

MASTER'S THESIS COMPUTING SCIENCE

Wav2vec 2.0 inside out

MARK TEN KLOOSTER
s4821122

July 20, 2023

First supervisor/assessor:
dr. L.F.M. ten Bosch

Second assessor:
Prof. dr. M.A. Larson

Abstract

In the last few years, end-to-end models have become state-of-the-art for performing automatic speech recognition. In particular, Transformer-based models, such as Wav2vec 2.0, perform significantly better than hybrid models. However, these models still work as a black box and are difficult to interpret.

In this research project, we have investigated how the Wav2vec 2.0 model responds to literal changes in CVC words. We created a dataset consisting of CVC words that were recorded by a male and a female speaker. From this dataset, we extracted all minimal pairs and fed those words pairwise to the Wav2vec 2.0 model. Using the hidden vectors of the hidden Transformer layers, we calculated the cosine similarity at each layer between the hidden activations. We have found that changing a character mainly affects layers in the front of the Wav2vec 2.0 network. Later in the network, the layers turned out to be less susceptible to a changing character. Another outcome of our research is that changing a character at one of the outer positions causes a larger drop in cosine similarity than changing the inner character of a CVC word. We also found that these effects occur with different speakers, and also with and without using a carrier phrase to initialise the model.

Contents

1	Introduction	3
1.1	End-to-end ASR models	3
1.2	Explainable AI	4
1.3	Wav2vec 2.0	4
2	Related Work	6
2.1	Phonetics in E2E models	6
2.2	Transformer-based models	7
3	Method	9
3.1	Dataset	11
3.2	Comparing two words	11
3.2.1	Dynamic Time Warping	12
3.2.2	Similarity of vectors	12
3.3	Quantifying a comparison	14
3.3.1	Valley position	14
3.3.2	Depth of the valley	14
3.3.3	Standard deviation of the cosine similarity	15
3.4	Pilot experiment	15
4	Results	17
4.1	Cosine similarities between words	17
4.1.1	Timestamp of minimum cosine similarity	17
4.1.2	Minimum and standard deviation of the cosine similarities	19
4.2	Aggregated results	19
4.2.1	Position of the valley	19
4.2.2	Standard deviation of the cosine similarity	20
5	Discussion	22
5.1	Effects of a different character	22
5.1.1	Discussion	23
5.2	Consonants versus vowel	23
5.2.1	Discussion	23

5.3	Speakers and carrier phrase	24
5.4	Dataset generation	24
5.5	Cosine similarity	25
6	Conclusions and Future Work	27
6.1	Reflection and acknowledgements	28
A	Heatmaps pilot experiments	32
B	Results	33

Chapter 1

Introduction

In the last few years, artificial intelligence and machine learning have received a great amount of interest from both the research community and the industry and end users. Many applications can now be solved by modern models trained on massive amounts of data. One of the foundations of the success of these models is the introduction of the Transformer model by Vaswani et al. in [1]. This new architecture enabled machine learning engineers to create models that take into account more long-term dependencies, while at the same time improving training performance. This resulted in new state-of-the-art models in many domains.

1.1 End-to-end ASR models

In the domain of automatic speech recognition (ASR), the field is moving away from hybrid modelling models to end-to-end (E2E) models. As described by Li et al. in [2], the E2E models offer some significant advantages compared to traditional hybrid methods. First and foremost, E2E models consist of only one single function responsible for the complete processing from audio to a sequence of text. Originally, hybrid models had many different separated parts, each doing a small function in the whole pipeline. This paradigm change greatly simplified ASR models, while also making them easier to develop, implement, and maintain. Furthermore, E2E models typically perform better than hybrid models, both in research and in real-world applications.

The currently best performing E2E ASR models also are based on a Transformer-style architecture. As noted by Li et al. in [2] and Mehrish et al. in [3], Transformer-based models outperform RNN- and LSTM-based models. However, both research papers point to the various challenges faced with such E2E models. It turns out that speaker adaptation and domain adaptation are challenges because models tend to perform worse on data with different characteristics than the data on which they were trained.

Also, because Transformers were initially designed to handle textual data, Transformers expect sequential text data with clearly separated tokens. With speech data, the data is continuous in the time dimension, and inherently there is no clear separation between tokens as words and phones naturally flow over in each other. Therefore, additional measures must be taken in the design of an E2E network to overcome this problem.

Another aspect is the large variety that can exist in speech data. In contrast to textual data, the same token can be expressed in many different ways. For example, tokens can be pronounced differently by different speakers or spoken at different speeds. These aspects require ASR models to be invariant for different speakers and small variations in linguistic features.

1.2 Explainable AI

Although E2E models currently are state-of-the-art, many E2E models also work as a black box. This black box problem has started to become a major challenge, since E2E models based on artificial neural networks outperform rule-based models from the past. As discussed in the literature (e.g. [4], [5], [6], [7]), the explainability of AI models is an important topic that receives a lot of attention from the research community. Many research papers describe the definition of explainable AI and what it entails. Also, the importance of explainable AI, and in some domains the legal requirements for explainable AI, is a recurring theme throughout the literature.

Zhang et al. in [7] describe three dimensions in which the interpretability of an AI model can be defined. First, they distinguish active methods from passive methods. Second, they suggest four formats that can be used to explain a model. Third, they define an ordinal scale to specify how broad an explanation is. For example, does the explanation give insight into one training example or does it describe a high-level characteristic that applies to the model as a whole? We take this taxonomy as a basis to describe our approach.

1.3 Wav2vec 2.0

Baevski et al. introduced the Wav2vec 2.0 model in 2020 [8]. This model has a Transformer-based architecture, combined with a Convolutional feature encoder followed by the Transformer layers. The model has been pre-trained with unlabelled data where parts of the input are masked at the level of the feature encoder. After pre-training, the model is fine-tuned for speech recognition with the LibriSpeech and TIMIT datasets. The authors show that the Wav2vec 2.0 model achieves state-of-the-art performance while using 100 times less labelled data.

In our research, we are interested in the hidden Transformer layers of the Wav2vec 2.0 model. We have chosen to use the Wav2vec 2.0 model because it showed good performance when we started with our research project, and because the model is fully open source and freely available. We try to find out how the model responds to small changes in the input that are clear to a human listener but can be very subtle for an artificial model. To do this, we will look at the Transformer layers and see what patterns we can find in the internal activations of these layers at different depths. For example, if two words are very similar to each other and only differ in one phone, somewhere the model should find which phone is different and how it should be encoded into text. Following the taxonomy defined by Zhang et al. in [7], we use a passive method to interpret the hidden semantics of the Wav2vec 2.0 model with global interpretability. In chapter 3, we will discuss our method in more detail. In conclusion, the main research question for this research project is:

- To what extent differ activation patterns through the hidden layers of Wav2vec 2.0 for words that are different in one character?

To answer the main research question, various sub questions need to be answered:

- How to create a suitable dataset of words that differ only in one character (minimal pairs) that can be used to trigger predictable activations in the Wav2vec 2.0 model? For example, word pairs like (*mum*, *mam*).
- How to measure the similarity between internal activations of the Transformer layers?
- How can we analyse how the internal activations are affected by minimal pairs, based on the similarity of those activations at a certain layer?

Chapter 2

Related Work

In the literature, many research papers can be found that investigated how machine learning models generate their answers. In the last few years, surveys have been done in [4], [5], [9], [10], [6], and [7]. A common theme throughout these research papers is the discussion of the importance of explainable AI. For example, Arrieta et al. in [6] surveyed the literature on methods to explain and interpret different kinds of machine learning models. They also describe six higher-level principles for explainable AI: fairness, privacy, accountability, ethics, transparency, and security/safety.

In addition to a better interpretation of the model, knowledge about the interpretation of a model can also be used in the development of new models. For example, one use case is to adapt a model to other tasks than what it was originally trained on. Knowledge of the internal layers of a network can then be useful in determining whether the complete network is needed or if a part of the network is sufficient. Sharif Razavian et al. in [11] and Yosinski et al. in [12] discuss this transferability of layers in Convolutional Neural Networks. Both research papers conclude that trained models for one task can be useful for training a model in another task. They found that the first layers in a network capture general features that are not necessarily specific to one task. Even in tasks that are very different from the original task, the authors found a benefit from transferring parts of a model. They describe that using trained model weights from a different task yields better results than starting with randomly initialised model weights.

2.1 Phonetics in E2E models

Before the rise of end-to-end ASR models, explicit training data about the phonetics of a language was used to train ASR models. Because E2E models are trained in an unsupervised manner without explicit phonetic data, many research projects have been carried out to investigate how E2E ASR models learn to capture phonetic information. Studies like [13] and [14] have

investigated (among others) how different hidden layers in a model help to find phonetic features in an audio fragment.

Belinkov et al. in [13] used a supervised classifier to predict phone labels from different hidden layers in the DeepSpeech2 E2E model. The authors found significant changes in accuracy throughout the network, with spikes after the first convolutional layer and in the final recurrent layers. Nagamine et al. analysed how nodes of hidden layers in a deep neural network respond to different phonetic features. They found that both individual nodes and groups of nodes respond to different phonetic features. Another interesting finding is that errors made by the deep neural network often occur with phonemes that share the same phonetic features, similar to mistakes that a human would make.

Li et al. in [15] also investigated the behaviour in different hidden layers by erasing parts of the representation. The authors set parts of the input word representations to 0, or in other experiments set parts of the hidden representations to 0. If the authors then observe a decrease in performance, they can conclude that their change was at an important point in the representation. If the output gets better, then it is a sign that the change was at a point where the model pays too much attention. By applying this method at different places in the model, the authors can give an explanation for many aspects that are important for ASR. The authors conclude that their findings provide an efficient and generally applicable tool that can be used to interpret and explain models based on neural networks.

2.2 Transformer-based models

In 2022, English et al. in [16] researched the hidden layers in the Wav2vec 2.0 model. They focused on the Transformers within the model and performed probing experiments on each of the Transformer layers. To do these experiments, they trained 12 multi-layer perceptron models as probing classifiers that were used to predict phoneme labels from the TIMIT dataset. The goal of their research was to find out how the hidden layers of the network respond to different phonetic categories. The authors found that the Transformer layers can catch relevant phonetic details, and that layers at different depths behave differently for similar-sounding phones.

Ma et al. in [17] investigated to which extent pre-trained models such as wav2vec and DeCoAR are able to capture phonetic properties. They performed probing experiments on the output layer of a model with five different tasks: speech activity detection, sonorant detection, vowel detection, fricative detection, and phone classification. The authors found that pre-trained models are able to represent detailed phonetic information, outperforming established methods that use MFCCs to represent the raw audio.

In contrast to the experiments in [17], Li et al. in [18] researched the

hidden representations of E2E ASR models. They used two LSTM-based ASR architectures, VGG-LSTM and pure-LSTM, to probe the hidden layers. In their experiments, the output of each hidden layer was extracted from the model. The extracted hidden representation was fed to an evaluation model to reconstruct the detected speech at that specific layer. The result was then compared with the original input. The authors found that layers near the output layer of the network produce more abstract representations than layers near the raw audio input. For example, background noise appears to be removed early on in the models. The authors conclude that their results are consistent with the existing literature, showing that their approach of reconstructing the speech from hidden layers is sound.

In [19], Shah et al. studied the hidden representations of the transformer layers in Wav2vec 2.0 and Mockingjay. The authors performed probing experiments for three different categories of features: audio, fluency, and pronunciation. The models were tested with three different kinds of speech types that differ in quality and spontaneity. Their results show that both Wav2vec 2.0 and Mockingjay have their own patterns across their hidden layers. It also differs per feature category, and even individual features behave in interesting patterns throughout the network. In many of the experiments, the results show that the last hidden layer does not perform better than all of the previous layers. Often, the best performance is achieved by layers in the second half of the network.

Chapter 3

Method

In our research, we will investigate where the hidden activations in the Wav2vec 2.0 model change when we change a character in a word. To do this, we have designed an experiment in which we will compare the activations of the hidden layers for pairs of input tokens.

To be able to relate a change in the input word tokens to a change in the hidden layers, we need a pair of words to have some amount of overlap. If a pair consists of two identical word tokens and the audio files are completely identical, the network will show the same activations for both tokens. When two completely different tokens are fed to the network, the activations will also be very different from each other. Therefore, to be able to relate a change in the input tokens to differences in the activations, we will only use pairs of words that differ by exactly one character.

To ensure that the results of different pairs can be compared with each other, we will only use input tokens that have a fixed number of characters and follow a pre-defined pattern. This ensures that any differences in activations are caused by the character that has changed.

Figure 3.1 visually shows our experimental setup with the Wav2vec 2.0 model. In this chapter, we discuss the complete pipeline in more detail. First, we discuss the dataset that we will use and how it has been generated. Second, we discuss how we compare tokens with each other and hidden activations of the Wav2vec 2.0 model. Third, we discuss how we quantify the results and how we have prepared our experiments with a pilot experiment.

All experiments were performed with Python 3.10. The code and the full list of words in the dataset can be found in our accompanying GitHub repository¹.

¹<https://github.com/mtkweb/master-thesis>

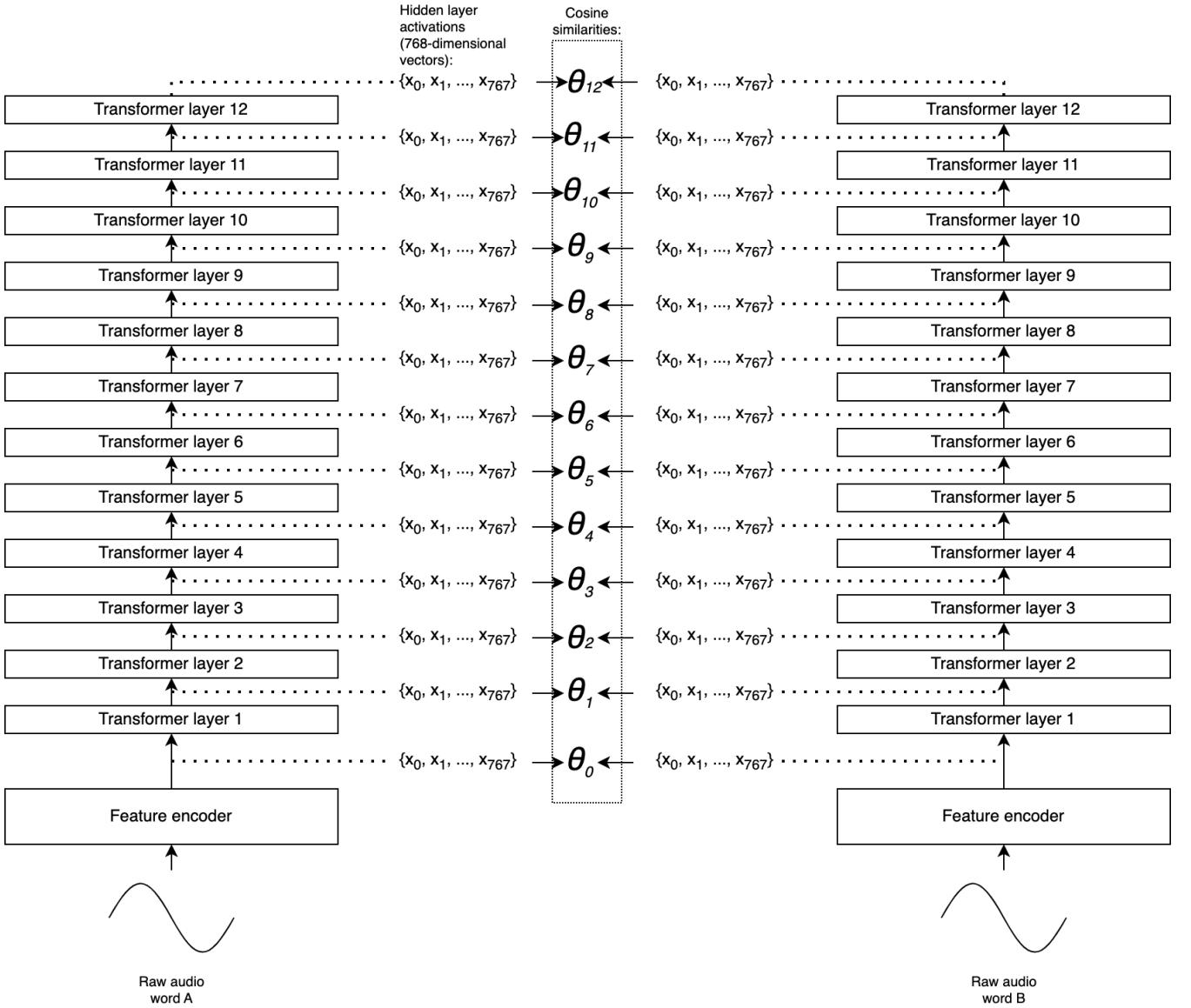


Figure 3.1: Schematic overview of the Wav2vec 2.0 model and comparing two words by calculating the cosine similarity at each layer. This figure shows one time step of the Wav2vec 2.0 model. The cosine similarities are calculated for all time steps in the alignment of the two words.

3.1 Dataset

We have recorded a custom dataset for our research. By doing so, we were able to exactly match the needs of our research and ensure the quality of the data. Our dataset consists of audio recordings of words following the consonant-vowel-consonant pattern (CVC words) based on the set $C = \{f, k, m, n, p, s, t\}$ for the consonants and the set $V = \{a, e, i, o, u\}$ for the vowel. This results in a dataset of 245 unique three-letter word types in total. These word types may or may not be existing words, for example, some word types from the dataset are *fat*, *nef*, *sum*.

To obtain a correct recording for each word, we recorded each word four times. The words were recorded in groups of four words plus one extra redundant word. These list-final redundant words were omitted from our analysis to avoid typical end-of-list intonation patterns playing a role in the words' make-up. From the first four recordings, we calculate the average duration and use the recording closest to the average duration in our experiments. This prevents slow or fast recordings from being used in the experiments and ensures that the experiments are based on the most accurate recordings available in the dataset. The whole recording process has been done twice, both by a male speaker and by a female speaker.

The words were recorded using a Shure SM58 microphone and a Focusrite Scarlett 2i2 audio interface with a sample rate of 16 kHz. The raw audio files that were recorded were split using the Pydub library [20], based on the silence between the word tokens. In total, our dataset consists of 1225 recorded audio fragments, each containing exactly one word token.

In addition, we also recorded a carrier phrase. We will use this carrier phrase in some of our experiments preceding the word recording. This audio concatenation results in a longer recording that helps to initialise the Wav2vec 2.0 model. Both speakers recorded the same carrier phrase '*The next word is*'. During the experiment, we prepend the carrier phrase to the recording and feed the resulting audio fragment to the model. With this, we make sure that a recording always is preceded by the carrier phrase from the same speaker. When analysing the hidden layers, we remove a fixed number of time steps equal to the length of the carrier phrase from the model's output. This ensures that we only analyse the activations for the word itself. Because the carrier phrase always has a fixed length, the number of time steps removed will be the same throughout the experiment.

3.2 Comparing two words

As discussed earlier, we only compare pairs of words that differ by exactly one character (minimal pairs). Formally stated, this entails all pairs of words of which the Hamming distance in spelling is 1 [21]. Because all words in

our dataset have a fixed length of three characters, we can distinguish three cases. The words within a pair differ at the first, second, or third character, respectively. In the Results chapter, we will distinguish these cases with separate plot lines. With 245 unique word types in our dataset, we get 1960 minimal pairs.

3.2.1 Dynamic Time Warping

Although all words have a fixed length of 3, the lengths of the audio fragments differ because the time required to pronounce a phone differs per phone. For our research, we are interested in comparing the activations of the hidden layers at time t_1 in word A with the activations at time t_2 in word B, such that t_1 and t_2 are at the same relative moment within their word. Therefore, we calculate the best alignment of two words before comparing the activations of the hidden layers.

For example, the words *sof* and *tof* differ by the first character. In this case, there is also a clear difference in the time required to pronounce the '*s*' character versus the '*t*' character. Figure 3.2 shows how the recordings of these words can be aligned. The plot clearly shows a different alignment at the beginning of the recordings compared to the rest of the recordings.

For each pair of word tokens, we calculate this alignment once, based on the activations of the feature vector layer (layer 0). We will then use this alignment to compare the activations of the other layers with each other.

To find the best alignment, we use Dynamic Time Warping (DTW), as implemented in the *dtw* Python package [22]. DTW is a technique to align two temporal sequences that have a different speed such that the start and end of both words are aligned exactly and that the path in between is continuous and does not go back in time. Its application in speech recognition was first researched in the 1970s by Sakoe et al. in [23].

As a cost function for the DTW algorithm, we use the similarity of the feature vectors generated by the convolutional layers in the Wav2vec 2.0 model. As shown in figure 3.1, this is the first layer of the Wav2vec 2.0 model. In the next section, we will discuss the similarity calculation of vectors in more detail.

3.2.2 Similarity of vectors

In our research, we use the base model of Wav2vec 2.0 . This model contains 12 transformer blocks, each having a model dimension of 768. Figure 3.1 also shows a schematic overview of the Wav2vec 2.0 model and its Transformer layers. Including the first feature encoding layer of the model, this gives us 13 768-dimensional vectors at each time step. With the calculations of the DTW algorithm, we can determine which time steps of two recordings

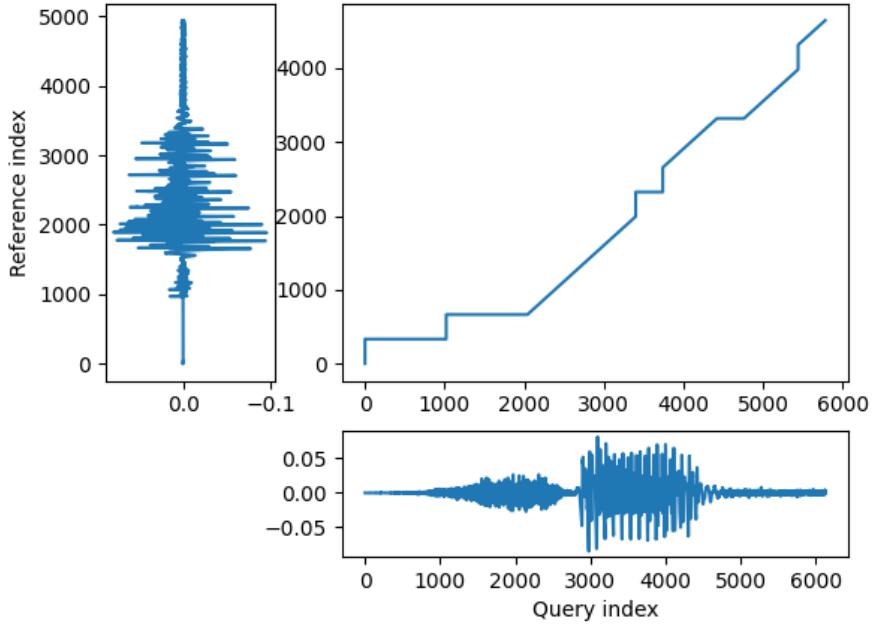


Figure 3.2: Plot of the alignment between the words *sof* (Query index) and *tow* (Reference index). The plot shows both waveforms of the recordings, and the best alignment between them, based on the feature vector layer of the Wav2vec 2.0 model. Both axes show the number of samples in the recording.

should be compared with each other. This comparison consists of calculating the similarity of the vectors at each of the transformer layers.

As a similarity measure, we use the cosine similarity. The cosine similarity quantifies the similarity by the cosine of the angles between the two vectors. The equation for the cosine similarity is shown in equation 3.1. Because the cosine similarity is equal to 1 for vectors pointing in the same direction, we can verify our process by making a comparison between the activations of two model predictions of the same audio fragment. Because the audio data is identical throughout the comparison, the activations will be equal, and therefore the cosine similarity will always be equal to 1.

$$\text{cosine_similarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \quad (3.1)$$

To further test the suitability of using the cosine similarity, we performed a test comparison between two identical audio fragments but changed the amplitude of one of them to half the original amplitude. This difference in

amplitude does not change the phonetic content of the recording, and it turns out that the cosine similarities still remained equal to 1 in all layers. From this and based on the fact that the model normalises the raw waveform [8], we conclude that the convolution layer in front of the Transformer layers filters out this difference when using the cosine similarity as a similarity measure. This ensures our idea to use the cosine similarity to determine to what extent two vectors represent the same token.

In addition to using cosine similarity, there are other mathematical ways to compare two vectors. For example, the dot product and the Euclidean distance could be used. However, contrary to the cosine similarities, these measures also take into account the length of the vectors. While on itself this can give useful information about the model, it does make it more difficult to compare the similarity across multiple layers. We assume that the length of the activation vectors may vary between layers. Therefore, the dot product or Euclidean distance also gets a different range. Using the cosine similarity is therefore favourable, since it gives a fixed reference for similar vectors (namely a similarity of 1).

3.3 Quantifying a comparison

Now that we can measure to which extent vectors are similar, we need a method to measure how much a comparison has a valley, at which relative position that valley occurs, and to which extent the cosine similarities vary. To measure these aspects, we extract three metrics for each layer for each comparison we make.

3.3.1 Valley position

We expect the position of the valley to depend on the position of the character that is different within a comparison. For example, if we compare again the words *sof* and *tof*, we expect to see a valley in the cosine similarity plot near $t = 0$. Figure 3.3 shows this comparison. We quantify the position of the valley by applying the *argmin* function to all cosine similarities at each layer. For each comparison between two words, this gives us 13 values (the feature vector layer plus 12 transformer layers) that represent where the smallest cosine similarity occurs at a certain layer.

3.3.2 Depth of the valley

Besides the relative position of the valley, we are also interested in the depth of the valley. A large depth implies a low cosine similarity at the deepest point in the valley. This means that at that point in time, the Wav2vec 2.0 model generates very different activations at a layer. We measure this

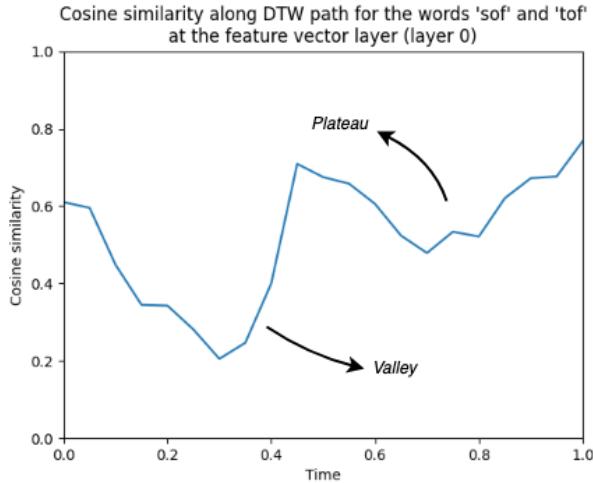


Figure 3.3: Plot of the cosine similarity at the feature vector layer between the words *sof* and *t of*. The x-axis shows the time normalized from 0 to 1, and the y-axis shows the cosine similarity.

aspect by simply applying the *min* function to all cosine similarities at each layer. This also gives us 13 values.

3.3.3 Standard deviation of the cosine similarity

Thirdly, we investigate the spread of the cosine similarity values by calculating the standard deviation. We expect the standard deviation to be independent of where exactly the valley occurs. This would indicate that the results are consistent across all word pairs for the regions of the valleys, as well as for the regions outside the valleys.

3.4 Pilot experiment

To test the feasibility of our method, we performed a pilot experiment before starting with the complete dataset and the entire pipeline. We first created a small dataset with 3-character words following the consonant-vowel-consonant pattern using these characters: {n,m}{a,i}{n,m}. This resulted in a dataset of 8 words. We recorded the pronunciation of these words and fed the audio fragments to the base Wav2vec 2.0 model. Then, we extracted the activations of the hidden layers of the model and compared the vector sequences throughout the duration of the audio fragment with the vector sequences of the same layers for another similar-sounding word.

To compare two vectors, we calculated the cosine similarity between the two vectors. Each vector in a sequence is compared with each other vector

in the other sequence. This results in a two-dimensional matrix of cosine similarities. The high values in this matrix indicate where the sequences match best and to which extent they match. From this, one can find the best alignment between the two vectors. Table A in Appendix A shows the similarities for two pairs of similar-sounding words at four different layers in the network, plotted as heatmaps.

The heatmaps clearly show some interesting patterns. The heatmaps from layer 4 and layer 7 do not have clear differences that indicate what the phonetic difference is between the compared words. Later in the network, layer 11 apparently clearly shows which phone is different. Interestingly, this information is not visible anymore in the final layer, which seems to do some form of encoding of the phones that were found.

Chapter 4

Results

In this chapter, we present the results of our experiments. First, we will present the empirical results of the cosine similarities throughout the duration of a word comparison. In section 4.2, we will show more high-level results, aggregated from the raw comparison data based on cosine similarities. We will only show the most important plots in this chapter. All other plots can be found in Appendix B.

4.1 Cosine similarities between words

4.1.1 Timestamp of minimum cosine similarity

With the cosine similarities between each minimal pair of words, we calculate three metrics: the timestamp of the minimum cosine similarity, the cosine similarity at that timestamp, and the standard deviation of all cosine similarities in a comparison of two words.

Table 4.1 shows the results at each layer of the timestamp of the minimum cosine similarity based on the recordings of the male speaker, without carrier phrase. Because the audio fragments are different in length, we have normalised the timestamps so that all the audio fragments start at time = 0, and end at time = 1. The results are grouped by the position at which there is a character difference between two words, with 0, 1 and 2 meaning that two words differ respectively at the first, second, or third character. We calculated both the mean and the standard deviation. In Appendix B we have included the data with carrier phrase, and the data based on the recordings by the female speaker.

We also plotted the results on a line chart to make them more intuitive. The plot on the left of figure 4.1 shows the similarity scores for all word pairs at the feature vector layer as recorded by the male speaker. The normalised time can be read from the x-axis and the y-axis shows the cosine similarity, ranging from 0 to 1. The plot contains the cosine similarity values for all word pairs as described in section 3.2. Each plot line has an error bar

Different at Layer	Timestamp of minimum mean					
	mean			std		
	0	1	2	0	1	2
0	0.31	0.54	0.76	0.25	0.27	0.15
1	0.31	0.56	0.75	0.25	0.25	0.13
2	0.30	0.57	0.73	0.23	0.23	0.13
3	0.30	0.55	0.73	0.22	0.22	0.13
4	0.34	0.55	0.76	0.26	0.26	0.16
5	0.35	0.49	0.68	0.26	0.26	0.19
6	0.47	0.57	0.70	0.32	0.29	0.22
7	0.53	0.58	0.67	0.33	0.31	0.28
8	0.58	0.59	0.65	0.35	0.34	0.32
9	0.57	0.61	0.67	0.33	0.32	0.28
10	0.39	0.50	0.59	0.26	0.29	0.26
11	0.52	0.55	0.58	0.22	0.19	0.17
12	0.35	0.43	0.49	0.18	0.19	0.18

Table 4.1: Results of the timestamp of the minimum cosine similarity at each layer. This data is from the recordings by the male speaker, and without carrier phrase. For each layer, the table shows for each character position (0, 1, or 2) the mean and the standard deviation of where the lowest point of the valley occurs on the normalised time axis.

showing the range of values at each time step. The error bars are configured so that 50 percent of the data points are within the error bar. This plot of the first layer in the Wav2vec 2.0 model roughly follows the literal character difference between two words. For example, the plot showing the data points for comparisons between words that differ at the first positions has a valley early on in time, before showing increasing similarity scores as a plateau later on. The same pattern can be seen in the other two plot lines.

We have also included another plot from a Transformer layer near the output of the model. The right plot in figure 4.1 shows the cosine similarities at layer 11. While the plot of the feature vector layer follows the literal differences between two words, the 11th Transformer layer has its own pattern. The three plot lines now roughly follow the same path, with a small valley around the centre.

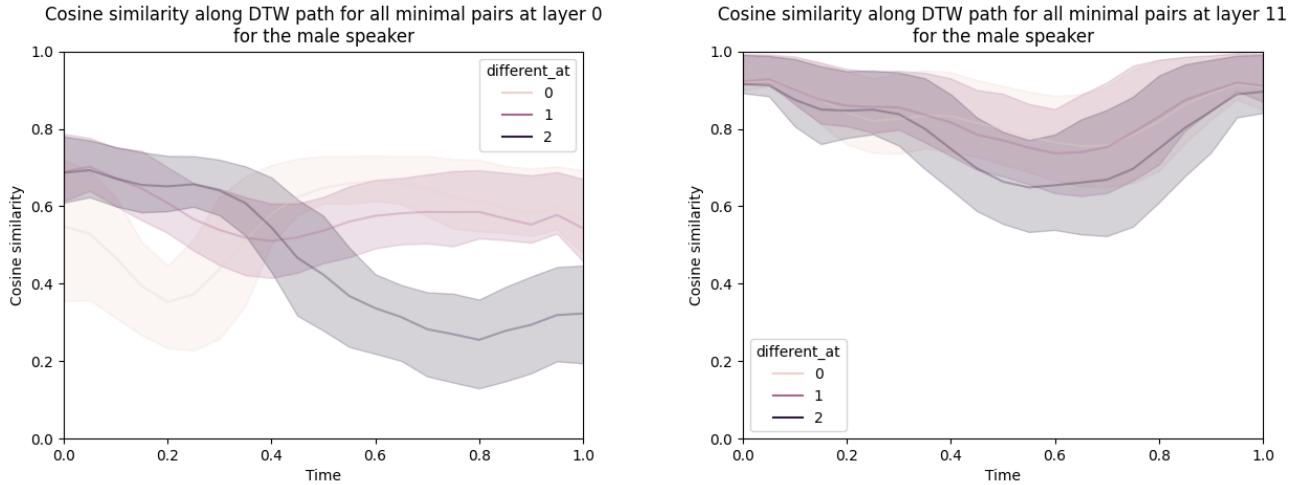


Figure 4.1: Cosine similarities at the feature vector layer and the 11th Transformer layer. The plots are based on the recordings by the male speaker, and without carrier phrase. Both plots also contain error bars to show the range in which 50 percent of the data points are.

4.1.2 Minimum and standard deviation of the cosine similarities

In this section, we present the results that show how strong the effects of a different character are. Table 4.2 shows the mean and standard deviation of the minimum cosine similarity and the standard deviation of the cosine similarities at each layer, for each character position based on the recordings by the male speaker without carrier phrase. Appendix B shows the rest of the data for these metrics.

4.2 Aggregated results

4.2.1 Position of the valley

As explained in section 3.3, we calculate three numeric values for each layer in all comparisons. Figure 4.2 shows the correlation at each layer between the timestamp of the minimum cosine similarity and the position at which the two words had a different character. As described, the timestamp of the minimum cosine similarity defines where the valley is located. Each line in the plot represents a part of the dataset, where each part is from one of the two speakers, with or without the carrier phrase.

The plot can be interpreted as follows: A high correlation coefficient means that the further in time the position of the valley gets, the higher the index of the character being different (with the index ranging from 0 to 2).

Different at Layer	Minimum similarity						Standard deviation of similarity					
	mean			std			mean			std		
	0	1	2	0	1	2	0	1	2	0	1	2
0	0.20	0.34	0.13	0.13	0.14	0.12	0.17	0.12	0.21	0.04	0.05	0.05
1	0.33	0.43	0.23	0.11	0.13	0.11	0.13	0.10	0.18	0.03	0.04	0.04
2	0.36	0.44	0.26	0.10	0.12	0.11	0.13	0.10	0.17	0.03	0.03	0.04
3	0.38	0.45	0.28	0.10	0.11	0.12	0.13	0.11	0.17	0.03	0.03	0.04
4	0.37	0.44	0.26	0.11	0.12	0.13	0.14	0.12	0.18	0.04	0.03	0.05
5	0.34	0.39	0.26	0.11	0.11	0.11	0.16	0.15	0.18	0.04	0.03	0.04
6	0.29	0.32	0.21	0.11	0.12	0.12	0.17	0.17	0.20	0.04	0.04	0.04
7	0.23	0.26	0.18	0.12	0.13	0.12	0.20	0.19	0.20	0.04	0.04	0.04
8	0.02	0.05	-0.01	0.11	0.15	0.12	0.26	0.24	0.26	0.05	0.05	0.05
9	0.14	0.16	0.09	0.10	0.13	0.10	0.21	0.20	0.23	0.05	0.05	0.05
10	0.26	0.30	0.21	0.10	0.12	0.11	0.17	0.15	0.19	0.04	0.04	0.05
11	0.59	0.59	0.50	0.18	0.18	0.18	0.11	0.11	0.14	0.06	0.06	0.07
12	0.22	0.27	0.21	0.15	0.17	0.15	0.18	0.17	0.19	0.05	0.05	0.05

Table 4.2: Results of the minimum cosine similarity and the standard deviation of the cosine similarity at each layer. This data is from the recordings by the male speaker, and without carrier phrase. For each layer, the tabel shows for each character position (0, 1, or 2) the mean and the standard deviation of the minimum cosine similarity of the valley, and of the standard deviation in a comparison.

4.2.2 Standard deviation of the cosine similarity

The plots in section 4.1 seem to indicate that the valleys are much deeper when one of the consonants differs versus when the vowel differs. The data in figure 4.3 partially confirms this hypothesis, by plotting the correlation coefficients per layer.

To calculate these correlation coefficients, we first pre-processed the data such that all comparisons that are different in the vowel get a value of 1, and the comparisons that differ in one of the consonants get a value of 2. We then calculated the correlation coefficients between this value and the standard deviation of the cosine similarities for each layer. In this case, a high correlation coefficient means that the standard deviation will be higher if the two words in a comparison are different in one of the consonants. In turn, a high standard deviation implies a deeper valley compared to cosine

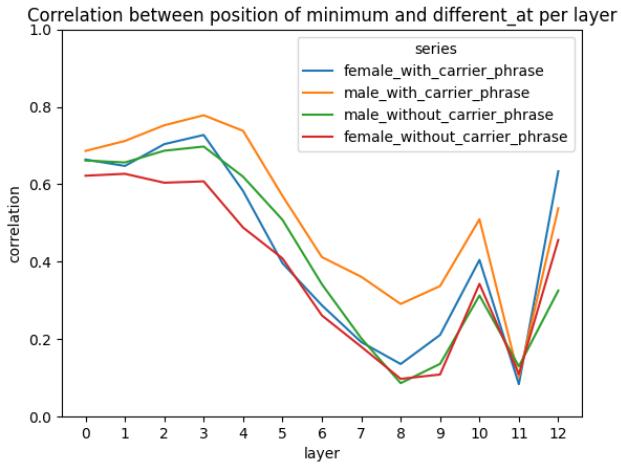


Figure 4.2: Correlation at each layer between the minimum cosine similarity, and the position at which the two words differ. This plot shows the data for both speakers, with and without a carrier phrase.

similarities outside the valley.

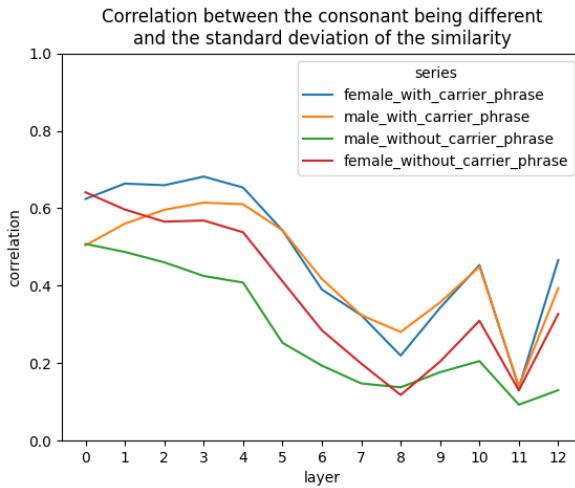


Figure 4.3: Correlation at each layer between the consonants being different, and the standard deviation of the cosine similarities. This plot shows the data for both speakers, with and without a carrier phrase.

Chapter 5

Discussion

We will discuss the results of our experiments in this chapter, followed by a discussion about how the choices made in the generation of the dataset and the design of the experiment affected the results.

5.1 Effects of a different character

The fundamental design of our research is to investigate how the model behaves when two almost identical words are compared to each other. Our results show that the position of the character being different can be seen in the internal layers of the Wav2vec 2.0 network. This effect can be seen in both the empirical plots of the cosine similarities and in the correlation coefficients between the minimum cosine similarity and the index of the different character.

Based on these correlation coefficients, our research shows two main effects. First of all, we found that the effects of a different character can be seen all the way from the feature vector layer to the fourth Transformer layer. The correlation coefficients in this part of the network are around 0.5, before it quickly drops significantly to lower values at higher layers. This suggests that the layers in the last part of the network encode more specific information for the speech recognition task compared to the layers near the input side of the network. The higher correlation coefficients of the first few layers indicate that these layers encode information that is more specific for the input fragments, whereas the layers in the last part take into account more context about the input fragment as a whole. This finding corresponds to other works in the literature.

Secondly, the plot of the correlation coefficients shows a higher correlation at the tenth and twelfth transformer layers, compared to neighbouring layers. In particular, the eleventh transformer layer shows a significant drop in the correlation coefficients. While it is not clear why this happens at the tenth layer, for the twelfth transformer layer, this can be explained since

that layer also provides the basis for the model’s output. It is then trivial to see that there must be some correlation between a character being different in the input and a character being different in the output.

5.1.1 Discussion

While a clear pattern can be seen with respect to where exactly the character is different, the correlation coefficients do not exceed about 0.8 for this relation. Also, some of the series (for example the recordings by the female speaker without carrier phrase) show significantly lower values across all layers than other series.

One cause of this pattern can be the way the recordings are aligned to each other. While in general the alignment process to align two recordings performs well, there might be inconsistencies in that process. For example, if there is a speed difference in speaking between two recordings, the alignment might end up slightly different. Later on, this also influences the values of the cosine similarities and, in turn, the correlation coefficients.

Another factor that plays a role in this is how the valley is detected in a comparison. In the plots as shown in the chapter 4, the valleys are detected by applying the *argmin* function to all the cosine similarities in a comparison. For most comparisons, this works well. However, there are some comparisons that have some extreme values for the cosine similarity near $t = 0$ or $t = 1$. The *argmin* function then returns one of those values for t , instead of the time when the actual valley occurs.

5.2 Consonants versus vowel

An observation that can be made based on the plots of the cosine similarity is that the word pairs that differ at the vowel character do not show a clear valley. Based on where the valleys are for word pairs that differ in one of the outer characters, we expected to also see a clear value around $t = 0.5$ for word pairs that differ in the inner character.

This observation is confirmed by the correlation coefficients between the consonants being different and the standard deviation of the cosine similarities. Again, the lower-level layers near the input of the model show higher correlation coefficients than the higher-level layers near the output. Overall, we can conclude that until the fourth transformer layer the comparisons that differ at the vowel have a much smaller valley than at later layers.

5.2.1 Discussion

The reason we see this effect might be related to the way the pronunciation works phonetically. Vowels, in general, all have a similar sound. Therefore, CVC-words that only differ at their vowel also have relatively small

differences in how they sound. Also, the consonants and the vowel must transition naturally to each other, causing the consonants to affect how the vowel sounds. Such phonetic aspects can be the reason we do not observe any valleys near the centre of the comparisons.

5.3 Speakers and carrier phrase

To make sure our findings are well-founded, we have recorded two speakers for our dataset. Based on all the empirical and aggregated results, we found that there are no significant differences between the speakers. Although the results show some small differences, our main findings can be clearly observed from both series of results.

Another measure we took to ensure the validity of our results is the use of a carrier phrase. Because Transformer-based models such as Wav2vec 2.0 take context into account, it might make a difference to prepend a carrier phrase in front of the word that will be fed to the network. In this case, there is also no big difference compared to the series that did not use the carrier phrase. However, the figures in section 4.2 do show slightly higher correlation coefficients when a carrier phrase was used. Based on this, we expect that the Wav2vec 2.0 model does benefit from extra context in front of the audio for which a transcription is needed, but specially designed experiments are needed to confirm this.

5.4 Dataset generation

Not related to any results in particular, there are some remarks about the dataset that are worth discussing. When we recorded the dataset, we recorded each word four times. This helps to reduce anomalies in the data because one can choose the best recording that will be used in the experiments. We selected a recording by comparing the duration of the recordings and then choosing the recording with the duration closest to the average duration. This approach worked well in our research, but there are other methods worth exploring in further research. For example, several recordings could be combined into one audio fragment, or choosing the best recording could be based on more advanced audio features.

During our research, we also found some inconsistencies in the splitting of words that end with a 'k' character. The sound of this character consists of two parts with a very short break between them. It turned out that our splitting algorithm sometimes inserted a split into this break, effectively shortening the recording before the pronunciation of a word was finished. We expect this to be caused by the fact that the splitting algorithm we have used is based on the sound level and the amount of silence between words. This accidental split then occurred when the speed in the recording was a

bit slower or when the last part of the 'k' character had a very low sound level. More advanced approaches can help prevent this, but require more research into the characteristics of the recordings.

5.5 Cosine similarity

All of our results are based on the cosine similarity between hidden activations. As explained in section 3.2.2, we have chosen this measure because it provides a fixed value for fully similar vectors, making it easier to compare the similarities between multiple layers. To validate the results of our experiments, we have repeated the calculation with the dot product as a similarity measure. Figure 5.1 shows the correlation at each layer between the position of the valley and the position at which the two words differ, now based on the dot product as a measure of similarity between the hidden activation vectors.

The correlation coefficients follow the same pattern as the cosine similarity, as shown in section 4.2. There are some notable differences though, mainly visible in the layers near the output of the model. Firstly, the sudden increase in similarity at layer 10 with the cosine similarity is less present with the dot product. Second, the drop in similarity at layer 11 is larger with the dot product than with the cosine similarity as a measure.

Despite these differences, the main plateau at the layers near the model's input side is still very present and significant. Also, both similarity measures show similar scores for the last Transformer layer. Taking all this into account, we found that both measures would have been suitable for this research project.

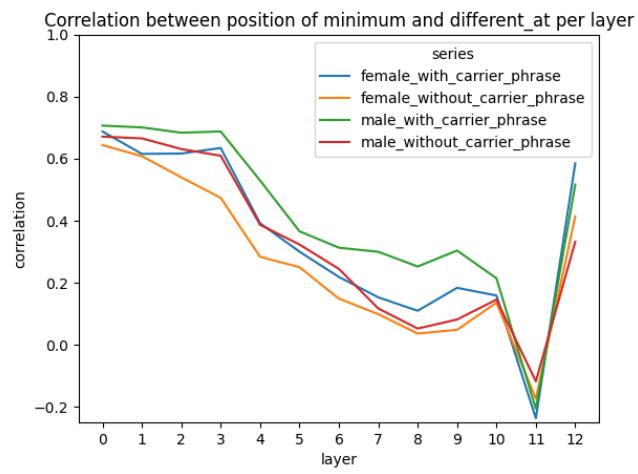


Figure 5.1: Correlation at each layer between the position of the valley, and the position at which the two words differ. This data is based on the dot product as similarity measure between the hidden activation vectors.

Chapter 6

Conclusions and Future Work

In this research, we have performed various experiments to empirically investigate how the Wav2vec 2.0 model responds to literal character changes in words. Before we did the experiments, we have composed a dataset of CVC words and recorded the dataset with two speakers. Our results show that there is a clear correlation in the first four layers of the network between the position of the character that was changed and the drop in cosine similarity. Layers higher up in the network have a much less clear response to a change in one character. Another finding is that the middle character causes a much more subtle effect compared to the outer characters.

We also found that these effects occur for both the male and the female speaker. This confirms that these effects are not just a characteristic that happens for one specific speaker, but that it generalises to other speakers as well. Future research could create a dataset with more speakers to investigate to what extent these effects occur when speakers are even more diverse. With regard to the carrier phrase, the effects are visible but minor. Throughout our results, the series with carrier phase got slightly higher correlation coefficients, but the series without a carrier phrase still follow largely the same patterns. This indicates that using a carrier phrase indeed helps to initialise the model. Future research could use this finding to research whether a carrier phrase can have a bigger impact when using multiple carrier phrases of different lengths.

There are some other aspects that future research could look at. First, future research could repeat the same research on other end-to-end ASR models that are based on Transformers. Second, the dataset used in our research could be extended with longer words to investigate whether the model shows the same patterns. A possible extension of this direction would be to also change more characters at once.

6.1 Reflection and acknowledgements

In the last 8 months, we have made the full journey for this thesis project. Starting with just a vague idea, to working on a concrete experimental setup with this thesis document as a final end product. When we look back at all the steps we took, we can conclude that it has been very instructive in many different ways. To name a few things, reading all the literature for this project, and working on the necessary Python code provided some great experience for the future. With regard to the literature and related works, our research setup turned out to be a unique way of investigating the Wav2vec 2.0 model. While on itself it should not be a problem, it does make it harder to compare our results to existing works. Many works in the literature investigated related topics, but we could not find a paper with a comparable research method. For future projects, we will consider this factor more heavily in the process of developing an experimental setup. In this project, we are satisfied with the results that we obtained during our pilot experiment, which turned out to be a good foundation for the rest of our experimental setup.

Now that I have completed this thesis project, I would like to thank the people who helped me finish this project. First of all, I want to thank Louis as my daily supervisor for all the help with this project. I really appreciate all the good conversations that we had to choose a topic, to develop the research method, and to write the thesis. I also want to thank Henrieke as the female speaker for helping me with the recordings and making the dataset more robust.

Bibliography

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [2] J. Li *et al.*, “Recent advances in end-to-end automatic speech recognition,” *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.
- [3] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria, “A review of deep learning techniques for speech processing,” *Information Fusion*, p. 101869, 2023.
- [4] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.
- [5] A. Adadi and M. Berrada, “Peeking inside the black-box: a survey on explainable artificial intelligence (xai),” *IEEE access*, vol. 6, pp. 52138–52160, 2018.
- [6] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai,” *Information fusion*, vol. 58, pp. 82–115, 2020.
- [7] Y. Zhang, P. Tiño, A. Leonardis, and K. Tang, “A survey on neural network interpretability,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021.
- [8] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in neural information processing systems*, vol. 33, pp. 12449–12460, 2020.
- [9] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, “A survey of methods for explaining black box models,” *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

- [10] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.,” *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
- [11] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.
- [12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *Advances in neural information processing systems*, vol. 27, 2014.
- [13] Y. Belinkov and J. Glass, “Analyzing hidden representations in end-to-end automatic speech recognition systems,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [14] T. Nagamine, M. L. Seltzer, and N. Mesgarani, “Exploring how deep neural networks form phonemic categories,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [15] J. Li, W. Monroe, and D. Jurafsky, “Understanding neural networks through representation erasure,” *arXiv preprint arXiv:1612.08220*, 2016.
- [16] P. C. English, J. Kelleher, and J. Carson-Berndsen, “Domain-informed probing of wav2vec 2.0 embeddings for phonetic features,” in *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 83–91, 2022.
- [17] D. Ma, N. Ryant, and M. Liberman, “Probing acoustic representations for phonetic properties,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 311–315, IEEE, 2021.
- [18] C.-Y. Li, P.-C. Yuan, and H.-Y. Lee, “What does a network layer hear? analyzing hidden representations of end-to-end asr through speech synthesis,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6434–6438, IEEE, 2020.
- [19] J. Shah, Y. K. Singla, C. Chen, and R. R. Shah, “What all do audio transformer models hear? probing acoustic representations for language delivery and its structure,” *arXiv preprint arXiv:2101.00387*, 2021.
- [20] jiaaro, “Github repository pydub,” commit 996cec4, 2022.

- [21] R. W. Hamming, “Error detecting and error correcting codes,” *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.
- [22] T. Giorgino, “Computing and visualizing dynamic time warping alignments in r: the dtw package,” *Journal of statistical Software*, vol. 31, pp. 1–24, 2009.
- [23] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

Appendix A

Heatmaps pilot experiments

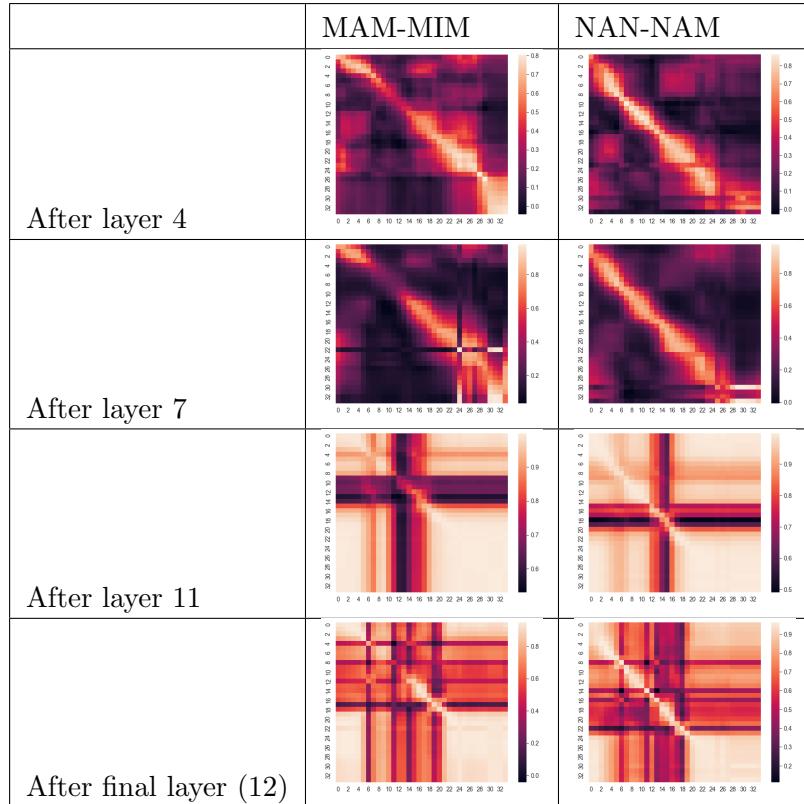


Table A.1: Cosine similarities between vectors of 4 hidden layers for two word pairs. In all heatmaps, both the x and y axes show the time steps of the word recordings. The color at each data point indicates the value of the cosine similarity between the activations, with a higher color indicating a higher cosine similarity.

Appendix B

Results

Table B.4: Cosine similarities at all layers. The plots are based on the recordings by both speakers, and without carrier phrase. Both plots also contain error bars to show the range in which 50 percent of the data points are.

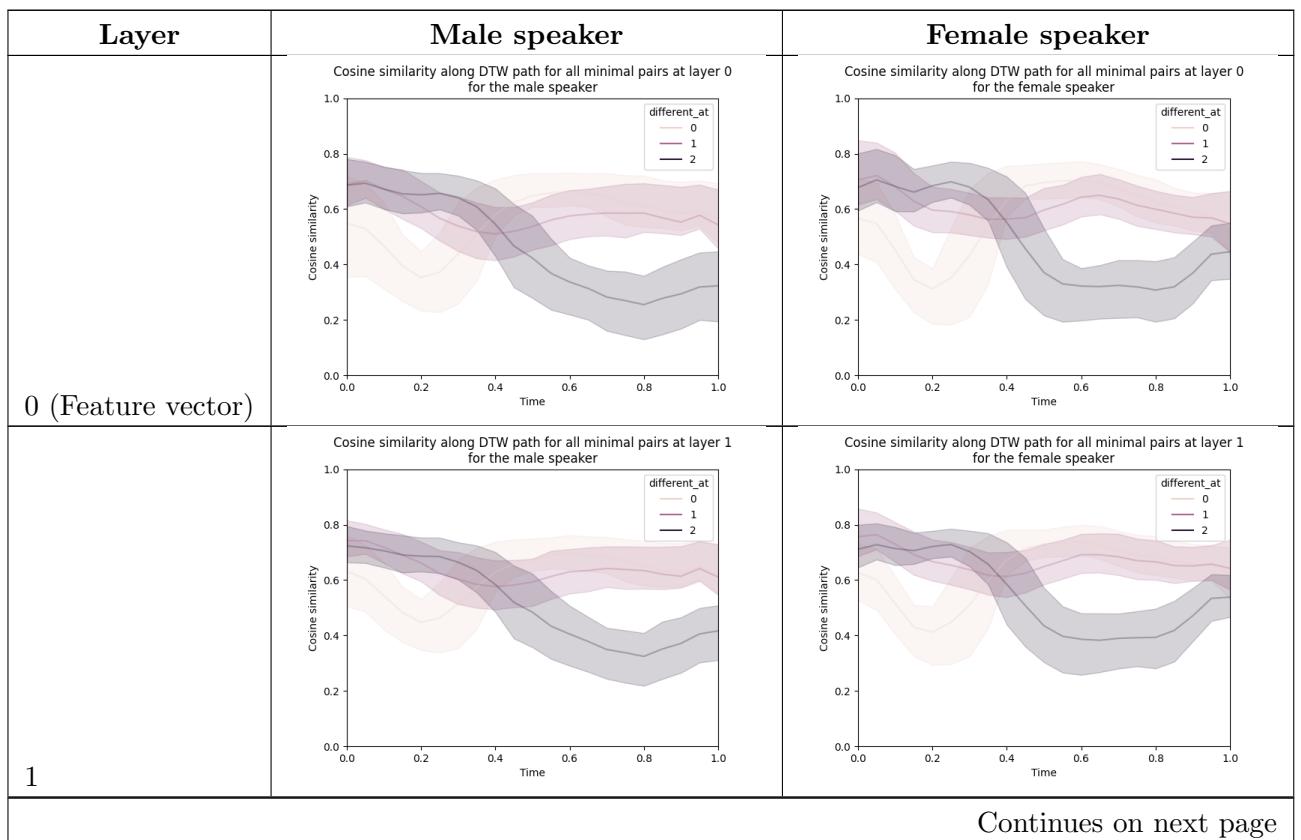


Table B.4 – continued from previous page

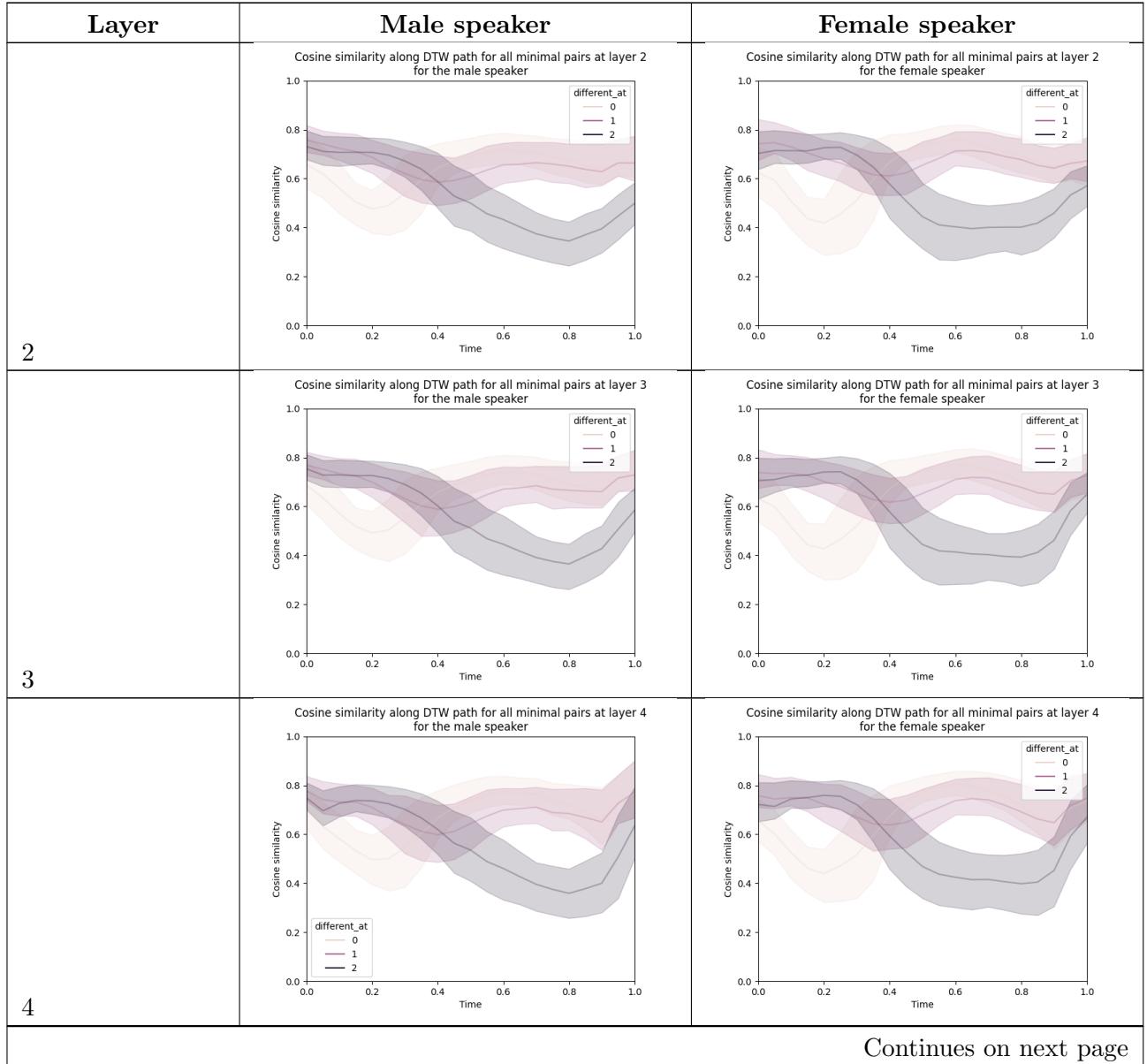


Table B.4 – continued from previous page

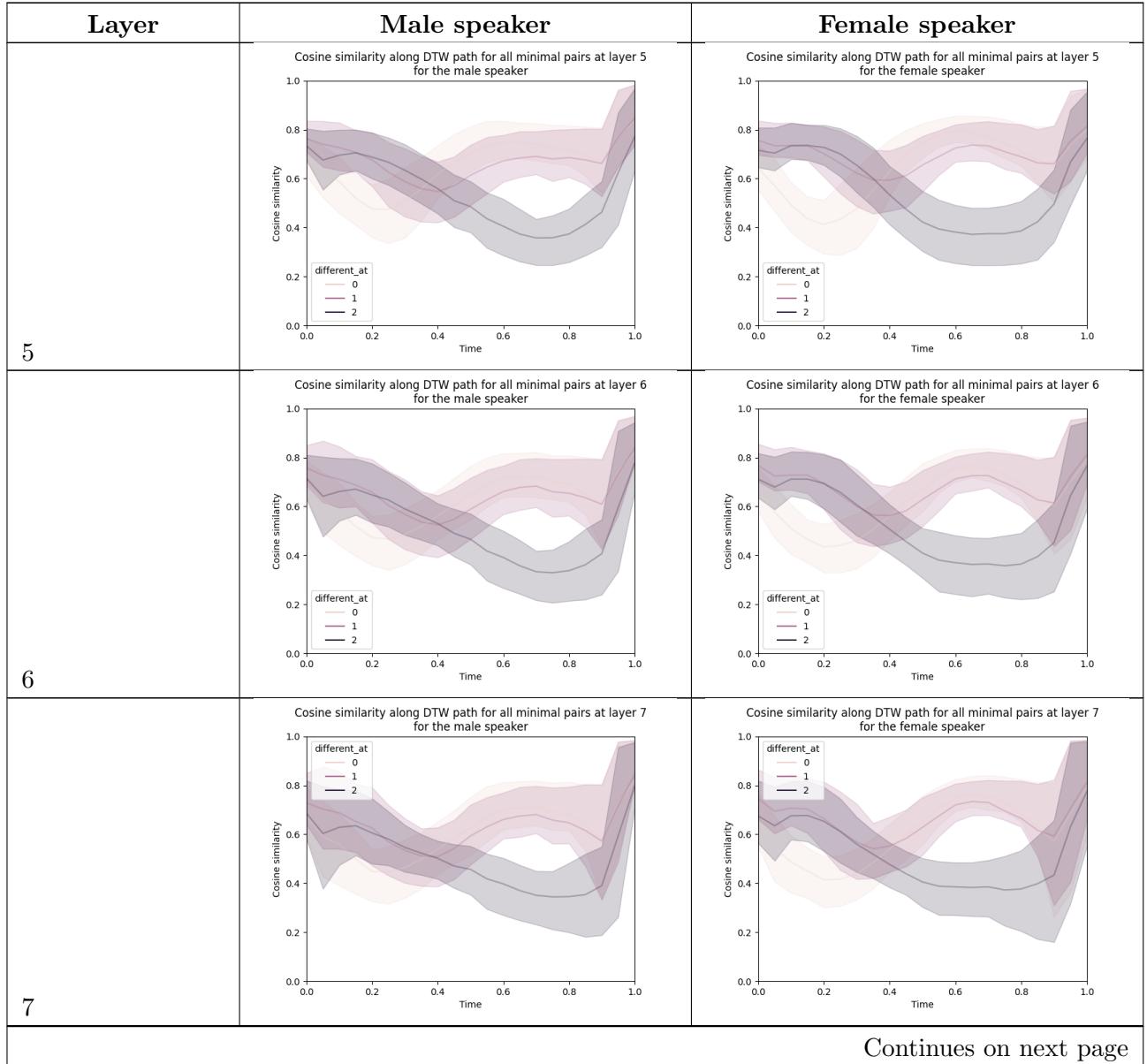


Table B.4 – continued from previous page

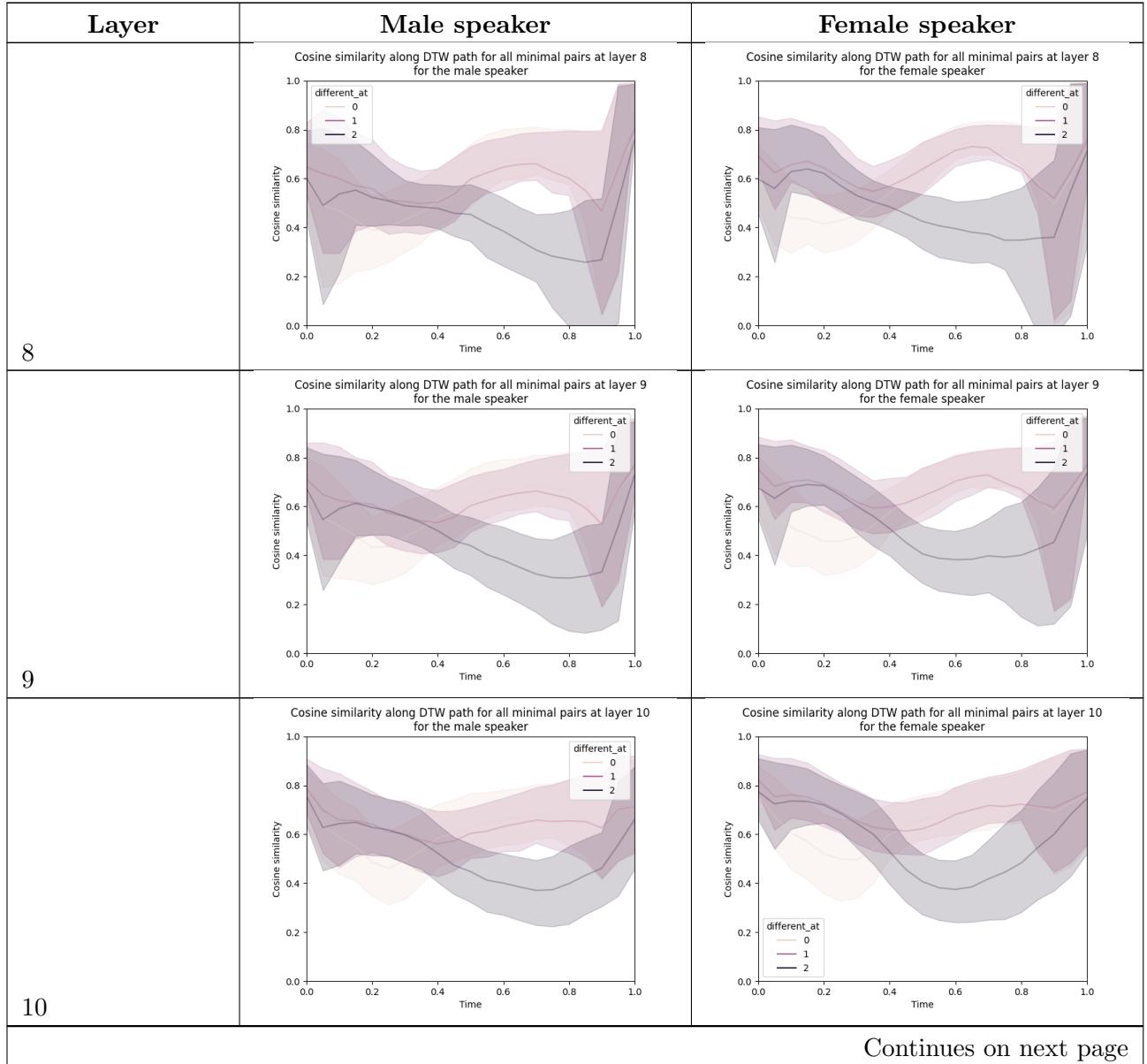
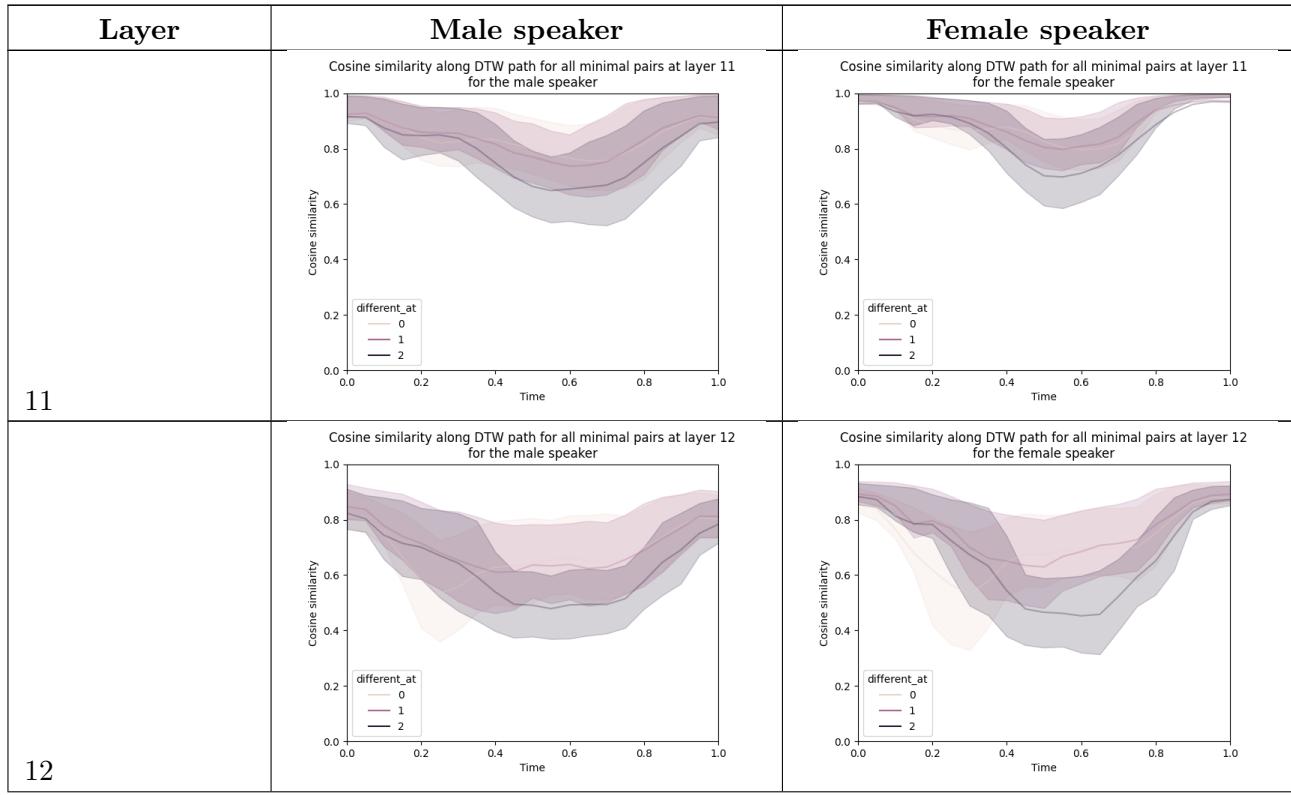


Table B.4 – continued from previous page



Different at Layer	Timestamp of minimum						Minimum similarity						Standard deviation of similarity					
	mean			std			mean			std			mean			std		
	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
0	0.35	0.60	0.78	0.22	0.25	0.13	0.22	0.38	0.12	0.15	0.13	0.18	0.14	0.24	0.04	0.05	0.05	
1	0.32	0.57	0.77	0.21	0.25	0.12	0.26	0.42	0.16	0.11	0.14	0.12	0.18	0.13	0.24	0.04	0.04	0.05
2	0.27	0.52	0.74	0.18	0.24	0.12	0.27	0.44	0.21	0.11	0.12	0.11	0.18	0.13	0.22	0.04	0.04	0.04
3	0.25	0.49	0.74	0.16	0.24	0.12	0.27	0.44	0.23	0.10	0.12	0.11	0.19	0.13	0.22	0.04	0.03	0.04
4	0.25	0.47	0.76	0.18	0.27	0.16	0.28	0.45	0.23	0.10	0.12	0.12	0.20	0.13	0.22	0.04	0.03	0.05
5	0.24	0.38	0.65	0.20	0.30	0.26	0.24	0.37	0.22	0.09	0.12	0.11	0.22	0.16	0.22	0.04	0.03	0.04
6	0.30	0.43	0.64	0.31	0.35	0.31	0.23	0.32	0.20	0.09	0.13	0.11	0.21	0.17	0.22	0.04	0.03	0.05
7	0.34	0.46	0.65	0.34	0.37	0.35	0.19	0.27	0.17	0.10	0.15	0.13	0.22	0.18	0.21	0.04	0.04	0.05
8	0.40	0.47	0.67	0.39	0.38	0.36	0.04	0.11	-0.00	0.16	0.22	0.16	0.24	0.21	0.25	0.05	0.06	0.06
9	0.45	0.52	0.74	0.38	0.38	0.29	0.17	0.23	0.10	0.10	0.16	0.12	0.22	0.19	0.25	0.04	0.04	0.05
10	0.35	0.53	0.72	0.26	0.33	0.23	0.24	0.34	0.16	0.11	0.13	0.13	0.21	0.17	0.25	0.04	0.04	0.05
11	0.54	0.57	0.59	0.20	0.17	0.15	0.63	0.61	0.48	0.18	0.20	0.21	0.11	0.12	0.17	0.06	0.06	0.08
12	0.34	0.51	0.58	0.15	0.18	0.16	0.12	0.24	0.12	0.14	0.17	0.15	0.24	0.20	0.26	0.04	0.05	0.04

Table B.1: Accompanying section 4.1, this table shows the timestamp of the minimum cosine similarity, the minimum cosine similarity, and the standard deviation of the cosine similarity at each layer. This data is from the recordings by the male speaker, and with carrier phrase. For each layer, the table shows for each character position (0, 1, or 2) the mean and the standard deviation.

Different at Layer	0	Timestamp of minimum						Minimum similarity						Standard deviation of similarity					
		mean			std			mean			std			mean			std		
		0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
0	0.26	0.54	0.69	0.21	0.33	0.16	0.20	0.39	0.16	0.12	0.11	0.18	0.11	0.20	0.04	0.03	0.03	0.04	
1	0.25	0.51	0.65	0.21	0.29	0.14	0.32	0.47	0.26	0.11	0.09	0.11	0.15	0.10	0.17	0.04	0.02	0.04	
2	0.25	0.50	0.64	0.22	0.29	0.16	0.32	0.46	0.26	0.12	0.10	0.12	0.15	0.10	0.17	0.04	0.02	0.04	
3	0.27	0.52	0.66	0.23	0.28	0.17	0.32	0.45	0.26	0.12	0.11	0.12	0.15	0.10	0.18	0.04	0.03	0.04	
4	0.36	0.59	0.70	0.29	0.30	0.20	0.29	0.40	0.24	0.11	0.15	0.12	0.17	0.12	0.18	0.04	0.03	0.04	
5	0.41	0.58	0.68	0.31	0.30	0.18	0.27	0.35	0.23	0.10	0.13	0.11	0.18	0.15	0.19	0.04	0.04	0.04	
6	0.54	0.65	0.71	0.34	0.29	0.20	0.24	0.28	0.18	0.11	0.15	0.11	0.18	0.16	0.20	0.04	0.04	0.04	
7	0.61	0.68	0.73	0.34	0.28	0.23	0.17	0.20	0.13	0.12	0.14	0.10	0.21	0.19	0.21	0.04	0.04	0.04	
8	0.68	0.71	0.74	0.32	0.29	0.26	-0.01	0.01	-0.04	0.14	0.16	0.10	0.25	0.23	0.25	0.05	0.05	0.05	
9	0.66	0.71	0.73	0.31	0.27	0.23	0.12	0.14	0.08	0.11	0.13	0.09	0.22	0.20	0.23	0.04	0.04	0.05	
10	0.41	0.60	0.62	0.27	0.28	0.20	0.24	0.30	0.21	0.11	0.11	0.18	0.17	0.21	0.04	0.03	0.05		
11	0.48	0.50	0.53	0.20	0.17	0.15	0.64	0.66	0.56	0.18	0.19	0.20	0.10	0.13	0.06	0.06	0.07		
12	0.32	0.42	0.50	0.15	0.16	0.15	0.26	0.17	0.14	0.16	0.14	0.20	0.17	0.22	0.04	0.05	0.05		

Table B.2: Accompanying section 4.1, this table shows the timestamp of the minimum cosine similarity, the minimum cosine similarity, and the standard deviation of the cosine similarity at each layer. This data is from the recordings by the female speaker, and without carrier phrase. For each layer, the table shows for each character position (0, 1, or 2) the mean and the standard deviation.

Different at Layer	0	Timestamp of minimum						Minimum similarity						Standard deviation of similarity					
		mean			std			mean			std			mean			std		
		0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
0	0.30	0.59	0.72	0.18	0.29	0.13	0.21	0.40	0.15	0.12	0.09	0.11	0.19	0.12	0.23	0.05	0.03	0.05	
1	0.28	0.57	0.67	0.17	0.29	0.13	0.24	0.44	0.18	0.12	0.10	0.12	0.19	0.11	0.23	0.05	0.02	0.05	
2	0.24	0.49	0.66	0.14	0.27	0.15	0.25	0.46	0.22	0.12	0.11	0.12	0.20	0.11	0.22	0.05	0.03	0.05	
3	0.23	0.45	0.68	0.13	0.28	0.15	0.25	0.47	0.21	0.11	0.11	0.12	0.20	0.12	0.23	0.05	0.03	0.05	
4	0.22	0.46	0.66	0.20	0.36	0.25	0.24	0.40	0.20	0.10	0.14	0.14	0.22	0.13	0.23	0.05	0.03	0.05	
5	0.31	0.48	0.63	0.31	0.39	0.29	0.21	0.29	0.17	0.09	0.12	0.11	0.23	0.17	0.24	0.04	0.03	0.05	
6	0.42	0.56	0.68	0.39	0.39	0.31	0.18	0.21	0.13	0.08	0.12	0.09	0.23	0.20	0.24	0.04	0.03	0.05	
7	0.56	0.61	0.72	0.39	0.38	0.30	0.13	0.16	0.09	0.08	0.13	0.10	0.24	0.21	0.24	0.04	0.04	0.05	
8	0.64	0.66	0.75	0.38	0.36	0.28	-0.04	-0.02	-0.08	0.11	0.16	0.11	0.27	0.24	0.28	0.05	0.06	0.06	
9	0.58	0.67	0.75	0.39	0.35	0.27	0.12	0.14	0.07	0.08	0.13	0.10	0.24	0.21	0.27	0.04	0.05	0.06	
10	0.39	0.55	0.68	0.29	0.35	0.22	0.25	0.33	0.19	0.10	0.11	0.14	0.21	0.17	0.25	0.04	0.03	0.06	
11	0.53	0.53	0.56	0.16	0.13	0.12	0.77	0.75	0.67	0.12	0.13	0.15	0.06	0.07	0.10	0.03	0.04	0.05	
12	0.32	0.47	0.58	0.13	0.16	0.12	0.14	0.29	0.16	0.13	0.16	0.14	0.22	0.18	0.25	0.04	0.05	0.04	

Table B.3: Accompanying section 4.1, this table shows the timestamp of the minimum cosine similarity, the minimum cosine similarity, and the standard deviation of the cosine similarity at each layer. This data is from the recordings by the female speaker, and with carrier phrase. For each layer, the table shows for each character position (0, 1, or 2) the mean and the standard deviation.