusoid position embeddings is visualized and discussed in section 5.8.

5.1 Experiment Settings

To set the learning rate, we have reviewed the choices from previous studies. Some transformers are trained with a learning rate of 0.05 with SGD optimizer, and reduces the learning rate to 0.005 from epoch 120 to epoch 160 [54]. Some transformers are fine-tuned with the largest learning rate 0.06 and finally drops to 0.003 [8]. It is found that learning rates ranging from 0.05 to 0.25 suggest no difference with SGD, while the learning rate of 0.01 leads to slower convergence [30]. In this project, the starting learning rate is set to 0.05. To prevent fluctuation of the training loss and bring it closer to the local minimum [51], it is changed to 0.001 after 70 epochs. Instead of training by batch, we train utterances one by one to avoid padding utterances having different lengths. Training is stopped at epoch 80 since it shows no evident improvement of PER on the dev set in the last 10 epochs.



(a) Training loss. (b) Test PER. (c) Test PER.

Figure 5.1: Training loss and test PER. Numbers 512 or 768 indicate model dimension. The first and the second inc or exc indicates position embeddings included or excluded during training and testing respectively.

PE Included or excluded	d_{model}	Final Training Loss	Best PER and Epoch
Included	512	0.0986	10.19%, epoch 78
Excluded	512	0.4522	23.00%, epoch 75
Included	768	0.0782	9.60%, epoch 79
Excluded	768	0.4134	22.98%, epoch 78

Table 5.1: Final training losses and the best PERs of different models. PE refers to position embeddings. d_{model} is the model dimension. Final training loss is the training loss at the 80th epoch. Best PER is measured when models are tested with the same configurations with training. Epoch is the corresponding epoch of the best PER.

5.2 Losses and PERs

The training loss and the PER along epochs are illustrated in Figure 5.1. The final training losses and the best PERs are listed in Table 5.1. We can see that transformers trained without position embeddings suggest some learning ability, while they are still worse than transformers trained with position embeddings. It is also noticed that the training loss of transformers without position embeddings fails to converge to zero, meaning that there exists something that could not be learned by the transformer without position embeddings during training. This leads to Conclusion 5.1 and Hypothesis 5.1.

Conclusion 5.1. Transformers without position embeddings partially learn phone classification.

Hypothesis 5.1. Transformers trained without position embeddings fail to learn to

Training Configuration	Name
PE Included, $d_{\text{model}} = 768$	M-inc
PE Excluded, $d_{\text{model}} = 768$	M-exc

Table 5.2: Naming of models, where PE refers to position embeddings.

Testing Configuration	Name
PE Included, on M-inc at epoch 79	inc-inc
PE Excluded, on M-exc at epoch 78	exc-exc
PE Excluded, on M-inc at epoch 79	inc-exc
PE Included, on M-exc at epoch 78	exc-inc

Table 5.3: Naming of testing configurations, where PE refers to position embeddings.

predict specific phones during training.

To validate this hypothesis, we test the model using 1000 utterances of the training set. We generate confusion matrices in section 5.3 on both the dev set and the 1000 training utterances. In addition, we get an extremely poor result when we test the transformer trained with position embeddings without using position embeddings. It means that the transformer trained with position embeddings learns position information during training and needs position embeddings when classifying phones. We generate confusion matrices for it as well to explore the model's performance with respect to phone classes.

Since it is observed that dimension 768 works better than dimension 512, all remaining experiments focus on transformers with dimension 768. For simplicity, we name the trained transformer models as well as the testing configurations. They are shown in Tables 5.2 and 5.3.

5.3 Confusion Matrices

When generating confusion matrices, we removed values of entries on the diagonal line. Entries on the diagonal line from left up to bottom right correspond to correctly classified cases and tend to have large values. Large values make confusion matrices makes entries of small values have similar shallow colors and hard to distinguishable. We additionally set the entry which corresponds to the target phone class /sil/ and the predicted phone class /nsn/ to zero, since it is comparable in magnitude to the diagonal

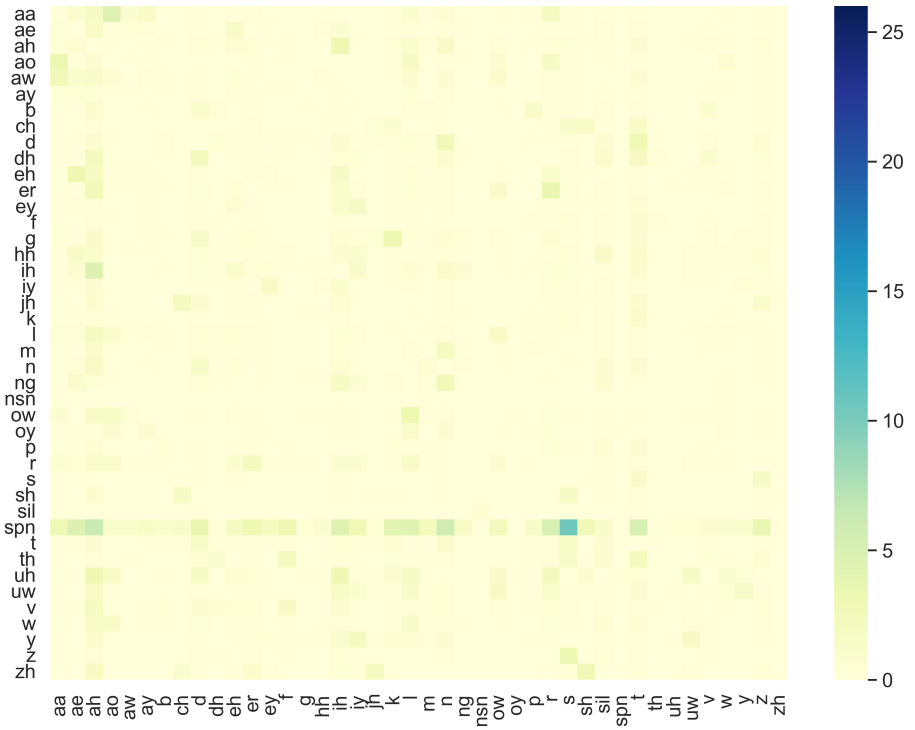


Figure 5.2: Confusion matrix of inc-inc on the dev set, where the maximum value of the color bar is normalized to 26 for easier comparison.

values and suggests similar in different configurations. Confusion matrices of inc-inc and exc-exc on the dev set are shown in Figures 5.2 and 5.3. Confusion matrices of inc-inc, exc-exc and inc-exc on the 1000 training utterances are shown in Figures 5.5, 5.6 and 5.4 respectively. Since the maximum value among Figures 5.2, 5.3, 5.5 and 5.6 is around 26 in Figure 5.3 , we normalize the maximum value of the color bars in the above figures to 26 for easier comparison.

By comparing Figures 5.2 and 5.3, we arrive at three findings below.

1. Problems that are not obvious on inc-inc become serious on exc-exc. For example, exc-exc struggles to distinguish some phone pairs that are similar on voicing, such as to distinguish /z/ from /s/, and /b/ from /p/. It also has trouble differentiating stops like /b/ /k/ /p/ /t/ from /sil/. We presume it is due to stops having closures that could be mistaken for silence. These errors rarely occur on inc-inc.

2. Some phones that are challenging for inc-inc to predict are even more difficult for exc-exc to correctly classify. For instance, inc-inc makes a few minor mistakes on occasionally classifying /ih/ and /uh/ as /ah/, /aa/ as /ao/, /ng/ as /n/, and /spn/ as /s/. These mistakes happen more often on exc-exc.

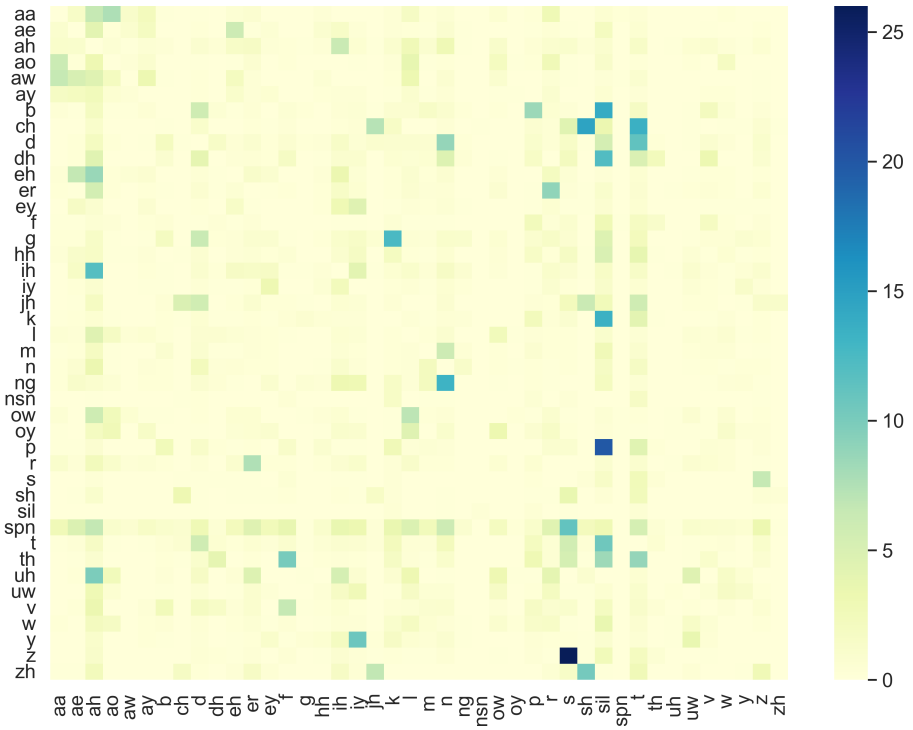


Figure 5.3: Confusion matrix of exc-exc on the dev set.

3. Plenty of phone classes are not easily confused by both inc-inc and exc-exc. They rarely mix up vowels and consonants. They also work well on separating some phones that are both vowels or consonants but show relatively less similarity on voicing, such as /ey/ and /ah/, /f/ and /k/.

According to Figure 5.4, M-inc utilizes position embeddings when predicting all phone classes. Figure 5.5 suggests that M-inc learns to fit the data well. According to Figure 5.6, the errors of exc-exc on the dev set are also evident on the training data, indicating that M-exc does not learn well to differentiate stops from silence and to separate phones similar on voicing during training. It validates our Hypothesis 5.2 and leads to Conclusions 5.2 and 5.3.

Conclusion 5.2. Without position embeddings, the transformer can learn to differentiate phones that are not similar on voicing.

Conclusion 5.3. Position embeddings are useful for the transformer (1) to distinguish phones which are similar on voicing, and (2) to distinguish stops from silence. Position embeddings play a key role in separating /z/ from /s/.

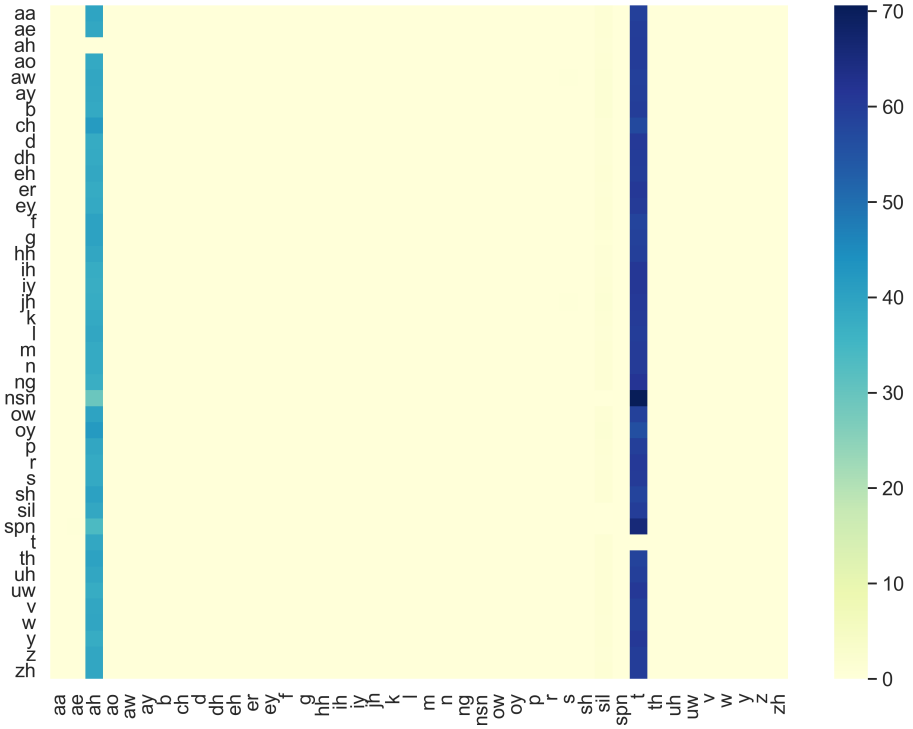


Figure 5.4: Confusion matrix of inc-exc on 1000 training utterances.

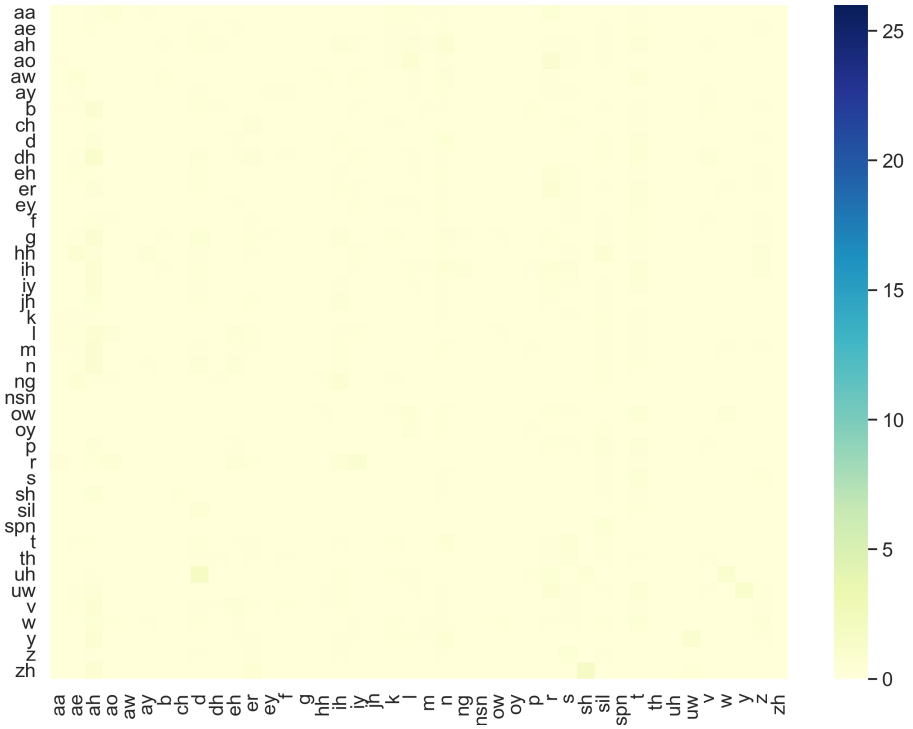


Figure 5.5: Confusion matrix of inc-inc on 1000 training utterances, where the maximum value of the color bar is normalized to 26 for easier comparison.

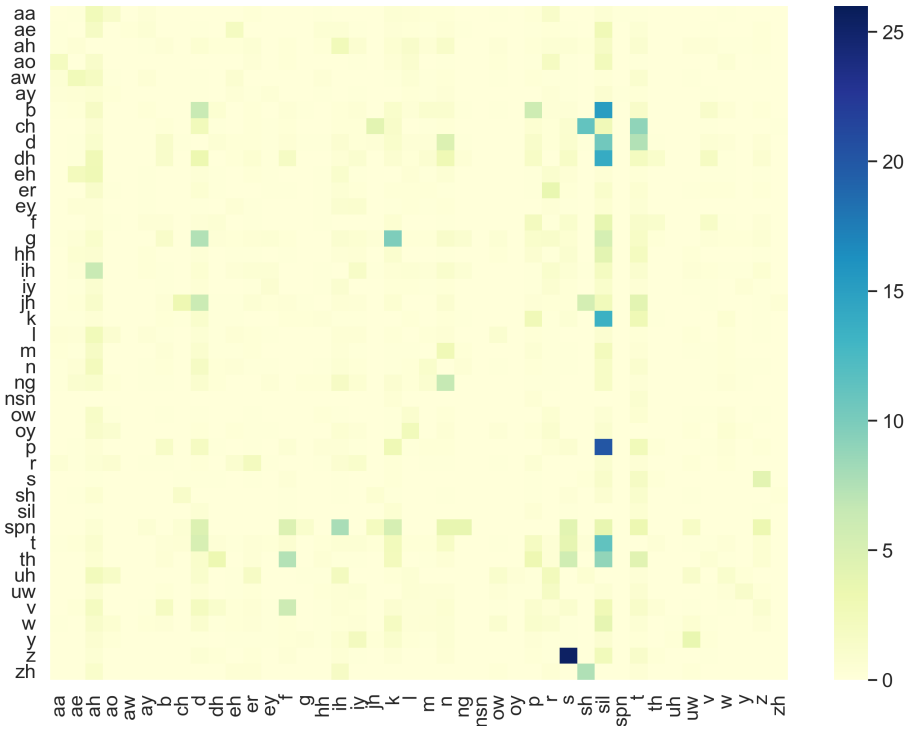


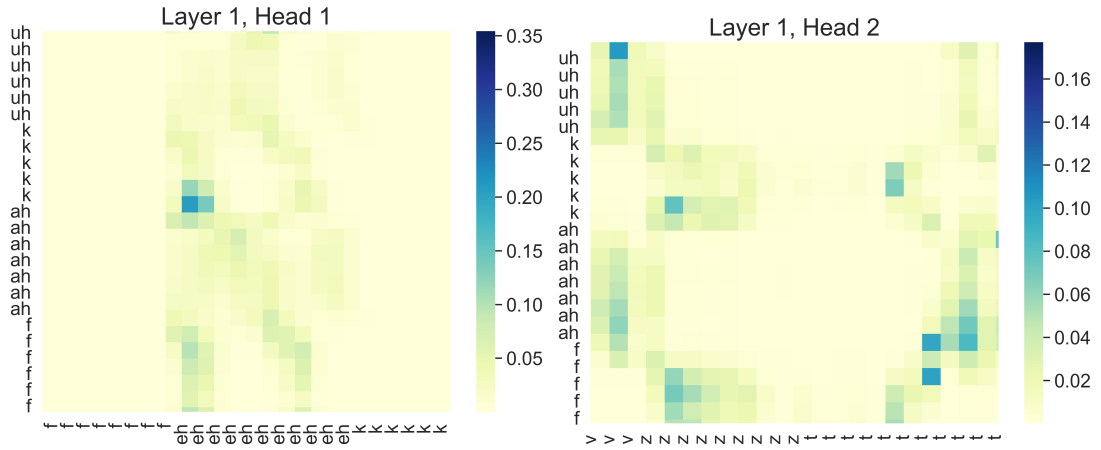
Figure 5.6: Confusion matrix of exc-exc on 1000 training utterances, where the maximum value of the color bar is normalized to 26 for easier comparison.

5.4 Attention Maps

To explore how position embeddings influence self-attention, we randomly pick up an utterance from the dev set and visualize attention maps of inc-inc and exc-exc. For simplicity, we name layer i as Li where $i \in [1,6]$. We name the j-th head in layer i as LiHj where $i \in [1,6]$ and $j \in [1,8]$.

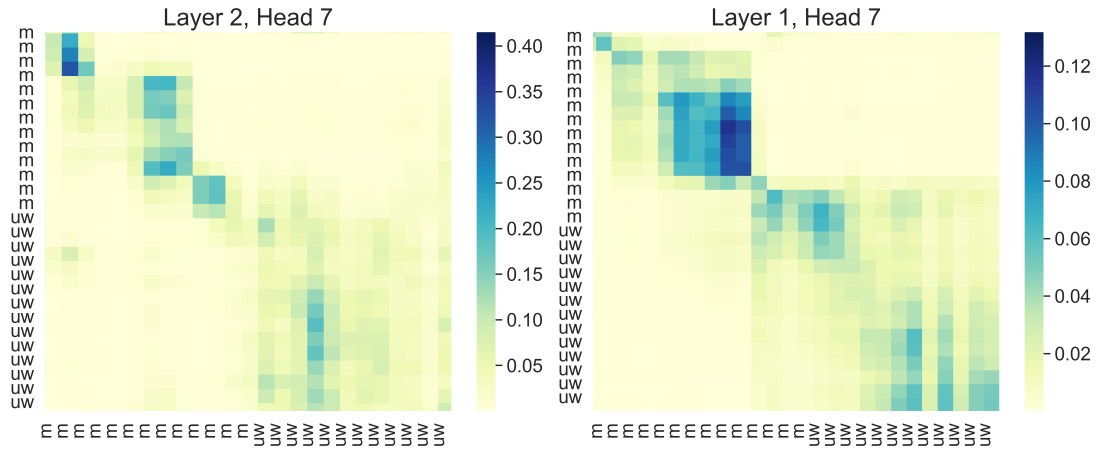
It is discovered that M-inc and M-exc learn some common patterns in self-attention. Both models contain heads that seem to separate consonants and vowels. In Figure 5.7, the attention given by consonants /k/ and /f/ differs from the attention given by vowels /uh/ and /ah/. Both models contain heads where frames of the same phone classes pay attention to each other (Figure 5.8). These heads appear to identify which frames belong to the same phone class. Both models contain heads that a group of frames of the same phone class pay attention to frames at the boundary of the phone class (Figure 5.9). Since it is often the case that frames at the boundary are misclassified to the class on the other side, we assume the model compares features of frames with boundaries around them to better identify the the boundaries.

Despite two models learning some common patterns, position embeddings make some differences. They are discussed in sections 5.4.1, 5.4.2 and 5.4.3 respectively.



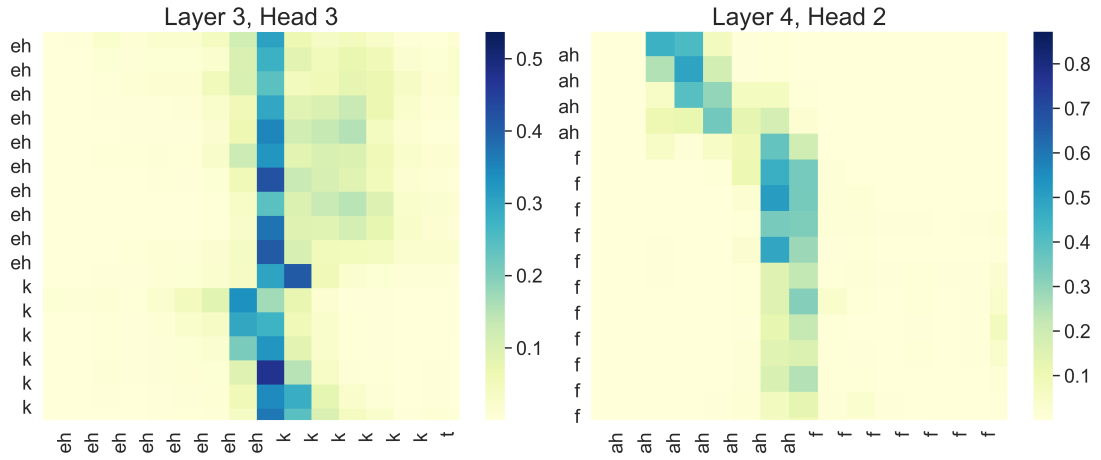
(a) Partial attention map of L1H1 of inc-inc. (b) Partial attention map of L1H2 of exc-exc.

Figure 5.7: Heads in inc-inc and exc-exc that separate consonants and vowels.



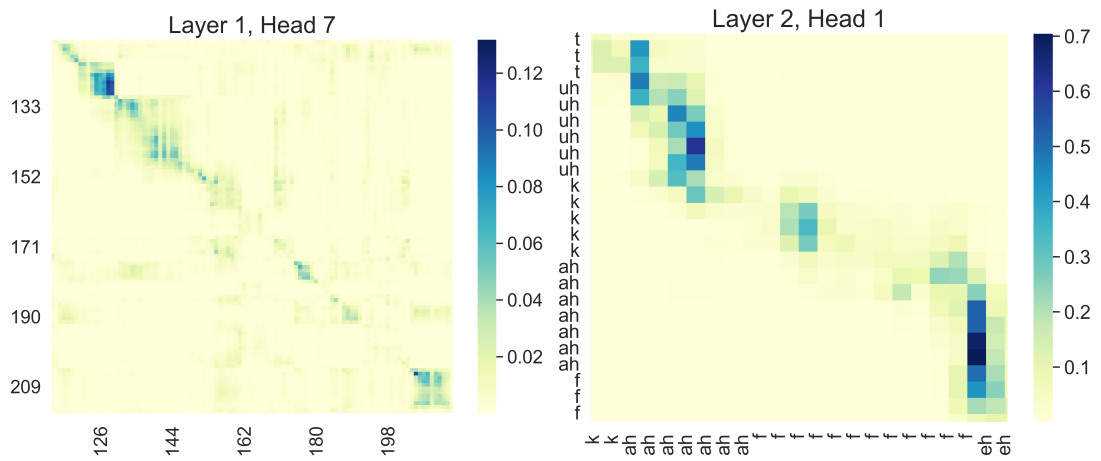
(a) Partial attention map of L2H7 of inc-inc. (b) Partial attention map of L1H7 of exc-exc.

Figure 5.8: Heads in inc-inc and exc-exc where frames of the same class pay attention to each other.



(a) Partial attention map of L3H3 of inc-inc. (b) Partial attention map of L4H2 of exc-exc.

Figure 5.9: Heads in inc-inc and exc-exc where frames pay attention to boundaries.



(a) Partial attention map of L1H7 of exc-exc. (b) Partial attention map of L2H1 of inc-inc.

Figure 5.10: A head in exc-exc where frames of the same class pay attention to themselves, and a head in inc-inc where frames of the same class pay attention to frames of the next class.

5.4.1 Difference of Attention on Phonetic Features

Compared to inc-inc, frames of the same class in exc-exc pay more attention to each other. The squares in Figure 5.10a suggest frames of the same class pay attention to each other. They are more often found in attention maps of exc-exc. In M-inc, attention distributed in patches usually looks like rectangles, where frames tend to pay attention to later frames or previous frames of different phone classes (Figure 5.10b). Even if the square attention pattern can be found in attention maps of inc-inc, it tends to be more evident in attention maps of exc-exc (Figure 5.8). We assume without explicit position information, more attention is paid to features to extract relations between frames. It leads to Conclusion 5.4.

Conclusion 5.4. The transformer without position embeddings tends to pay more attention to phonetic features.

5.4.2 Difference of Attention on /z/ and /s/

Attention maps in L6 of exc-exc appear to have two column lines, which correspond to a frame of /z/ and a frame of /s/ respectively (Figure 5.11). In the eight heads of L6, nearly all frames pay the greatest attention to /z/, little or similar attention to /s/ and nearly no attention to other phones. No heads in inc-inc show this pattern. We suppose L6 of M-exc mainly identifies /z/ and /s/. We find /z/ pays little attention to both /z/

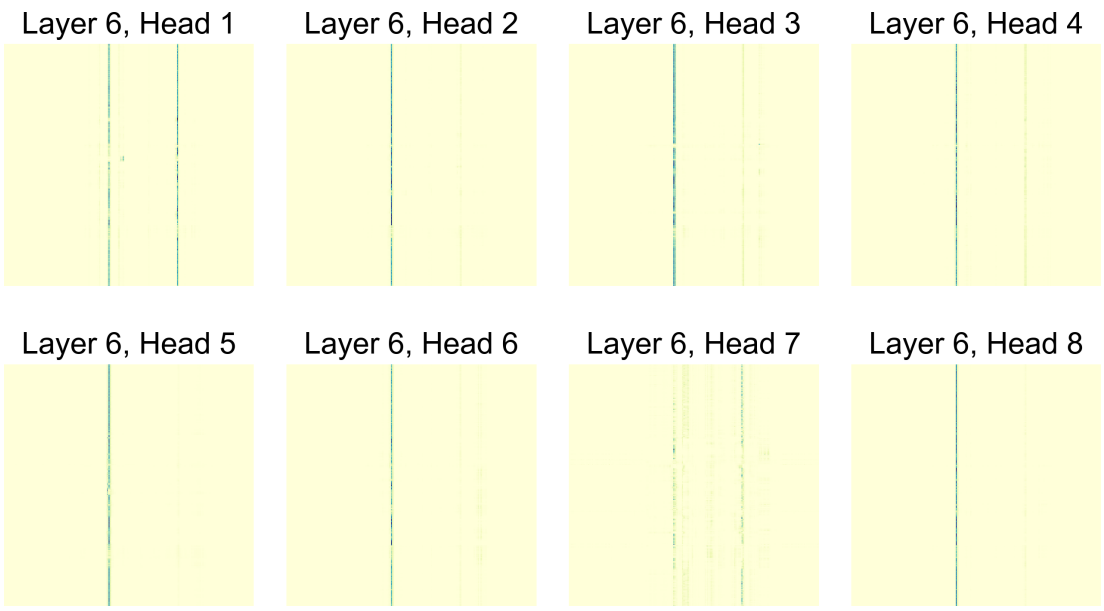
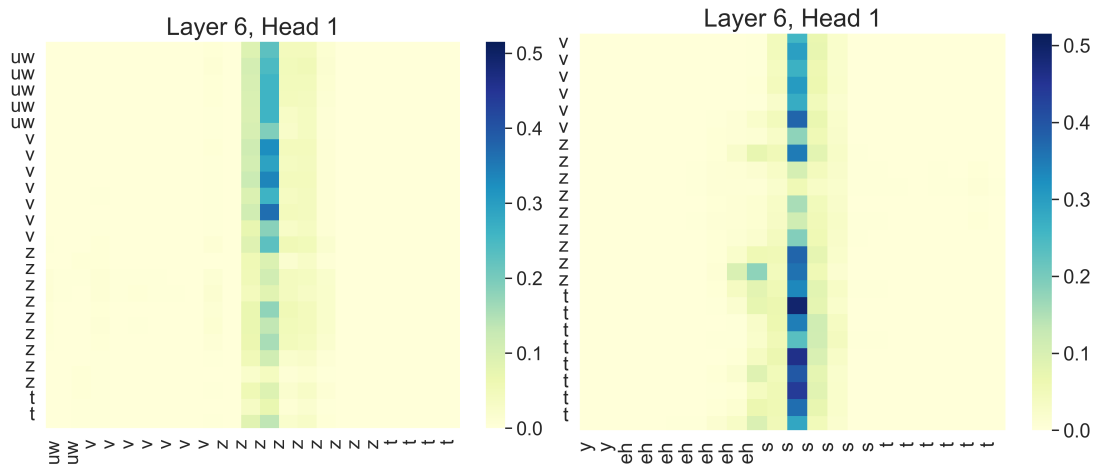


Figure 5.11: Attention maps of L6 of exc-exc.

and /s/ (Figure 5.12, 5.14a). It may relate to the fact that /z/ is often classified to /s/. In addition, /s/ pays different attention to /z/ and /s/ (Figure 5.13, 5.14b). It may relate to the fact that /s/ is rarely classified to /z/. Recalling that M-exc is particularly prone to misclassifying /z/ as /s/, it appears to make a great effort in L6 to differentiate between the two based solely on features.

In order to explore how M-exc learns to classify /z/ and /s/, we visualize the attention maps of L6 at epoch 30 (Figure 5.15). We name the configuration that testing without position embeddings on M-exc at epoch 30 as exc-exc-epoch30. It is found that in exc-exc-epoch30, /z/ is paid with much greater attention by almost all frames, while nearly no attention is given to /s/ and other phones. Also, all frames pay almost the same attention to /z/, except for /z/ itself (Figure 5.16). We assume they are attempting to set /z/ apart from all other phones. Therefore, M-exc first attempts to identify /z/ by comparing all frames with a particular /z/ frame. The range of comparison expands to a few frames during training. It leads to Conclusion 5.5.

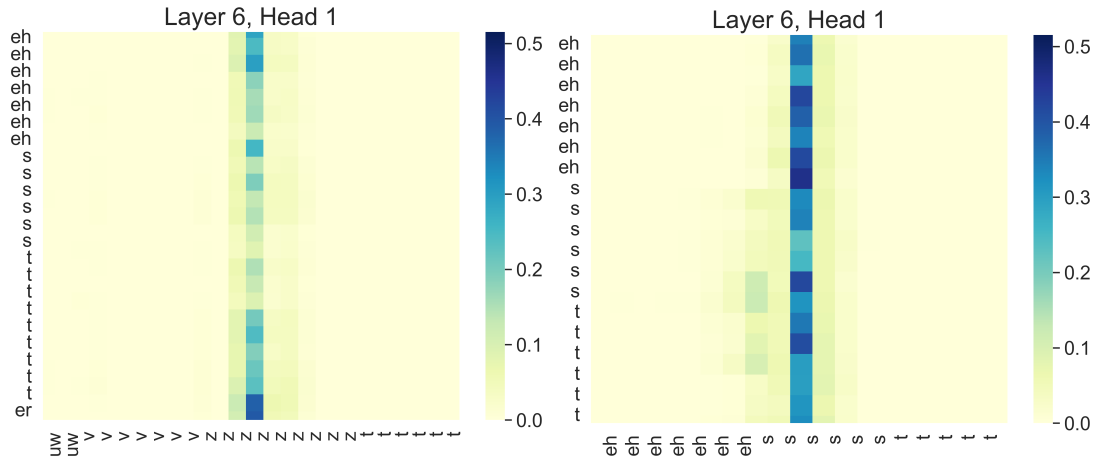
Conclusion 5.5. Without position embeddings, the transformer tends to filter /z/ and /s/ at the last self-attention layer by comparing all frames with correspondingly evident frames of /z/ and /s/.



(a) /z/ pays little attention to itself.

(b) /z/ pays little attention to /s/.

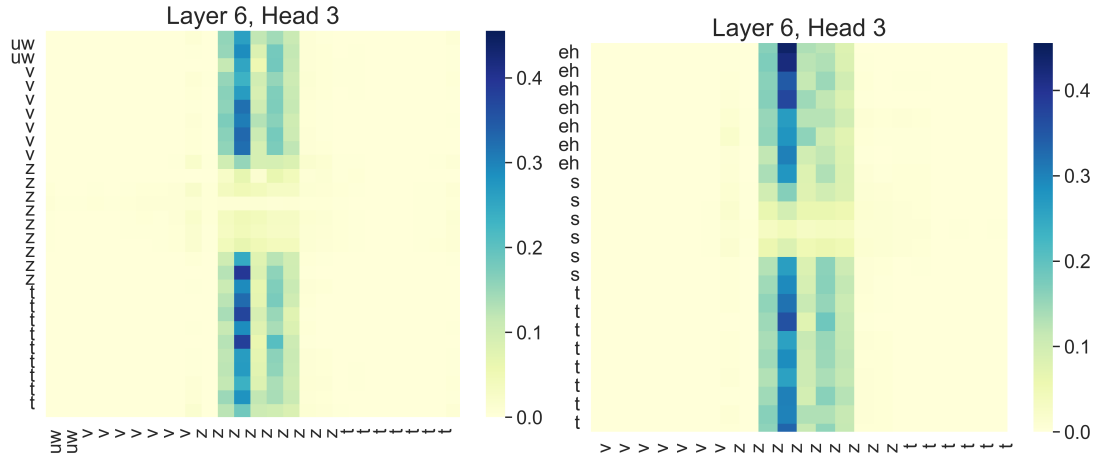
Figure 5.12: Partial attention maps of L6H1 of exc-exc, where /z/ pays little attention to itself and /s/.



(a) /s/ pays little attention to /z/.

(b) /s/ pays much attention to itself.

Figure 5.13: Partial attention maps of L6H1 of exc-exc, where /s/ pays much attention to itself but little to /z/.



(a) Partial attention map of L6H3 of exc-exc, where /z/ pays little attention to itself.

(b) Partial attention map of L6H3 of exc-exc, where /s/ pays little attention to /z/.

Figure 5.14: Partial attention maps of L6H3 of exc-exc.

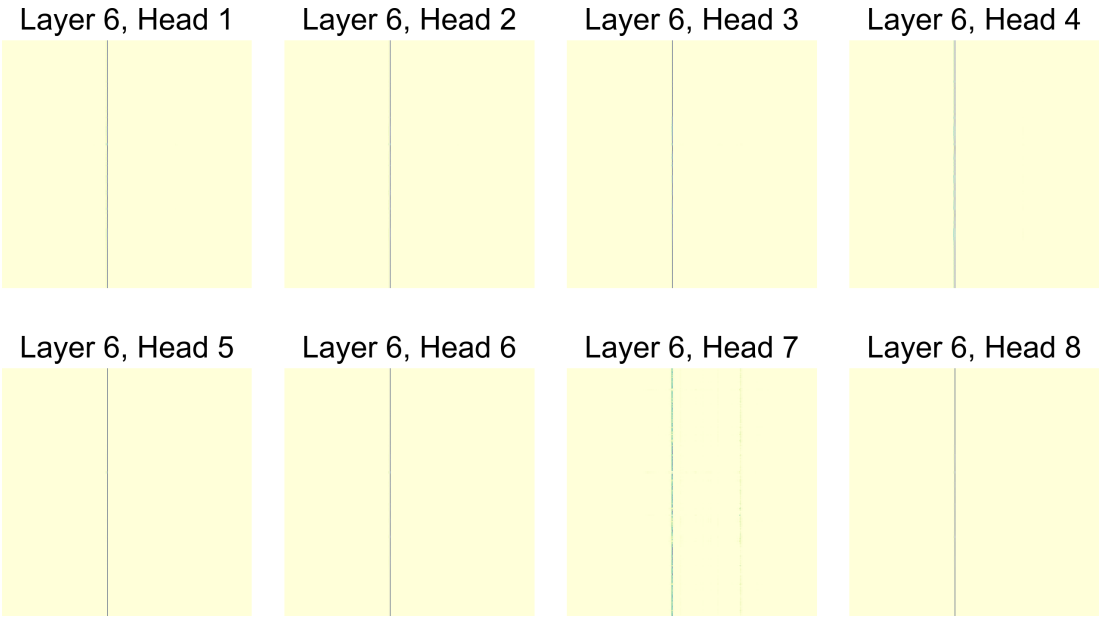


Figure 5.15: Attention maps of L6 of exc-exc-epoch30.

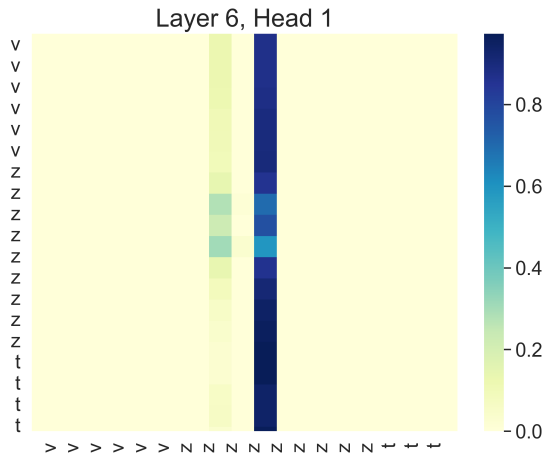


Figure 5.16: Partial attention map of L6H1 of exc-exc-epoch30.

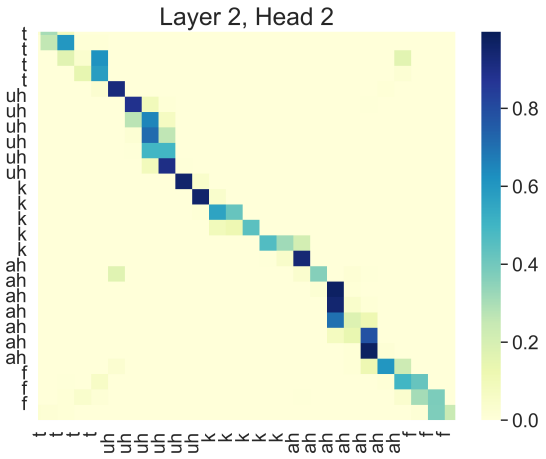


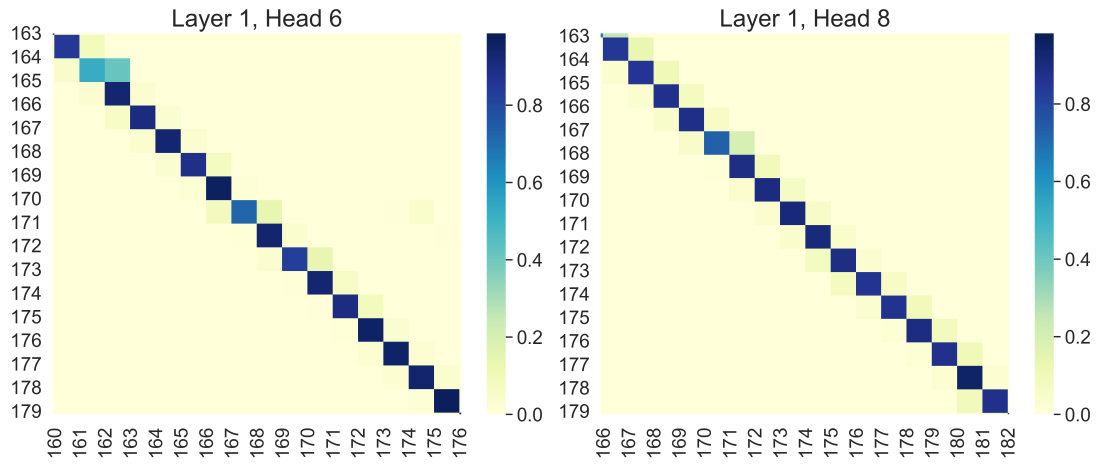
Figure 5.17: Partial attention map of L2H2 of exc-exc.

5.4.3 Difference of Diagonal Attention

L1H6 and L1H8 in inc-inc shows clear diagonal attention (Figure 5.18a, 5.18b). Frames pay confident attention to frames at three positions before them in L1H6 and frames at three positions after them in L1H8. It seems that the two heads learn relative position information, and distance three is especially useful. This finding is consistent with the property translation invariance of position embeddings proposed by [45]. A similar pattern is also found in L2H6. Each frame in the silence area of L2H6 pays attention to both four or five positions before it and two or three positions after it (Figure 5.18c). Non-silence area also suggests similar attention, while frames tend to restrict their attention to frames of the same classes (Figure 5.18d). This head appears to utilize the position information provided by heads L1H6 and L1H8, combined with feature information. It is expected to allow the transformer to detect if frames at a few positions far away belong to their phone classes as well.

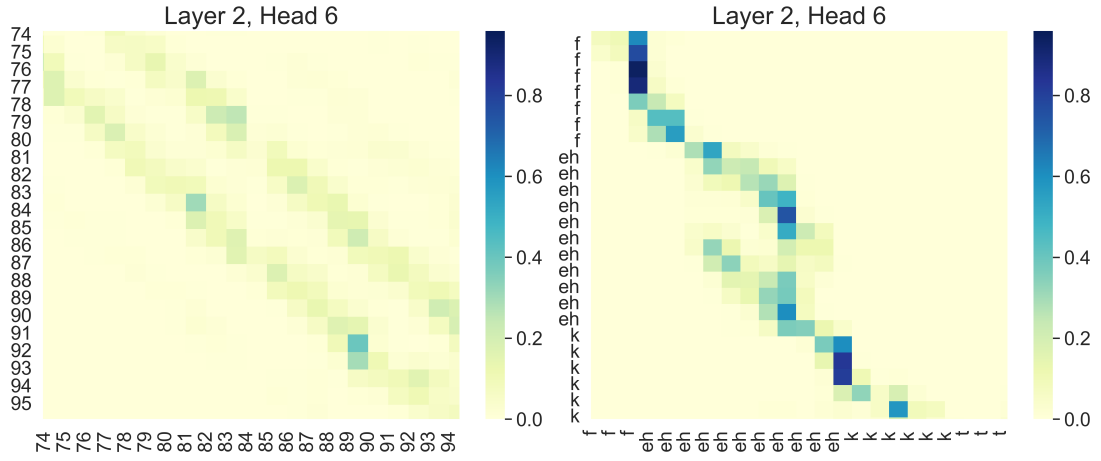
We also notice that frames rarely pay attention to themselves. It is most likely because the information of each frame itself has been provided by residual connections. Furthermore, frames tend to pay more attention to frames that are a few positions away. One possible explanation is that frames next to each other tend to have very similar features, and thus the additional information conveyed by them is limited and not as useful as the information of frames at two or three positions away.

We hypothesize that when the model makes the decision on a frame, it considers nearby frames. As a result, when it finds frames before the frame and after the frame



(a) Partial attention map of L1H6 of inc-inc.

(b) Partial attention map of L1H8 of inc-inc.



(c) Partial attention map of L2H6 of inc-inc, where frames are silence.

(d) Partial attention map of L2H6 of inc-inc, where frames are not silence.

Figure 5.18: Heads in inc-inc which learn position information.

belong to the same class, the model is inclined to believe the frame is also in that class. It becomes Hypothesis 5.2.

Hypothesis 5.2. Position embeddings reduce errors by improving the smoothness of the output.

Hypothesis 5.2 is discussed in section 5.5.

A similar but different attention pattern in exc-exc is shown in Figure 5.17, where the attention is approximately diagonal as well. It means that M-exc learns to identify neighbouring frames. Although it presents some position related information, we claim that

1. this position related information could not be obtained from the explicit position information since we pad position embeddings with zeros, and

2. this position related information seems hard to be obtained from the implicit order information since self-attention is expected to ignore orders.

Hence, the position related information is expected to derive from features. We hypothesize that the two adjacent frames tend to contain more similarities than others, resulting in frames in L2H6 of exc-exc paying attention to their neighbouring frames. It turns into Hypothesis 5.3.

Hypothesis 5.3. In transformers without position embeddings, frames identify neighbours based on features.

Hypothesis 5.3 is discussed in section 5.6.

5.5 Validation of Hypothesis 5.2

To validate Hypothesis 5.2, two experiments are designed. The first experiment aims to validate if position embeddings improve the smoothness of the output. The second experiment aims to explore how the smoothness can help to reduce errors.

5.5.1 Experiment I

In this experiment, we count the number of changes in phone classes for inc-inc and exc-exc. Besides the utterance that we picked when visualizing attention maps, we randomly pick another two utterances from the dev set. According to Table 5.4, the number of changes in exc-exc is much more than that in inc-inc. It means with position

Utterance Key	4b7c040p	4bgc031c	4ayc030p	Total
Length	375	424	752	1551
Class Changes in inc-inc	22	43	95	160
Class Changes in exc-exc	31	76	150	257

Table 5.4: Number of changes of the phone class in inc-inc and exc-exc. Length is the number of frames

embeddings, the transformer tends to reduce the changes in the output, and thus the smoothness is improved. It leads to Conclusion 5.6.

Conclusion 5.6. Position embeddings in the transformer improve the smoothness of the output.

5.5.2 Experiment II

In this experiment, we focus on the errors that cover no more than three continuous frames for the following reasons.

1. When errors cover too many continuous frames, they convey incorrect context information. The smoothness based on the incorrect context is not considered helpful for the model to make a correct prediction.

2. According to our previous findings in section 5.4, transformers learn distance three. In three continuous incorrect frames, the maximum distance is two, and M-inc is expected still able to learn context information outside these frames and avoid corresponding errors from happening.

We have the definitions below.

Neighbour of a frame: one frame before the frame or one frame after the frame.

Frame at the boundary: the frame which belongs to a different target phone class to at least one of its neighbours.

Frames at the boundary: a frame at the boundary, or a group of continuous frames where at least one of them is at the boundary.

Frames not at the boundary: a frame that is not at the boundary, or a group of continuous frames where none of them is at the boundary.

Errors covering no more than three continuous frames are classified into the following three types. Each error type follows examples.

Type 1 errors. In this case, frames at the boundary are incorrectly classified as the target class on the other side of the boundary (Table 5.5, 5.6).

Type 2 errors. The model incorrectly classifies the frames at the boundary as a phone class that is not the target phone class of either side of the boundary (Table 5.7).

Type 3 errors. Frames not at the boundary are misclassified (Table 5.8).

We assume that the context information may help the transformer avoid Type 2 and Type 3 errors because the frames in these two scenarios are misclassified as phone classes that don't fit in with their neighbours.

We count the number of frames that encounter the three types of errors in inc-inc and exc-exc respectively using the three utterances same to Experiment I. When finding a group of no more than three frames are misclassified, we take account in the errors, look for which errors they belong to, and count the errors by frames. Compared to exc-exc, inc-inc shows less Type 1 errors in utterances 4b7c040p and 4bgc031c, while utterance 4ayc030p suffers from more Type 1 errors in inc-inc than exc-exc. Compared to exc-exc, inc-inc makes fewer Type 2 errors.

Frame	178	179	180	181	182
Target	uh	uh	uh	k	k
Predict	uh	uh	k	k	k

Table 5.5: Type 1 error made by inc-inc in utterance 4b7c040p at frame 180.

Frame	316	317	318	319	320
Target	v	dh	dh	dh	ah
Predict	v	v	v	v	ah

Table 5.6: Type 1 error made by inc-inc in utterance 4bgc031c at frames 317 - 319.

Frame	99	100	101	102	103
Target	ah	ah	ah	r	r
Predict	ah	ah	v	v	r

Table 5.7: Type 2 error made by exc-exc in utterance 4ayc030p at frames 101 and 102.

Frame	228	229	230	231	232
Target	k	k	k	k	k
Predict	k	sil	sil	sil	k

Table 5.8: Type 3 error made by exc-exc in utterance 4bgc031c at frames 229 - 231.

According to our hypothesis, M-inc may tend to classify the frame at the boundary to the classes of its neighbours in order to keep the smoothness of the output. We hypothesize that this may reduce Type 2 errors but increase Type 1 errors to some extent, and it could be the reason that results in more Type 1 errors of inc-inc in utterance 4ayc030p. To validate our hypothesis, we compare the prediction of inc-inc and exc-exc frame by frame. According to Table 5.10, at frame 231, exc-exc makes a Type 2 error and classifies the frame /n/ to /l/, while inc-inc makes a Type 1 error and classifies the

Utterance Key		4b7c040p	4bgc031c	4ayc030p	Total
Length		375	424	752	1551
Type 1 errors	inc-inc	6	14	49	69
	exc-exc	15	22	39	76
Type 2 errors	inc-inc	0	0	3	3
	exc-exc	0	1	15	16
Type 3 errors	inc-inc	0	1	9	10
	exc-exc	3	20	18	42

Table 5.9: Statistics of three types of errors made by inc-inc and exc-exc in three utterances.

frame to the class of the later frame /ao/. We assume the feature at frame 231 could be confusing and somewhat similar to /l/. Considering the smoothness of the output, inc-inc tends to make a decision between /n/ and /ao/. In this way, Type 2 error is avoided while Type 1 error happens. Similar cases can also be found in Table 5.11. Position embeddings prevent M-inc from misclassifying the frame as /t/ and make it classify the frame to /d/ since it is the class of the previous frame.

Frame	229	230	231	232	233
Target	n	n	n	ao	ao
inc-inc	n	n	ao	ao	ao
exc-exc	d	d	l	ao	ao

Table 5.10: Type 1 error made by inc-inc and Type 2 error made by exc-exc at frame 231 in utterance 4ayc030p.

Frame	199	200	201	202	203
Target	d	d	ih	ih	ih
inc-inc	d	d	d	ih	ih
exc-exc	d	dh	t	t	ih

Table 5.11: Type 1 error made by inc-inc and Type 2 error made by exc-exc at frame 201 in utterance 4ayc030p.

In addition, inc-inc has fewer Type 3 errors. It means when the prediction of a set of frames of the same class is generally correct, it tends to avoid the mistakes in the middle of them (Table 5.12, 5.13). To be concluded, position embeddings improve the smoothness of the prediction, and the smoothness suggests effectiveness in reducing Type 2 errors and Type 3 errors. Our Experiment I and II together validate Hypothesis 5.2. Experiment II leads to Conclusion 5.7.

Conclusion 5.7. The smoothness of the output reduces specific types of errors.

Frame	356	357	358	359	360
Target	ow	ow	ow	ow	ow
inc-inc	ow	ow	ow	ow	ow
exc-exc	ow	ow	ow	ah	ow

Table 5.12: Type 3 error made by exc-exc in utterance 4ayc030p at frame 359.

Frame	25	26	27	28	29
Target	sil	sil	sil	sil	sil
inc-inc	sil	sil	sil	sil	sil
exc-exc	sil	sil	t	p	sil

Table 5.13: Type 3 error made by exc-exc in utterance 4bgc031c at frame 27 and 28.

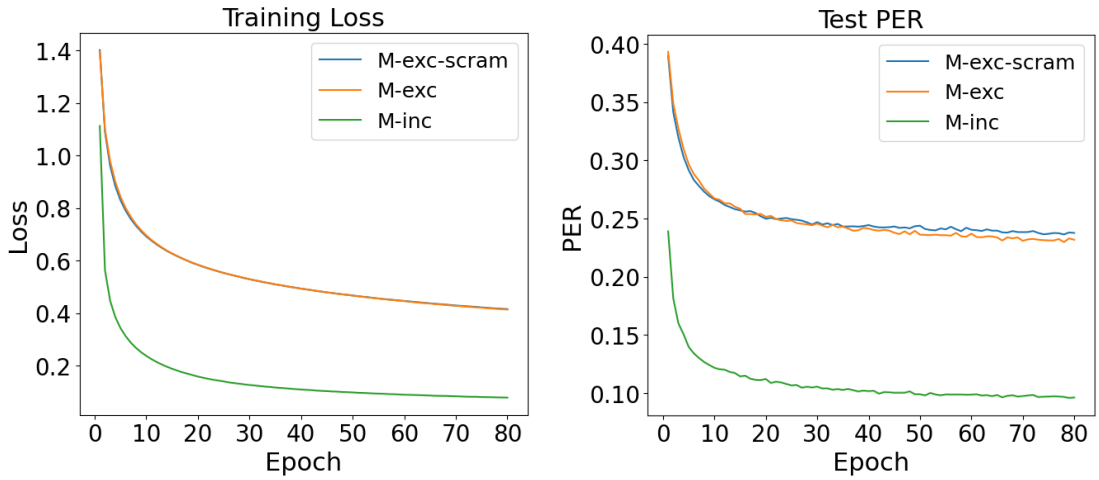
5.6 Validation of Hypothesis 5.3

To validate Hypothesis 5.3, we apply different permutations to different utterances at different epochs, and no position embeddings are fed into the model. In this way, the order information is removed, and the inputs fed into the transformer are only bags of frames. We assume if M-exc learns to identify neighbours based on orders of the inputs rather than features, the performance of the model would suggest an evident decrease when the order is scrambled. We name the model that is trained with scrambled inputs without position embeddings as M-exc-scram. Other model settings of M-exc-scram are the same as M-inc and M-exc.

Model	Final Training Loss	Best PER and Epoch
M-exc-scram	0.4151	23.64%, epoch 78
M-exc	0.4134	22.98%, epoch 78
M-inc	0.0782	9.60%, epoch 79

Table 5.14: Final training losses and Best PERs of M-exc-scram and previous models for comparison. Final training loss is the training loss at the 80th epoch. Best PERs are measured when models are tested with the same configurations to training. Epoch is the corresponding epoch of the best PER.

According to Figure 5.19 and Table 5.14, when the inputs are scrambled, the model does not suggest an evident decline in performance. When visualizing attention maps, it is found that the only diagonal attention is about each frame paying attention to itself (Figure 5.20), which contains no order related information. No attention maps in the scrambling case are found to contain diagonal order related attention. Hence, the transformer without position embeddings learn identify neighbours based on features



(a) Training loss.

(b) Test PER.

Figure 5.19: Training loss and test PER of M-exc-scram, M-exc and M-inc. Models are tested using the same configurations as training.

rather than the orders. It validates our Hypothesis 5.3 and leads to Conclusion 5.8 and 5.9.

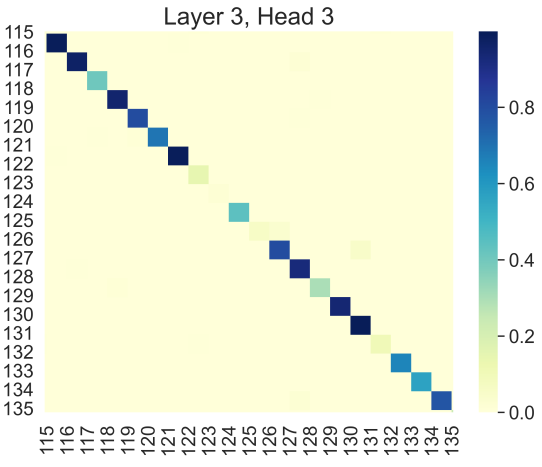


Figure 5.20: Partial attention map of L3H3 of M-exc-scram.

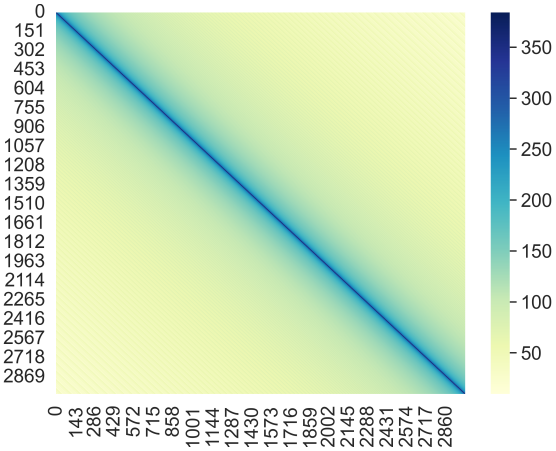
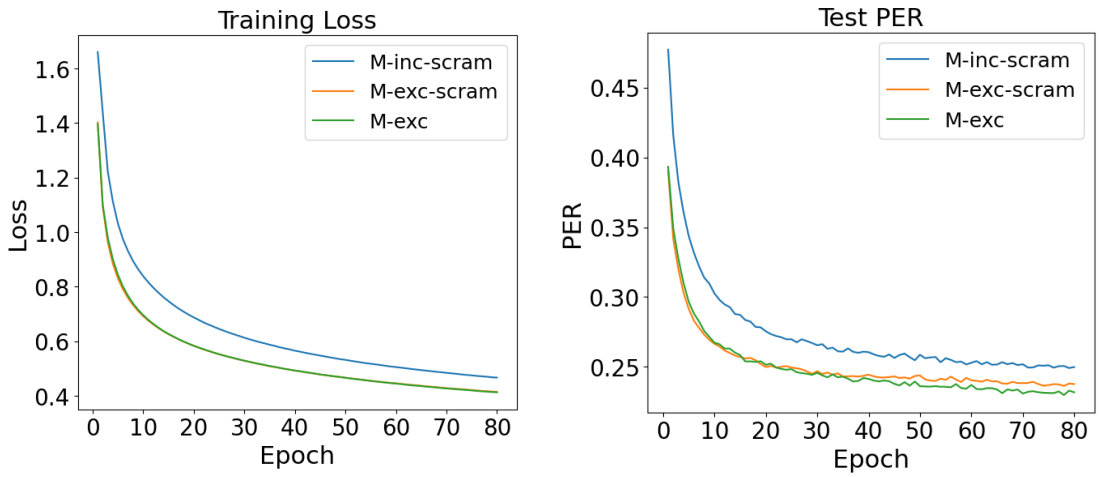


Figure 5.21: Similarity map of the fixed absolute sinusoid position embeddings proposed by [44] with length 3000.

Conclusion 5.8. The transformer without any order information can partially learn phone classification. The performance is similar to the transformer trained with unscrambled inputs but without position embeddings.

Conclusion 5.9. The transformer without position embeddings learns limited



(a) Training loss.

(b) Test PER.

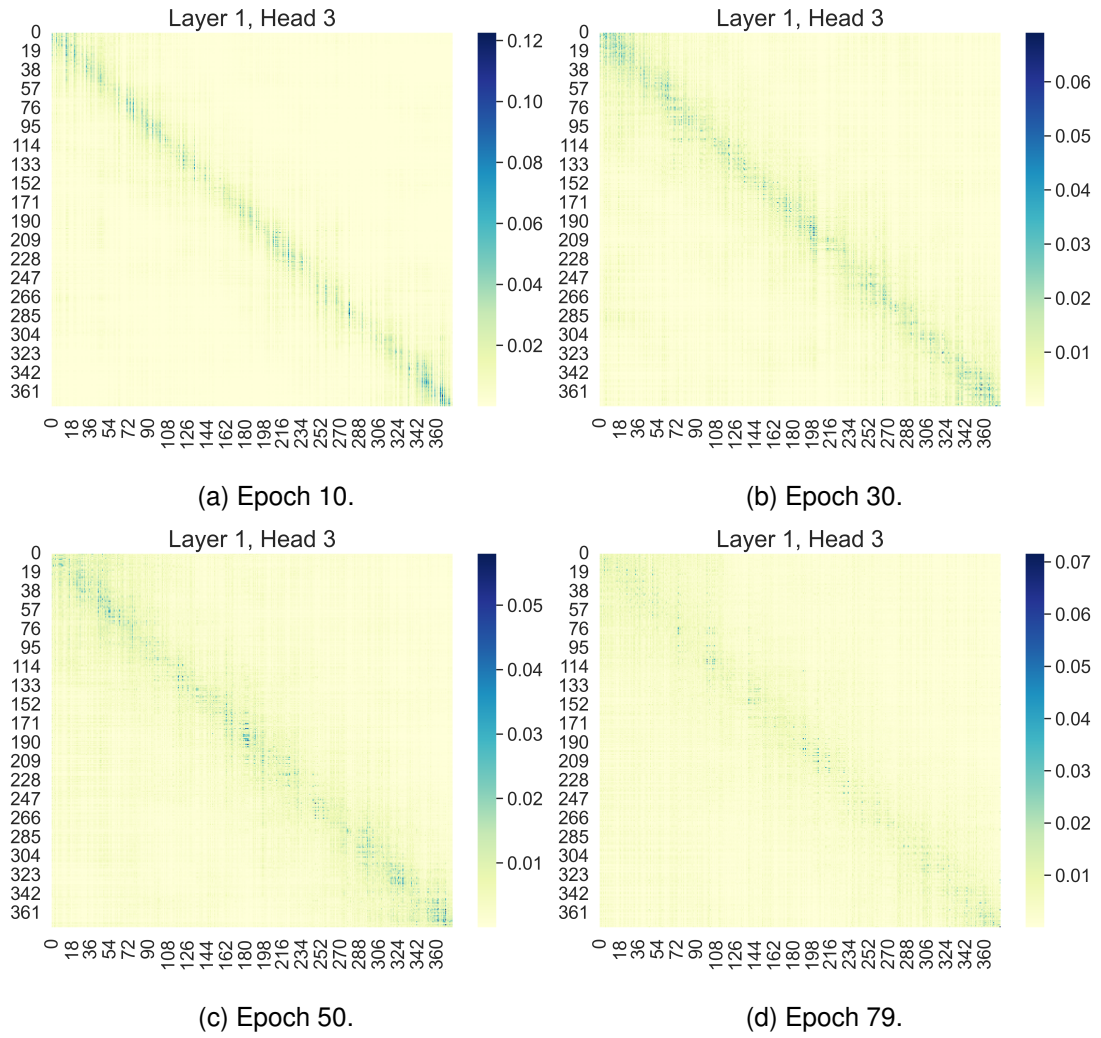
Figure 5.22: Training loss and test PER of M-inc-scram, M-exc-scram and M-exc. Models are tested using the same configurations as training.

position related information based on similarity of features to identify neighbours of frames.

5.7 Scrambled Inputs With Unscrambled Position Embeddings

We scramble the inputs and apply them with the unscrambled position embeddings. In this way, for the input, position embeddings carry disordered position information. We name the model that is trained with scrambled inputs and unscrambled position embeddings as M-inc-scram. Other model settings of M-inc-scram are the same as M-inc and M-exc. According to Figure 5.22, when unscrambled position embeddings are applied to the transformer with scrambled inputs, the training loss and the PER keep going down, whereas it is hard to achieve the performance of the transformer trained with scrambled inputs but no position embeddings. We hypothesize that when the model is provided with disordered position embeddings, it tries to ignore the position embeddings but could not totally ignore them.

To validate our hypothesis, we generate attention maps for epochs 10, 30, 50, and 79 respectively, finding that the diagonal attention gradually diminishes during training. We claim that the diagonal attention comes from position embeddings because the sequence is of scrambled order while position embeddings keep their original order. The gradually



(a) Epoch 10.

(b) Epoch 30.

(c) Epoch 50.

(d) Epoch 79.

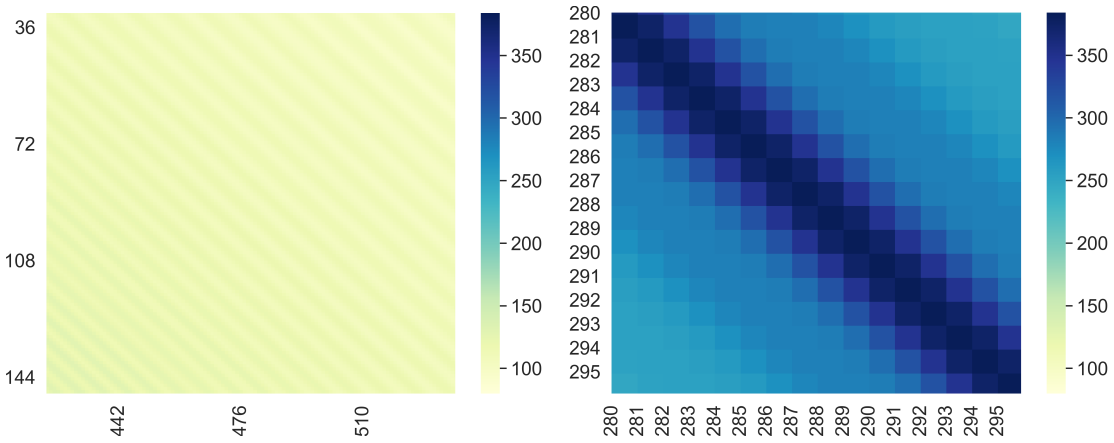
Figure 5.23: Attention maps of L1H3 of M-inc-scram across different epochs. M-inc-scram is tested with scrambled inputs and unscrambled position embeddings as well.

weakened diagonal attention confirms our hypothesis. It fails to completely disregard position embeddings but still makes their impact down to a low level. It leads to Conclusion 5.10.

Conclusion 5.10. Transformers applied with disordered position embeddings learn to lessen the impact of position embeddings, whereas they cannot entirely disregard position embeddings.

5.8 Similarity of Postion Embeddings

As described in section 4.5, we calculate the dot product similarity of the fixed absolute sinusoid position embeddings proposed by Vaswani et al. [44] as shown in Figure 5.21.



(a) Partial similarity map of position embeddings, where the similarity does not keep decreasing as the distance goes up.

(b) Partial similarity map of position embeddings where position embeddings of identical distance have the identical dot product similarity.

Figure 5.24: Enlarged similarity maps.

It can be found that the similarity of position embeddings generally decreases as the distance between positions grows. Shadow diagonal lines in the enlarged similarity map (Figure 5.24a), however, show that the similarity of position embeddings at some particular distances is higher than the similarity at distances closer to them. It indicates that the fixed absolute sinusoid position embeddings of [44] partially satisfy the property of monotonicity in [45]. We can also notice that position embeddings of the identical distance have the identical dot product (Figure 5.24b). It is in accordance with the property of translation invariance in [45]. It further demonstrates that the attention patterns in Figure 5.18a and 5.18b come from position embeddings.

From Figure 5.24b, the similarity changes sharply in the distances ranging from one to five. It implies that between distances from one to five, fixed absolute sinusoid position embeddings offer evident relative position information. It suggests that the help of position embeddings for long dependencies between frames is limited. Transformers, on the other hand, typically learn shorter distances, like three, in phone classification. It means that the transformers' demand for relative position information for phone classification can be met by the fixed absolute sinusoid position embeddings.

Chapter 6

Conclusions

In this project, we explore the role of position embeddings in transformers for phone classification, a task closely related to ASR. We eliminate position embeddings in transformers by padding them with zeros. Several experiments are carried out to compare transformers with and without position embeddings. We arrive at the following conclusions in the context of phone classification.

1. Position embeddings help the transformer to separate phones with similar voicing and to differentiate stops from silence. They are crucial in distinguishing /z/ from /s/. Position embeddings change the way that the transformer learns, especially how to identify /z/ and /s/. Position embeddings tell the transformer specific useful distances so that the smoothness of the output is improved. It suggests that the transformer learns relative position information from our absolute position embeddings. Errors that are thought to relate to the smoothness are reduced. When position embeddings carrying disordered sinusoid position information are fed to the transformer, it learns to reduce their influence but fails to totally ignore it.

2. Even trained without position embeddings, the transformer can distinguish a large number of phone pairs and maintain an acceptable phone error rate. The transformer trained without position embeddings cannot learn well on distinguishing phones with similar voicing or distinguishing stops from silence during training.

3. Without position embeddings, the transformer learns limited position related information based on features. It allows the transformer to identify neighbouring frames.

Due to the close relation between phone classification and ASR, we assume the role of position embeddings in transformers for phone classification also applies to ASR. Since it has been validated that the model learns relative position information from the absolute position embeddings, we also demonstrate the importance of convolutional networks in wav2vec 2.0 [2] and HuBERT [15].

In spite of the conclusions mentioned above, the project still has limitations. They are discussed in section 6.1. Work that can be done in the future is discussed in section 6.2.

6.1 Limitations

Our experiments focus on the fixed sinusoid absolute position embeddings proposed by [44]. It results in the following limitations.

1. We have shown that the transformer learns relative position information. It remains unknown if absolute position embeddings bring additional help compared to relative position embeddings.

2. Learned position embeddings are often employed in transformers as well [6, 8, 45]. We have not investigated that to what extent the learned position embeddings in transformers facilitate phone classification.

As a result, our conclusions on position embedding in transformers for phone classification are partial.

Moreover, our conclusions draw from attention maps are based on a single utterance. Our analysis in section 5.5 is based on three random utterances. Hence, the generalizability of our conclusions is limited.

The metric in section 5.5 has limitations as well. We classify errors into three types, assuming some are related to the smoothness, and count them. However, it is not always the case that the errors we count are the results of a lack of smoothness. Also, there could be other types of errors mitigated by the smoothness while we do not consider them. For example, we take account of errors covering no more than three continuous frames since we notice relative position three is learned by the transformer. However, since the model suggests to pay attention to frames on both sides, it is possible that the smoothness also suggests assistance in avoiding errors continuing four or five frames. Our metric is only a rough estimate.

We did not verify the downside of the position embeddings. Absolute position embeddings may make the model more easily overfit. Transformers trained with absolute position embeddings may rely too much on absolute position information when classifying phones. The smoothness of the output may bring negative effects within an incorrect context as well.

6.2 Future Work

According to the limitations in section 6.1, the following work can be done in the future.

1. Conduct experiments on learned absolute position embeddings in transformers for phone classification.

2. Experiments can be done to explore if absolute position embeddings in transformers provide additional help compared to relative position embeddings. It has been shown that relative position embeddings have more advantages than absolute position embeddings [17, 23, 5], especially for sequences that are longer than those in the training set [34]. It is also claimed that the sinusoid position embeddings may decrease performance on long utterances if they contain similar acoustic features at different positions [55].

3. Extend the experiments to ASR to validate to what extent our conclusions hold.

Bibliography

[1] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

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

[3] William Chan, Daniel Park, Chris Lee, Yu Zhang, Quoc Le, and Mohammad Norouzi. Speechstew: Simply mix all available speech recognition data to train one large neural network. *arXiv preprint arXiv:2104.02133*, 2021.

[4] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg

Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017.

[10] Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. A position-aware bidirectional attention network for aspect-level sentiment analysis. In *Proceedings of the 27th international conference on computational linguistics*, pages 774–784, 2018.

[11] Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[12] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

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

[16] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[17] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. *arXiv preprint arXiv:2009.13658*, 2020.

[18] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? *arXiv preprint arXiv:2001.08248*, 2020.

[19] Naoyuki Kanda, Guoli Ye, Yashesh Gaur, Xiaofei Wang, Zhong Meng, Zhuo Chen, and Takuya Yoshioka. End-to-end speaker-attributed asr with transformer. *arXiv preprint arXiv:2104.02128*, 2021.

[20] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

[21] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[22] Xinjian Li, Siddharth Dalmia, Juncheng Li, Matthew Lee, Patrick Littell, Jiali Yao, Antonios Anastasopoulos, David R Mortensen, Graham Neubig, Alan W Black, et al. Universal phone recognition with a multilingual allophone system. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8249–8253. IEEE, 2020.

[23] Tatiana Likhomanenko, Qiantong Xu, Gabriel Synnaeve, Ronan Collobert, and Alex Rogozhnikov. Cape: Encoding relative positions with continuous augmented positional embeddings. *Advances in Neural Information Processing Systems*, 34:16079–16092, 2021.

[24] Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*, 2019.

[25] Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *International conference on machine learning*, pages 6327–6335. PMLR, 2020.

[26] Yang Liu, Alexandras Neophytou, Sunando Sengupta, and Eric Sommerlade. Cross-modal spectrum transformation network for acoustic scene classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 830–834. IEEE, 2021.

[27] Diego Jesús Lozano-Mejía, Enrique Paul Vega-Uribe, and Willy Ugarte. Content-based image classification for sheet music books recognition. In *2020 IEEE Engineering International Research Conference (EIRCON)*, pages 1–4. IEEE, 2020.

[28] Tomohiro Nakatani. Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration. In *Proc. Interspeech*, 2019.

[29] Jinhwan Park, Chanwoo Kim, and Wonyong Sung. Convolution-based attention model with positional encoding for streaming speech recognition on embedded devices. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 30–37. IEEE, 2021.

[30] Martin Popel and Ondřej Bojar. Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*, 2018.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[33] Shaghayegh Reza, Seyyed Ali Seyyedsalehi, and Seyyedeh Zohreh Seyyedsalehi. Persian language phone recognition based on robust extraction of acoustic landmarks. In *2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)*, pages 106–112. IEEE, 2020.

[34] Jan Rosendahl, Viet Anh Khoa Tran, Weiyue Wang, and Hermann Ney. Analysis of positional encodings for neural machine translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, 2019.

[35] Saurabh Sahu, Vikramjit Mitra, Nadee Seneviratne, and Carol Y Espy-Wilson. Multi-modal learning for speech emotion recognition: An analysis and comparison of asr outputs with ground truth transcription. In *Interspeech*, pages 3302–3306, 2019.

[36] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 2 (Short Papers), pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[37] Kyuhong Shim, Jungwook Choi, and Wonyong Sung. Understanding the role of self attention for efficient speech recognition. In *International Conference on Learning Representations*, 2021.

[38] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. Self-attentional acoustic models. *arXiv preprint arXiv:1803.09519*, 2018.

[39] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *International Conference on Machine Learning*, pages 9625–9635. PMLR, 2020.

[40] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Towards online end-to-end transformer automatic speech recognition. *arXiv preprint arXiv:1910.11871*, 2019.

[41] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Transformer asr with contextual block processing. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 427–433. IEEE, 2019.

[42] KG Van Leeuwen, P Bos, Stefano Trebeschi, Maarten JA van Alphen, Luuk Voskuilen, Ludi E Smeele, Ferdi van der Heijden, RJJH Van Son, et al. Cnn-based phoneme classifier from vocal tract mri learns embedding consistent with articulatory topology. In *Interspeech*, pages 909–913, 2019.

[43] Jan Vaněk, Josef Michálek, and Josef Psutka. Recurrent dnns and its ensembles on the timit phone recognition task. In *International Conference on Speech and Computer*, pages 728–736. Springer, 2018.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in {bert}. In *International Conference on Learning Representations*, 2021.

[46] Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. Self-attention with structural position representations. *arXiv preprint arXiv:1909.00383*, 2019.

[47] Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6874–6878. IEEE, 2020.

[48] Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

[49] Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. Nezha: Neural contextualized representation for chinese language understanding. *arXiv preprint arXiv:1909.00204*, 2019.

[50] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10033–10041, 2021.

[51] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.

[52] Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE, 2019.

[53] Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. Pushing the limits of semi-supervised learning for automatic speech recognition. *arXiv preprint arXiv:2010.10504*, 2020.

[54] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[55] Pan Zhou, Ruchao Fan, Wei Chen, and Jia Jia. Improving generalization of transformer for speech recognition with parallel schedule sampling and relative positional embedding. *arXiv preprint arXiv:1911.00203*, 2019.