A Time Delay Neural Network with Shared Weight Self-Attention for Small-Footprint Keyword Spotting

Ye Bai [1,2]*, *Jiangyan Yi*[1]*, *Jianhua Tao*[1,2,3]*, *Zhengqi Wen*[1],
Zhengkun Tian[1,2], *Chenghao Zhao*[4], *Cunhang Fan*[1,2]

[1]NLPR, Institute of Automation, Chinese Academy of Sciences, China,
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, China,
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, China,
[4]Jiangsu Normal University, China.

{ye.bai, jiangyan.yi, jhtao}@nlpr.ia.ac.cn

Abstract

Keyword spotting requires a small memory footprint to run on mobile devices. However, previous works still use several hundred thousand parameters to achieve good performance. To address this issue, we propose a time delay neural network with shared weight self-attention for small-footprint keyword spotting. By sharing weights, the parameters of self-attention are reduced but without performance reduction. The publicly available Google Speech Commands dataset is used to evaluate the models. The number of parameters (12K) of our model is $1/20$ of state-of-the-art ResNet model (239K). The proposed model achieves an error rate of 4.19% , which is comparable to the ResNet model.

Index Terms: keyword spotting, small-footprint, tdnn, shared weight self-attention

1. Introduction

Keyword spotting (KWS) is a kind of speech technology for users to control intelligent devices with voice, such as mobile phones, tablets, and "smart home" devices. Because it commonly runs on mobile devices, a small memory footprint and efficient computation are required.

Two mainstream approaches for KWS are large vocabulary continuous speech recognition (LVCSR) based methods [1] and keyword/filler hidden Markov models (HMMs) based methods [2]. However, these two approaches cost a big memory footprint and require high computation. So these approaches are limited in on-device applications.

Deep KWS [3] considers keyword spotting as an audio classification problem, where each keyword is denoted as a class. An additional "filler" class is defined for all other words. A deep neural network processes acoustic features and outputs the posteriors of the keywords. When the confident score exceeds a threshold, the keyword is detected. This approach shows the small footprint, low computational cost, low latency, high performance, and draws much attention recently [4, 5, 6]. However, previous work still use several hundred thousand parameters to achieve state-of-the-art performance. We believe that the number of parameters can still be reduced and the performance will not be hurt.

ResNet based KWS systems, which leverage residual connections to improve network depth and small size convolution filters to reduce model size, have achieved good performance [6]. However, because of the number of hidden layers and filters, their best model still has more than 200K parameters. A stacked time delay neural network (TDNN) based model with transfer learning was proposed [5]. However, the stacked network architecture makes the model size large.

The attention mechanism weights inputs to achieve a high-level representation and obtains success in many tasks [7, 8, 9]. It has also been successful when used in KWS [10, 11]. However, recurrent structures in these networks are hard to be implemented in parallel, so that the computation speed is slow. Self-attention mechanism, which captures relations of different positions by pairwise similarities, has shown promising performance in machine translation [12] and ASR [13, 14]. The feed-forward structure of self-attention makes it fast to compute.

In this paper, we propose to use a TDNN with a Shared Weight Self-Attention (SWSA) module for keyword spotting. The TDNN captures local features of a sequence, and the self-attention module captures global features. For self-attention, we propose the SWSA module to reduce parameters. The original self-attention utilizes three weight matrices to project features into different spaces. Instead of the three weight matrices, the proposed SWSA uses a shared weight matrix to project the input into the same space, so the parameters are reduced. We demonstrate that our proposed method is effective for KWS with limited parameters. The experiments are conducted on the publicly available Speech Commands V1 dataset. Compared with state-of-the-art ResNet model, our proposed model achieves a very close classification error rate (4.19% vs. 4.12%) and a much smaller model size (12K vs. 239K).

The rest of this paper is organized as follows. Section 2 describes the proposed models. Section 3 introduces the experimental setup and results. Section 4 concludes the paper.

2. Proposed methods

2.1. Architecture

Fig. 1(a) shows the complete architecture of the model. It consists of six layers. The input of the network is a T-by-D matrix, where T is the length of a feature sequence, and D is the dimensionality of a feature. TDNN based subsampling, which is referred to as TDNN-SUB at layer 1, is used to reduce the length of the sequence. The proposed SWSA is set at layer 2. Layer 3 and layer 4 are TDNNs. The high-level representation is obtained by global mean pooling at layer 5. The top layer is a softmax classifier to output posteriors of each keyword. For example, when the number of keywords is 10, then the number of output nodes is 11, including a filler label representing any non-keyword.

Different from other attention based KWS work [10, 11],

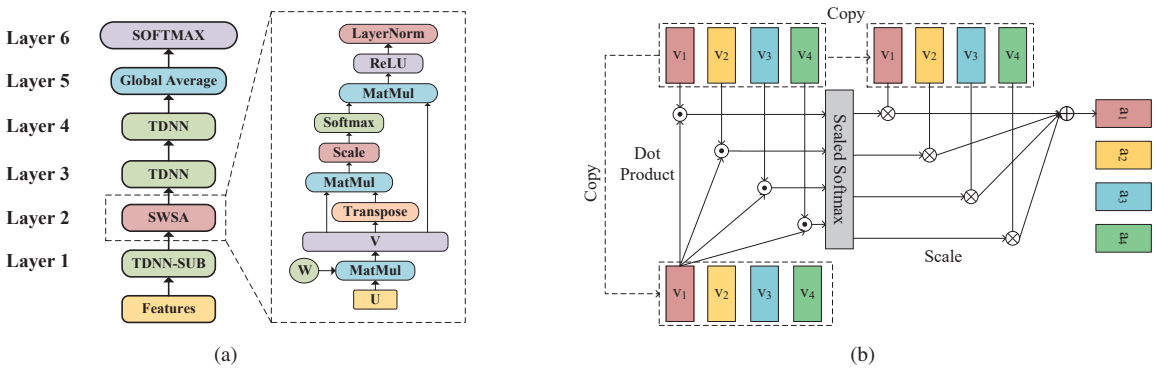

Figure 1: *(a) is a schematic of the architecture. TDNN-SUB means TDNN based subsampling. Shared Weight Self-Attention (SWSA) module is shown in the dashed box. (b) shows the scaled dot self-attention. v_i stands for the i-th vector of the SWSA input, and a_i stands for attended representation at the corresponding location i. First, the dot products of v_i and each vector is computed. Then, the corresponding attention scores are computed with a softmax function and scaling. At last, the vectors of the input, which are weighted by attention scores, are fused into the representation a_i. Each vector of the output is generated by the above procedure. The sizes of the input and the output are the same.*

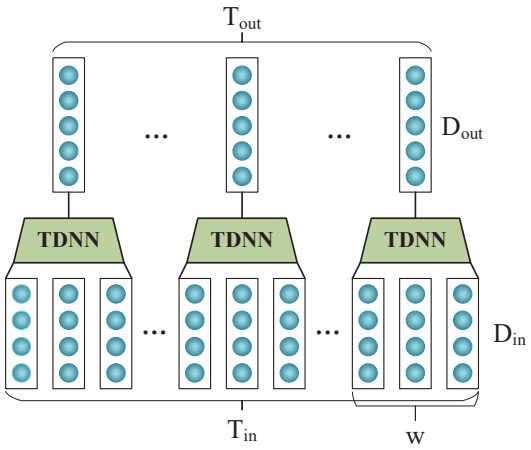

Figure 2: *A schematic of the TDNN based subsampling. T_{in} and D_{in} are length and dimensionality of the input. T_{out} and D_{out} are the length and dimensionality of the output. After subsampling, the length is reduced to $\lceil (T_{in} - w + 1)/k \rceil$. w is the length of TDNN window.*

we do not use recurrent structure in our model. The SWSA module is a feed-forward structure, and the computation is much smaller than RNNs based methods. In addition, the model can be implemented in parallel due to the feed-forward structure. Moreover, a shared weight matrix replaces three different matrices corresponding to queries, keys, and values, so the number of parameters is further reduced.

TDNN is a classic network architecture, and has achieved success in recent speech recognition work [15]. A TDNN can be considered as a DNN moving along time. At each step, w contiguous features are spliced and inputted into a TDNN, and the TDNN outputs one vector. Then, the TDNN moves k steps. Commonly, k is set to 1, i.e. moving step by step, to achieve representation at each location. A TDNN is time invariant and suitable to process speech [16].

TDNN based subsampling, which is shown Fig. 2, is utilized to reduce the length of a sequence of representations. Specifically, k is chosen to be an integer which is larger than 1 but smaller than $w + 1$. When the input is a T_{in}-by-D_{in} matrix, the output is a T_{out}-by-D_{out} matrix, where T_{in} is the length of the input, D_{in} is the dimensionality of the input, $T_{out} = \lceil (T_{in} - w + 1)/k \rceil$ is the length of the output, D_{out} is the dimensionality of the output. This mechanism is used at layer 1.

2.2. Shared weight self-attention

Inspired by success of self-attention mechanism in many tasks [12, 13, 14], we propose a Shared Weight Self-Attention (SWSA) module to improve the performance and reduce the computation. A schematic of SWSA is shown in the dashed box of Fig. 1(a). First, the shared weight self-attention is used to capture long-term dependencies. After self-attention, ReLUs with layer normalization [17] is used as activations.

The original self-attention mechanism can be formulated as follows:

$$\text{Attend}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}})\mathbf{V}, \quad \text{where } \mathbf{Q} = \mathbf{U}\mathbf{W}_q,\ \mathbf{K} = \mathbf{U}\mathbf{W}_k,\ \mathbf{V} = \mathbf{U}\mathbf{W}_v. \tag{1}$$

$\mathbf{Q} \in \mathbb{R}^{T_u \times D_k}$, $\mathbf{K} \in \mathbb{R}^{T_u \times D_k}$, $\mathbf{V} \in \mathbb{R}^{T_u \times D_v}$ denote a query matrix, a key matrix, and a value matrix, respectively, and $\mathbf{U} \in \mathbb{R}^{T_u \times D_u}$ is the input matrix of the attention module. T_u is the length of the input matrix $\mathbf{U}$, D_k is the dimensionality of queries and keys, D_v is the dimensionality of values. Weight matrices $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are leveraged to project $\mathbf{U}$ into different spaces.

The above self-attention mechanism shows promising performance. However, three different projection matrices $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ have many parameters. Because the core operation of the self-attention mechanism is the inner product, we consider the three matrices, i.e. the queries, the keys, and the values, should be projected in the same space. Therefore, we propose a shared weight self-attention (SWSA) module:

$$\text{Attend}(\mathbf{V}) = \text{softmax}(\frac{\mathbf{V}\mathbf{V}^T}{\sqrt{D}})\mathbf{V}, \quad \text{where } \mathbf{V} = \mathbf{U}\mathbf{W}. \tag{2}$$

$\mathbf{W} \in \mathbb{R}^{D_u \times D}$ is the shared weight matrix to project inputs into values. Then the attention is leveraged on the values. Instead of

queries, keys, and values, only a value matrix is used in SWSA. So the size of the weight matrix is much smaller than original attention in Eq. 1.

Fig. 1(b) shows an illustration of shared weight self-attention. First, similarities between a vector (v_1 in Fig. 1(b)) and all other vectors in the input sequence of SWSA are computed by dot product. The attention scores are computed in terms of the similarities after scaling and softmax. Then, the attention representation (a_1 in Fig. 1(b)) corresponding to the vector is generated by weighted sum. Each attention representation in the output sequence of SWSA is computed by this procedure. The lengths of the input and the output of SWSA are the same.

The module can be extended to a multi-head version [12], which is denoted as follows:

$$\begin{aligned} &\text{MultiHead}(\boldsymbol{V}) = \text{Cat}(\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n), \\ &\boldsymbol{h}_i = \text{Attend}(\boldsymbol{U}\boldsymbol{W}_i), \end{aligned} \tag{3}$$

where n is the number of the heads, $\mathbf{W}_i \in \mathbb{R}^{D_u \times (D/n)}$ is the weight matrix of each head, D/n is an integer. So the dimensionalities of the input and the output are the same, and the number of heads does not affect the number of parameters of SWSA. After attention, the heads are concatenated together.

3. Experiments

3.1. Datasets

An English dataset Google Speech Commands (GSC)[1] [18], which is designed for device controlling tasks, is used to evaluate our proposed methods. We use the same version with other work (V1). The dataset consists of 64752 recordings of 30 words. The length of each recording is one second, and each recoding has one word. 10 words are used as keywords and the others are used as fillers. The 10 keywords and the 20 fillers are listed in Table 1. All the fillers are labeled as "_unknown_". We use the standard configuration of the dataset. Specifically, 6835 utterances are used as testing set, 6798 utterances are used as validation set, and 51088 utterances are used as training set.

3.2. Experimental setup

We use 40-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) extracted every 10ms with 25ms of frame length as acoustic features. The 99 contiguous frames are spliced together and inputted to the system.

We refer to the proposed model as `tdnn-swsa` in the rest of the paper. The configuration of `tdnn-swsa` is shown in Table 2. Batch normalization [19] is added after each TDNN layer, while the number of parameters of BN is not listed in Table 2. The number of heads of SWSA is 4. The bottom row is the total number of the parameters of the whole model (including normalization parameters).

All models in the experiments are trained with the same procedure. The weights are initialized with Xavier initialization [20]. The Adam algorithm [21] is used as an optimizer in training procedure. The initial learning rate is set at 0.001. After each epoch, the network is evaluated on the validation set. If no significant improvement (10%) of average cross entropy on validation set is observed, learning rate is halved. The total number of training epochs is 13. The mini-batch size is 32. We use an early stop strategy. Specifically, we store the model which achieves the best accuracy on validation set in the training procedure as the final model.

[1] http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

Table 1: *Word labels in Speech Commands dataset.*

Keywords	"down", "go", "left", "no", "off", "on", "right", "stop", "up", "yes"
Fillers	"bed", "bird", "cat","dog", "happy", "house", "marvin", "sheila", "tree", "wow", "zero", "one", "two", "three", "four", "five","six", "seven", "eight", "nine"

Table 2: *The configuration of* `tdnn-swsa`. *w and k are the window length and the number of moving steps of a TDNN, respectively. The size of output matrix of each layer is $l \times d$, where l is the length of sequence, and d is the dimensionality. #Para is the number of parameters of the weight matrix. #Mult. is the number of multipliers in matrix multiplication.*

Layer	w	k	d	l	#Para.	#Mult.
INPUT	-	-	40	99	-	-
TDNN-SUB	3	3	32	33	3840	126720
SWSA	-	-	32	33	1056	72768
TDNN	3	1	32	33	3072	101376
TDNN	3	1	32	33	3072	101376
Global Pooling	-	-	32	-	-	-
SOFTMAX	-	-	-	-	352	352
Total					11755	402592

We first followed [6] to use classification error rate as a measure to show the performance in number:

$$error = 1 - \frac{\sum_{i=1}^{N} I(\hat{y}_i, y_i)}{N},$$

where $I(\cdot, \cdot)$ is 1 if the variables are the equal, and 0 otherwise, $\hat{y}_i$ is predicted label, y_i is ground truth label, N is the total number of the samples. Each experiment in this section was tested five times. And we evaluated the results with 95% confidence intervals. Specifically, we computed mean error rate of the five experiments, and computed confidence intervals by

$$1.96 \times \frac{\sigma}{5},$$

where σ is the standard deviation. We further draw modified receiver operating characteristic curves to show the performance, which follows [3] and [6].

3.3. Experimental results

3.3.1. The effectiveness of shared weight self-attention

We investigate the effectiveness of the proposed SWSA module. The results are shown in Table 3. `tdnn` is a pure TDNN model, i.e. the SWSA at layer 2 is replaced by a TDNN. To limit `tdnn`'s number of parameters to 12K, the window size and the moving steps of the TDNN-SUB module are set to 4 and 2, respectively, and the window sizes of TDNN modules are all set to 2. `tdnn-blstm` is built by replacing `tdnn-swsa`'s layer 2 with a 32 cells of bidirectional LSTM layer. `swsa` is a model whose layer 2 to 4 are SWSA. We also replace the SWSA in `tdnn-swsa` by a four-head original self-attention module to build `tdnn-sa`. The three projection matrices in self-attention of `tdnn-sa` are all 32-by-8.

Table 3: *The effectiveness of SWSA.*

Model	Error Rate	#Para.
tdnn	$5.62\% \pm 0.341$	12K
tdnn-blstm	$5.79\% \pm 0.189$	20K
swsa	$9.81\% \pm 0.203$	8K
tdnn-sa	$4.24\% \pm 0.149$	16K
tdnn-swsa	$\mathbf{4.19}\% \pm 0.191$	12K

Table 4: *The impact of location of SWSA. The numbers of parameters of three models are the same.*

Model	Error Rate
tdnn-swsa-l3	$4.32\% \pm 0.255$
tdnn-swsa-l4	$4.74\% \pm 0.259$
tdnn-swsa	$\mathbf{4.19}\% \pm 0.191$

We find that combining TDNN and SWSA can achieve the largest gain of performance. swsa has the fewest parameters. Compared with tdnn, tdnn-swsa achieves a 25.44% of relative improvement. Because of the recurrent structure of tdnn-blstm, its computation cost is larger than the other three. The performance of tdnn-blstm is similar to tdnn.

The performances of tdnn-sa and tdnn-swsa are very close. However, the number of parameters of self-attention in tdnn-sa is about three times larger than SWSA. We analysis that the shared weight version of self-attention has enough representation ability in the KWS task.

3.3.2. *The impact of location of SWSA*

We investigate the impact of the location of the proposed SWSA. tdnn-swsa-l3 means that the SWSA is set at layer 3, and tdnn-swsa-l4 means that the SWSA is set at layer 4. Note that the location of the SWSA does not change the number of parameters of the whole model. Table 4 shows the impact of the SWSA location. tdnn-swsa whose SWSA is at lower layer achieves the best performance. However, the difference between the three models is very small.

3.3.3. *A comparison with other models*

We show the comparison with other models in Table 5. Because the results in [6] are not conducted on standard testing set of GSC, we reimplement res15 (which is state-of-the-art) and res8-narrow (with the smallest model size), and achieve comparable accuracy. We use the tensorflow officially implemented cnn-trad-fpool3[2] [4] as another baseline. Because the input of this version of cnn-trad-fpool3 is a whole word utterance rather than a small window, the parameters of cnn-trad-fpool3 are more than original version in [4]. stacked-tdnn is referred to the model which stacks a phone network and a word network together in [5].

From Table 5, we can see that tdnn-swsa achieves 4.19% of error rate with 12K parameters, which is 1/20 of res15. Compared with res8-narrow, tdnn-swsa achieves a relative improvement of 60.80%. tdnn-swsa obtains smaller error rate than stacked-tdnn which uses a complex transfer learning training procedure.

We draw modified receiver operating characteristic (ROC) curves to show the performances the models. All the curves are drawn with the best trial of the model. The lower the better. In Fig. 3, we can see that the performance of tdnn-swsa is similar to res15, and better than res8-narrow and cnn-trad-fpool3.

[2]https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

Table 5: *A comparison with other models.*

Model	Error Rate	#Para.
ResNet15 * [6]	4.2%	238K
stacked-tdnn * [5]	5.7%	251K
cnn-trad-fpool3 †	$7.82\% \pm 0.373$	878K
res15 †	$4.12\% \pm 0.232$	239K
res8-narrow †	$10.69\% \pm 0.867$	20K
tdnn-swsa (ours)	$4.19\% \pm 0.191$	**12K**

* is from the literature.
† is reimplemented.

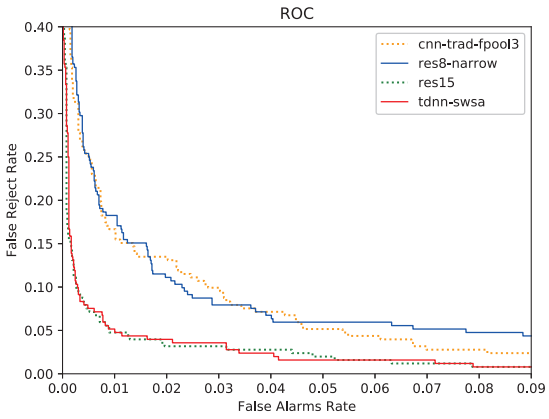

Figure 3: *ROC curves for different models.*

4. Conclusions

In this paper, we introduce a TDNN with SWSA for small-footprint KWS. The SWSA is a self-attention mechanism which uses a shared weight matrix to project input into the same space. A TDNN subsampling is also used to reduce the length of a sequence. We evaluate our model on a public available Google Speech Commands dataset. Compared with previous state-of-the-art model, the proposed model achieves a 4.19% of error rate, which is very close to previous state-of-the-art ResNet model (4.12%). The number of parameters of our model is 1/20 of the ResNet model. In the future, we will investigate robustness of the models in noisy environments.

5. Acknowledgments

This work is supported by the National Key Research & Development Plan of China (No.2017YFC0820602) and the National Natural Science Foundation of China (NSFC) (No.61425017, No.61831022, No.61773379, No.61771472), and Inria-CAS Joint Research Project (No.173211KYSB20170061).

6. References

[1] Y. Bai, J. Yi, H. Ni, Z. Wen, B. Liu, Y. Li, and J. Tao, "End-to-end keywords spotting based on connectionist temporal classification

for mandarin," in *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2016, pp. 1–5.

[2] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting." in *INTERSPEECH*, 2016, pp. 1878–1882.

[3] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 4087–4091.

[4] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech 2015*, 2015, pp. 1478–1482.

[5] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," in *Proc. Interspeech 2018*, 2018, pp. 1264–1268. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1979

[6] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5484–5488.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, 2015, pp. 2048–2057.

[9] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 4774–4778.

[10] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017, pp. 474–481.

[11] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proc. Interspeech 2018*, 2018, pp. 2037–2041. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1777

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998–6008.

[13] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for asr," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5874–5878.

[14] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," in *Proc. Interspeech 2018*, 2018, pp. 3723–3727. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1910

[15] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech 2015*, 2015, pp. 3214–3218.

[16] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, March 1989.

[17] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[18] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 448–456.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256.

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