MULTI-VIEW ACOUSTIC WORD EMBEDDINGS WITH TRANSFORMER ENCODERS

Brandon Rhodes[1]*, Rujual Bains*[2]

[1]The University of Chicago, Department of Linguistics and Department of Statistics
[2]The University of Chicago, Department of Statistics

bprhodes@uchicago.edu, bains@uchicago.edu

Abstract

Embeddings have been shown to be an effective pretraining method for neural architectures, and there has been recent work on acoustic word embeddings — fixed-dimensional vector representations of arbitrary-length speech segments corresponding to words. They have been used with success in end-to-end whole word ASR as well as speech retrieval and detection tasks. Multi-view acoustic word embeddings use both acoustic and character (or phoneme, as in our case) data to jointly learn an embedding for each view of the data. The previous networks used in this multi-view setting were bidirectional LSTMs with a contrastive loss; we will explore the use of transformer encoders with a contrastive loss. We study the different effects of model size (number of layers and number of heads) for the phoneme view. Our results do not improve over previous approaches for the task of word discrimination and cross-view word discrimination, but the performance is encouraging for future work.

Index Terms: acoustic word embeddings, transformer, speech recognition, multi-view learning

1. INTRODUCTION

Neural word embeddings have been an important part of natural language processing (NLP) research in recent history [1, 2]. They have not played as central a role in the speech community, but there has been some increased in interest in acoustic word embeddings in the past few years [3, 4, 5, 6]. Acoustic word embeddings are vector representations of fixed-dimension for utterances of arbitrary length. They are of interest because they can be used in whole-word ASR [7, 8], removing the potential ambiguities in sub-word representations, and they can allow for more efficient and more accurate distance computations in a spoken term detection task [9] or query-by-example search [10]. Multi-view acoustic word embeddings are a direct extension of acoustic word embeddings, and were first proposed in [11]. In this work, embeddings for both sequences of acoustic frames and sequences of characters were jointly learned. These embeddings were shown to improve upon previous results on word discrimination tasks.

The work here will be an extension of [11]: we will use a transformer encoders architecture instead of a bidirectional LSTM architecture for both views. Transformers burst onto the scene in [12] in the field of NLP, most notably with BERT in [13]. The speech community has started exploring the use of transformers and self-attention [14, 15, 16], but it is still an area of research where there is much left to be explored. The major contribution of the first transformer [12] was the use of self-attention. With self-attention, the input at each layer can attend to itself. For language modelling this has been shown to be very useful for contextualized word embeddings [13].

In the work here, acoustic frames and characters will attend to each other: instead of the sentence being the complete sequence and words elements in that sequence, we will consider the word the complete sequence and elements as either acoustic frames or characters. Self-attention within the word could be reasonable for analogous reasons that it is in the context of NLP contextualized word embeddings: speech sounds exhibit bidirectional, non-local dependencies. Consider the words *anaconda*, *butter* and *bitwise*: in the first word, we see full and reduced vowels alternating; for the second and third word, we have a different [t] pronunciation depending on the previous and following sounds. Our hypothesis is that explicitly modeling these dependencies will yield useful acoustic word embeddings.

We will be concerned with finding a good transformer architecture for the phoneme view. There has been work with transformers in the context of speech recognition [16, 15], so we will use an architecture for the acoustic view similar to the ones found to be useful there; however, there has not been as much work on character level language modelling with transformers, let alone phoneme level. The most prominent work is from [17], and it required a sufficiently deep architecture, 64 layers. We are interested in exploring the consequences of two parameter choices in the phoneme view: number of layers and number of heads per layer.

2. OUR MULTI-VIEW MODEL

2.1. Transformers

The transformer is a seq2seq encoder-decoder architecture, and it was first introduced in [12]. The defining characteristics of this architecture are that it uses no recurrent structure and it has multi-head attention in both the encoder and decoder networks. Moreover, it only uses a combination of multi-head self-attention and point-wise feed forward neural networks. To obtain position and relative position information, we must inject that information into the input, which we have done with the typical position encodings, described in 3.2.1. We will only use the encoder part in our work, so below we will only describe this component.

2.1.1. Scaled dot-product attention

Self-attention is used to describe an attention mechanism which allows the inputs to attend over each other, and the type of self-attention used here is multi-head scaled dot-product attention. For a given attention head, there is a set of keys, queries and values. The output for a query is a weighted sum of the values, where the weights of each value are determined by some function of the keys and queries. In our case, we will use scaled dot-product attention as in [12], where the weight of each value is determined by the dot-product of a given key and query. Pictori-

ally, this is shown in Figure 1. Mathematically, let $\mathbf{Q} \in \mathbb{R}^{t_q \times d_q}$ be the matrix of queries, $\mathbf{K} \in \mathbb{R}^{t_k \times d_k}$ be the matrix of keys and $\mathbf{V} \in \mathbb{R}^{t_v \times d_v}$ be the matrix of values. The variables t_* and d_* correspond to the amount of frames/phonemes and dimensionality of the keys, queries and values respectively. As is typical, we say $d_q = d_k$ and $t_q = t_k = t_v$. The output of the self-attention are the following, where the scaling factor was found to be effective with managing the gradient:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (1)$$

Multi-head attention with h heads is done by doing this scaled dot-product attention h times, where linear projections $\mathbf{W}_i^Q \in \mathbb{R}^{t_q \times d_q}, \mathbf{W}_i^K \in \mathbb{R}^{t_q \times d_k}, \mathbf{W}_i^V \in \mathbb{R}^{t_q \times d_v}$ for each attention head, $1 \leq i \leq h$, are used to project the keys, queries and values into smaller subspaces before computing the attention. Then, the output of each attention head is concatenated together to get a vector of same dimension as d_{model}. We assume $d_q = d_k = d_v = d_{model}/h$. Note that $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \quad (2)$$

$$\text{where} \quad \text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (3)$$

2.1.2. *Encoder layer*

The encoder layer consists of a linear projection of the input so that they have dimensionality d_{model}, a position encoding layer, which is the typical sinusoidal position encodings discussed in 3.2.1, multi-head self-attention, two rounds of layer normalization and a point-wise feedforward neural network. There are residual connections in both of the sublayers. All of these components work as expected.

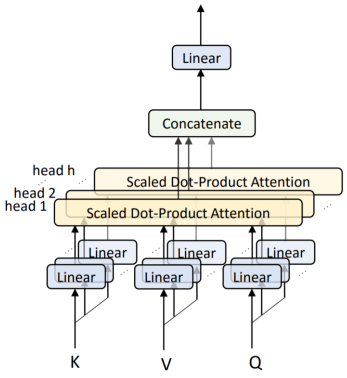

Figure 1: *Scaled dot-product multi-head attention is performed in parallel.*

2.2. Multi-view acoustic word embeddings with transformers

For acoustic word embeddings, we want to learn a fixed-dimension vector representation of an acoustic signal of arbitrary length: in the multi-view setting, we will learn embeddings for the corresponding phoneme sequences of given acoustic words alongside the acoustic embeddings for these words. The multi-view set up in [18] is shown below in Figure 3. This work used a bidirectional LSTM hidden sizes of 512. What we propose is shown in Figure 4. We will simply replace the

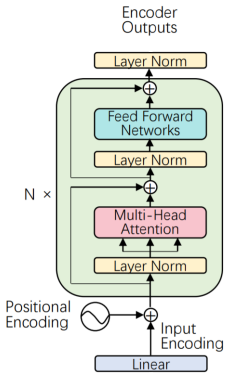

Figure 2: *This is the layer structure used in the transformer encoder. The linear projection layer can be thought of as an embedding matrix for the phoneme view.*

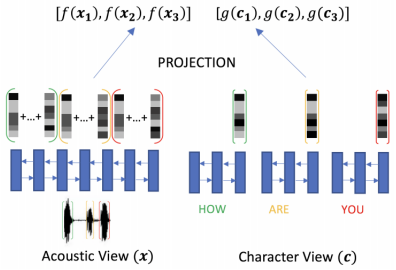

Figure 3: *The multi-view learning setup used in [18]. In this work they the average of the hidden vectors for the sequence of acoustic frames* $\mathbf{x}$ *to get the acoustic embedding* $f(\mathbf{x})$ *and take the final hidden state for the character sequence* $\mathbf{c}$ *for the character embedding* $g(\mathbf{c})$.

bidirectional LSTM with a transformer encoder. In [11], the last hidden state from the acoustic network and the character network were taken as the acoustic and character embeddings, while in [18] the average of the acoustic hidden states (or average of the hidden states' projections) and the final hidden state (or projection of it) of the character sequence were used as the acoustic and character embeddings. This project will take the average in both cases and will use phoneme sequences instead of character sequences.

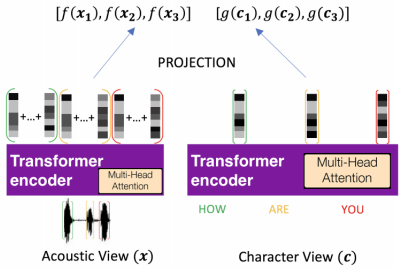

Figure 4: *The multi-view learning setup used in this work. We will average the hidden vectors for the sequence of acoustic frames* $\mathbf{x}$ *to get the acoustic embedding* $f(\mathbf{x})$ *and average the hidden states for the character sequence* $\mathbf{c}$ *for the character embedding* $g(\mathbf{c})$.

The loss used in training the network here is a contrastive loss, originally based on Siamese networks of [19], but more specifically related to the several objectives introduced in [11] and [18]. We aim to find embedding functions f, g which minimize the objective, where N is the number of training pairs $(\mathbf{x}_i, \mathbf{c}_i)$. The notation $char(\mathbf{x})$ represents the character sequence corresponding to the word label of acoustic sequence $\mathbf{x}$.

$$\begin{aligned}\min_{f,g} \sum_{i=1}^{N} &\Big[m + d(f(\mathbf{x}_i), g(\mathbf{c}_i)) - \min_{\mathbf{c}\neq\mathbf{c}_i} d(f(\mathbf{x}_i), g(\mathbf{c}))\Big]_+ \\ &+ \Big[m + d(g(\mathbf{c}_i), f(\mathbf{x}_i)) - \min_{\mathbf{c}\neq\mathbf{c}_i} d(g(\mathbf{c}_i), g(\mathbf{c}))\Big]_+ \\ &+ \Big[m + d(g(\mathbf{c}_i), f(\mathbf{x}_i)) - \min_{char(\mathbf{x})\neq\mathbf{c}_i} d(g(\mathbf{c}_i), f(\mathbf{x}))\Big]_+ \end{aligned} \tag{4}$$

The distance metric used is cosine distance $d(\mathbf{a}, \mathbf{b}) = 1 - \left\langle \frac{\mathbf{a}}{||\mathbf{a}||}, \frac{\mathbf{b}}{||\mathbf{b}||} \right\rangle^2$, and the margin m is a parameter to be tuned by hand.[1] The terms in the loss above correspond to obj^0, obj^1 and obj^2 in [17]. In line with [18], we don't minimize over all $\mathbf{c} \neq \mathbf{c}_i$ and all $char(\mathbf{x}) \neq \mathbf{c}_i$; instead, we select the k most offending examples within the mini-batch and use the mean cosine distance.

3. EXPERIMENTAL SETUP

3.1. Data

We used the 300-hour Switchboard corpus of conversation English speech [20], and we have take the acoustic features of 36-dimensional MFCCs+Δ+$\Delta\Delta$. Following previous work, we use the same train/dev/test split, which gave us 9971/10966/11024 acoustic word and corresponding phoneme sequence pairs for each set.

The phoneme sequence embeddings consisted of the all the phonemes in ARPAbet as well as 3 sounds ([NOISE], [VOCALIZED-NOISE], [LAUGHTER]), making the vocab size 45. Words spelled in digits are spelled out phonetically, and reductions in spoken words are indicated by using $-$ for the boundary of the word and $[\cdot]$ for the actual unspoken part: a reduction of *you're* to *ya'* would be listed in the dictionary as {you['re]- : y-uh}. Word-level alignments for the acoustic data were obtained from a competitive ASR system for this corpus. Acoustic words which were shorter than 2 frames as well as words outside the training vocabulary were omitted from the training, and an utterance was removed if all the words did not qualify to be a part of the training. This follows [18] closely.

3.2. Experiments

For training we used a batch size of 256 acoustic word and corresponding phoneme sequence pairs, which typically led to approximately 210 unique words per batch. We trained on one GPU. The same learning rate schedule is used as in [18]: if the held-out performance doesn't improve after 4 epochs, the learning rate is decayed by a factor of 10 and the model is reset to the previous best. We used the Adam optimizer [21] with an initial learning rate of 0.0005 and with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Training stopped when the learning rate was less than 10^{-7}. We used the PyTorch toolkit [22] for all experiments, and we used the PyTorch class for transformer encoders.

[1] However, in [11] they consider an adaptive margin.

3.2.1. Transformer encoder for acoustic view

The acoustic view model is a transformer encoder based on the previous works [15, 16, 14]. Specifically, we will use the base model found in [15]: the transformer will have $N = 6$ layers, $h = 4$ attention heads per layer, a $d_{ff} = 1024$ feed-forward dimension and $d_{model} = 256$. For the non-linearity we will use RELU. The position encoding used is the sinusoidal encoding, which works as follows for $0 \leq i \leq d_{emb}/2$, where d_{emb} is the dimensionality of the input embedding[2]:

$$\begin{aligned}\text{PE}(t, 2i) &= \sin \frac{t}{5000^{2i/d_{emb}}} \\ \text{PE}(t, 2i+1) &= \cos \frac{t}{5000^{2i/d_{emb}}}\end{aligned} \tag{5}$$

There is a projection layer at the beginning to transform the acoustic feature vectors to have the same dimensionality for the multi-head self-attention mechanism, and there is an optional projection layer at the end to project the embeddings into a vector space of different dimensionality. We do not explore this optional projection in the project. There is a dropout of $p = 0.1$ applied to the sub-layer output before it is added to the sub-layer input and normalized, following [12].

3.2.2. Transformer encoder for the phoneme view

Since transformers are a relatively new architecture, much work still remains to be done; furthermore, there is not a wide body of research on character-level modeling with transformer architectures. The most prominent work [17] trained a very deep architecture, 64 layers, and had to introduce auxiliary losses. We will stick with the more basic paradigm, and explore shallow to moderately sized architectures. We will fix the dimensionality of the multi-head self attention and the number of attention heads to be the same as the acoustic view at $d_{model} = 256$, but we will vary the number of layers N and number of attention heads h per layer. There is an embedding layer at the start of the network and an optional projection layer at the end. We will not explore the optional projection layer. The position encodings for the input vectors for this view are the same sinusoidal ones as above, and the dropout of $p = 0.1$ is the same as well.

3.2.3. Embedding choices for the phoneme view

For the acoustic view, obtaining the embeddings is straightforward: the input is a vector of continuous values, and it makes sense to take the average of the resulting vectors of the encoder transformer. There is more flexibility with the phoneme view. We can do the same as the acoustic view and average output of the transformer encoder, or we can select any one of the vectors as a 'representative' for the word. This makes sense because there is a finite vocabulary of phonemes, and this is not a novel idea: in [13], they use a special ['CLS'] token as a summary for the sentence when doing downstream tasks. Unfortunately, given the minor implementation snags[3] of implementing the analogous idea for our phoneme view embeddings, we decided to use an approximation to this idea by taking the output of the transformer encoder for the first phoneme as a summary of the word. We will present results for both choices below.

[2] This is an important part of the transformer architecture, but for the sake of time we unfortunately did not explore it.

[3] We couldn't quickly figure out a way to do this without either (i) changing the data in Shane's directory or (ii) copying all of the data into our directory and then changing it.

3.2.4. Evaluation on the development set

The embeddings are trained to optimize the contrastive objective above in (4), and the embedding's quality is evaluated using a cross-view word discrimination task applied to the validation set, as in [18, 11]. In this task, the goal is to determine whether a given acoustic word and written word (using ARPAbet) are the same; if the cosine distance between the embeddings of the two views is below some threshold, we consider them the same. We then compute the precision over a range of thresholds to get the average precision (AP), and this is our validation performance measure. For M acoustic words and a vocabulary size of N phonemic words, we compute the average precision over $M \times N$ pairs.

4. RESULTS

The results we will present below are based on models run for 15 epochs. Training until convergence takes about 60-70 epochs, and this is around 4-6 hours of run-time on a single GPU, but the 15 epoch mark was a good indicator of relative performance of the models within this class.

Model	N	acoustic AP	cross-view AP
Small Avg.	1	0.398	0.272
Small Rep.	1	0.079	0.030
Medium Avg.	4	**0.425**	**0.298**
Medium Rep.	4	0.379	0.247
Large Avg.	8	0.003	0.000
Large Rep.*	8	0.003	0.000
Codebase	—	0.705	0.665

Table 1: *Results from testing different transformer encoder architectures. The number of layers are varied and whether or not the average (Avg.) or representative (Rep.) of the final transformer encoder output was taken is shown. (* indicates that this was stopped at 6 epochs because there was no improvement from the first epoch, which was the same as the other 8 layer model.)*

Table 1 shows the results of the experiments with different depths of the model within the 1–8 range. We see that the medium-sized models performed better on average than the others. The small model only achieves reasonable performance when the final output for the phoneme transformer encoder is averaged, and the larger models do poorly all together. The models where the final output is averaged for the phoneme view perform better than those where a representative is taken. We even see that taking the average with the smaller model is better than taking a representative with a medium-sized model. Maybe this would change if we had a special ['CLS'] token added to the vocabulary, but that will have to be explored in future work.

Working in a semi-greedy fashion, we then further tuned by doing a small search of the optimum number of heads per layer in the phoneme view, after fixing the number of layers to be $N = 4$. The results are shown in Table 2. We see that around 16 attention heads per layer is best; increasing it to 32 heads gives a slight improvement on cross-view AP, but it decreases the acoustic AP by 3 points.

We then chose a model with a phoneme view of $N = 4$ layers and $h = 16$ heads to be the one we evaluate on the test set. Both models had $d_{model} = 256$ and $d_{ff} = 1028$. The acoustic model had $N = 6$ layers and $h = 4$ heads, as before in the experiments. We see in Table 3 and Table 4 that the model performs worse than the codebase provided by 9 and 20 points respectively.

h	acoustic AP	cross-view AP
1	0.373	0.294
4	0.425	0.298
8	0.429	0.317
16	**0.447**	0.318
32	0.413	**0.320**

Table 2: *Results from taking a medium-sized model ($N = 4$ layers) and trying out different values for the amount of attention heads h per layer. We used averaging of the final output from the transformer encoder to get the phoneme view embeddings.*

Method	**Test AP** (acoustic view)
MFCCs + DTW (Kamper et al., 2016)	0.214
Corsp. AE + DTW (Kamper et al., 2015)	0.469
Phone post. + DTW (Carlin et al., 2011)	0.497
Siamese CNN (Kamper et al., 2016)	0.549
Siamese LSTM (Settle & Livescu, 2016)	0.671
Multi-view LSTM (He et al., 2016)	**0.806**
Codebase LSTM (Settle et al., 2020)	0.790
Our multi-view Transformer encoder	0.708

Table 3: *Average precision on the test set for previous approaches and ours.*

Method	**Test AP** (cross-view)
Multi-view LSTM (He et al., 2016)	**0.892**
Codebase LSTM (Settle et al., 2020)	0.750
Our multi-view Transformer encoder	0.551

Table 4: *Cross-view average precision on the test set for the previous work on multi-view embeddings and ours.*

5. Conclusions

In this project, we investigated the prospect of creating acoustically grounded word embeddings using a transformer encoder architecture. We found that using a transformer encoder architecture did not improve upon the results using a bidirectional LSTM, but we were not able to do an extensive search of the parameter space, so it is inconclusive whether or not this architecture has the potential to produce more competitive embeddings than a recurrence based architecture. We focused solely on tuning the phoneme view transformer, and it showed that a medium-size architecture performed best. Going beyond 8 layers in this view would seem to require more machinery, which is consistent with what is said in [17]. Future directions for this work would be to investigate the type of position encoding and to explore what the attention heads are actually doing, especially in the phoneme view.

6. Acknowledgements

Thank you very much to Karen, Ankita and Shane for their help and resources for this project; it was a lot of fun.

7. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5818–5822, 2015.

[4] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," *CoRR*, vol. abs/1510.01032, 2015. [Online]. Available: http://arxiv.org/abs/1510.01032

[5] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 410–415.

[6] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," *CoRR*, vol. abs/1611.02550, 2016. [Online]. Available: http://arxiv.org/abs/1611.02550

[7] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, p. 2067–2080, Sep 2011. [Online]. Available: http://dx.doi.org/10.1109/TASL.2011.2112350

[8] S. Bengio and G. Heigold, "Word embeddings for speech recognition," 2014.

[9] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. sigir*, vol. 7, 2007, pp. 51–57.

[10] I. Szöke, L. J. Rodriguez-Fuentes, A. Buzo, X. Anguera, F. Metze, J. Proenca, M. Lojka, and X. Xiong, "Query by example search on speech at mediaeval 2015." in *MediaEval*, 2015.

[11] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *CoRR*, vol. abs/1611.04496, 2016. [Online]. Available: http://arxiv.org/abs/1611.04496

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[13] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[14] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7115–7119.

[15] L. Dong, S. Xu, and B. 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)*. IEEE, 2018, pp. 5884–5888.

[16] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. 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)*. IEEE, 2020, pp. 6874–6878.

[17] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3159–3166.

[18] S. Settle, K. Audhkhasi, K. Livescu, and M. Picheny, "Acoustically grounded word embeddings for improved acoustics-to-word speech recognition," *CoRR*, vol. abs/1903.12306, 2019. [Online]. Available: http://arxiv.org/abs/1903.12306

[19] J. BROMLEY, J. W. BENTZ, L. BOTTOU, I. GUYON, Y. LECUN, C. MOORE, E. SÄCKINGER, and R. SHAH, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 07, no. 04, p. 669–688, Aug 1993. [Online]. Available: http://dx.doi.org/10.1142/S0218001493000339

[20] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ser. ICASSP'92. USA: IEEE Computer Society, 1992, p. 517–520.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.