TRANSFORMER-BASED ACOUSTIC MODELING FOR HYBRID SPEECH RECOGNITION

Yongqiang Wang[1], *Abdelrahman Mohamed*[1], *Duc Le*[1], *Chunxi Liu*[1], *Alex Xiao*[1], *Jay Mahadeokar* [1,⋆], *Hongzhao Huang* [1,⋆], *Andros Tjandra* [2,⋆†], *Xiaohui Zhang* [1,⋆], *Frank Zhang* [1,⋆], *Christian Fuegen*[1,⋆], *Geoffrey Zweig*[1,⋆], *Michael L. Seltzer*[1,⋆]

[1]Facebook AI, USA [2]Nara Institute of Science and Technology, Japan

ABSTRACT

We propose and evaluate transformer-based acoustic models (AMs) for hybrid speech recognition. Several modeling choices are discussed in this work, including various positional embedding methods and an iterated loss to enable training deep transformers. We also present a preliminary study of using limited right context in transformer models, which makes it possible for streaming applications. We demonstrate that on the widely used Librispeech benchmark, our transformer-based AM outperforms the best published hybrid result by 19% to 26% relative when the standard n-gram language model (LM) is used. Combined with neural network LM for rescoring, our proposed approach achieves state-of-the-art results on Librispeech. Our findings are also confirmed on a much larger internal dataset.

***Index Terms*—** hybrid speech recognition, acoustic modeling, transformer, recurrent neural networks

1 Introduction

Since the introduction of deep learning in automatic speech recognition (ASR) [1], a variety of neural network architectures for acoustic modeling have been explored [2–6]. Among them, recurrent neural networks (RNNs), especially long short-term memory (LSTM) [7] neural networks, are widely used, either in conventional hybrid systems (e.g., [3, 8]), sequence-to-sequence-based (e.g. [9, 10]) or neural-transducer-based end-to-end systems (e.g. [11]). However, RNNs have several well-known limitations: 1) due to the vanishing or exploding gradient problem discovered in [12], RNNs cannot model long term temporal dependencies well; 2) the recurrence nature of RNNs makes it difficult to process speech signal in parallel. To address these issues, a variety of neural network architectures have been proposed to replace RNNs, including time delay neural networks (TDNN) [5], feed-forward sequential memory networks (FSMN) [6], and convolution neural networks (CNN) [4, 13], while only limited success has been achieved.

Recently, self-attention network [14] has demonstrated promising results in a variety of natural language processing tasks (e.g., [14–16]). Different from RNNs and CNNs, self-attention connects arbitrary pairs of positions in the input sequence directly. To forward (or backward) signals between two positions that are n steps away in the input, it only needs one step to traverse the network, compared with $O(n)$ steps in RNNs and $O(\log n)$ in CNNs. Moreover, computation in self-attention can be easily parallelized. On top of self-attention, the transformer model [14] leverages multi-head attention and interleaves with feed-forward layers. Self-attention and transformer models were also used for ASR, mostly in the sequence-to-sequence architecture [17–19] with notable exceptions of [20,21].

In this work, we propose and evaluate transformer-based acoustic models (AMs) for hybrid ASR. We explore several modeling choices, including methods to encode either absolute or relative positional information into the input of transformer and an iterated loss to enable training deep transformers. Though our focus in this work is to investigate the potential of transformer-based AMs without any constraint, we do explore streamable transformers and present our initial experimental results. We show that our proposed transformer-based AMs can yield significant word error rate (WER) improvement over very strong bi-directional LSTM (BLSTM) baselines, both on the widely-used Librispeech benchmark and our internal dataset. The results we obtained on Librispeech improve over the previous best hybrid WER by 19% to 26% when the standard 4-gram language model (LM) is used; combined with neural LM rescoring, our system achieves state-of-the-art performance on this dataset.

2 Hybrid Architecture

In hybrid ASR [22], an *acoustic encoder* is used to encode an input sequence $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T$ to a sequence of high level embedding vectors $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_T$. These embedding vectors are used to produce a posterior distribution of tied states of hidden Markov model (HMM), such as senone [23] or chenone [24], for each frame. These posterior distributions are then combined with other knowledge sources such as lexicons and LMs to construct a search graph. A decoder is then used to find the best hypothesis. Different neural networks can be used as the encoder: in DNN, TDNN and CNN, $\boldsymbol{z}_t$ is a function of $\boldsymbol{x}_t$ and its fixed number of neighboring frames; in uni-directional RNNs, $\boldsymbol{z}_t$ is a function of $\boldsymbol{x}_1$ to $\boldsymbol{x}_t$, while in bi-directional RNNs, $\boldsymbol{z}_t$ is a function of the entire input sequence.

Though compared with the sequence-to-sequence or neural transducer architecture, the hybrid approach is admittedly less appealing as it is not end-to-end trained, it is still the best performing system for authors' practical problems. It also has the advantage that it can be easily integrated with other knowledge sources (e.g., personalized lexicon) that may not be available during training. In this work, we aim to leverage the transformer to improve hybrid acoustic modeling.

3 Acoustic Modeling Using Transformer

In this section, we first briefly review the transformer network and discuss various modeling choices when using the transformer as the acoustic encoder. Relation to other works is also discussed in Section 3.5.

3.1 Self-Attention and Multi-Head Attention

Self-attention first computes the attention distribution over the input sequence using dot-product attention, i.e., for every $\boldsymbol{x}_t \in \mathbb{R}^{d_i}$, a

⋆ Equal contribution;

† Work was done when Andros was an intern at Facebook.

distribution $\boldsymbol{\alpha}_t$ is obtained by:

$$\alpha_{t\tau} = \frac{\exp(\beta \cdot \boldsymbol{x}_t^{\mathsf{T}} \mathbf{W}_{\mathrm{q}}^{\mathsf{T}} \mathbf{W}_{\mathrm{k}} \boldsymbol{x}_\tau)}{\sum_{\tau'} \exp(\beta \cdot \boldsymbol{x}_t^{\mathsf{T}} \mathbf{W}_{\mathrm{q}}^{\mathsf{T}} \mathbf{W}_{\mathrm{k}} \boldsymbol{x}_{\tau'})} \tag{1}$$

where $\mathbf{W}_{\mathrm{q}}, \mathbf{W}_{\mathrm{k}} \in \mathbb{R}^{d_k \times d_i}$ transforms $\boldsymbol{x}_t$ to *query* and *key* space, $\beta = \frac{1}{\sqrt{d_i}}$ is a scaling factor. Note that for language modeling, the dot-products between the current position and future positions are masked to $-\infty$ to prevent future information leaking to the current embedding. Though for acoustic modeling, it is possible to attend to the entire sequence, in many applications, we only attend to limited right context frames to enable streaming processing of speech signals (i.e., dot-product between t and $\tau, \tau > t + R$ is masked to $-\infty$). Given $\boldsymbol{\alpha}_t$, the output embedding of self-attention is obtained via:

$$\boldsymbol{z}_t = \sum_\tau \mathrm{Dropout}(\alpha_{t\tau}) \cdot \mathbf{W}_{\mathrm{v}} \boldsymbol{x}_\tau \tag{2}$$

where $\mathbf{W}_{\mathrm{v}} \in \mathbb{R}^{d_v \times d_i}$ maps the input vectors to *value* space.

Self-attention is often combined with multi-head attention (MHA), where h self-attention heads are applied individually on the input sequences, and the output of each head are concatenated and linearly transformed to a common space, i.e.,

$$\boldsymbol{z}_t = \mathbf{W}_{\mathrm{o}} \begin{pmatrix} \cdots \\ \sum_\tau \mathrm{Dropout}(\alpha_{t\tau}^{(i)}) \cdot \mathbf{W}_{\mathrm{v}}^{(i)} \boldsymbol{x}_\tau \\ \cdots \end{pmatrix} \tag{3}$$

where $\mathbf{W}_{\mathrm{o}} \in \mathbb{R}^{d_i \times h d_{\mathrm{v}}}$, $\alpha_{t\tau}^{(i)}$ and $\mathbf{W}_{\mathrm{v}}^{(i)}$ are the attention weights and the value matrix of the i-th head.

3.2 Architecture of Transformer

In addition to the MHA sub-layer, each transformer layer contains a fully-connected feed-forward network (FFN), which is composed by two linear transformations and a nonlinear activation function in between. The FFN network is applied to each position in the sequence separately and identically. To allow stacking many transformer layer together, residual connections are added to the MHA and FFN sub-layers. Dropouts are also applied after MHA and linear transformation as a form of regularization. Figure 1 summarizes the architecture of one transformer layer. Note that different from [14], layer normalization [25] is applied before MHA and FFN and the third layer normalization (LN_3 in Figure 1) is necessary to prevent bypassing the transformer layer entirely. Note, following [15], we use "gelu" non-linearity [26] in the FFN network.

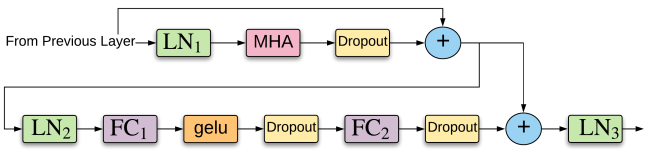

Fig. 1: Architecture of one transformer layer. "LN" means layer normalization [25]; "FC" means fully connected linear transformation; "gelu" means the gelu nonlinear activation [26].

3.3 Positional Embedding

One obvious limitation of the transformer layer is that the output is invariant to the input order permutation, i.e., for any permutation π applied on the input sequence $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T$, the output of the transformer layer can be obtained by applying the same permutation π on $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_T$. This means that transformer does not model the order of the input sequence. In the original transformer work [14], this is solved by injecting information about absolute positions into the input sequence via sinusoid positional embeddings. We argued that different from NLP applications, relative position could be more useful for speech signals. In this work, we compare a few ways to encode positional information into the input of transformer:

- *Sinusoid positional embedding:* a sinusoid positional embedding $\boldsymbol{p}_t$ is added to $\boldsymbol{x}_t$, where the i-th element of $\boldsymbol{p}_t$ is $\sin((t/10000)^{i/d_i})$ for even i and $\cos((t/10000)^{(i-1)/d_i}$ for odd i. This encodes *absolute positional* information;
- *Frame stacking:* a simple way to break the permutation invariance is to stack n contextual vectors together, i.e., $\overline{\boldsymbol{x}}_t = (\boldsymbol{x}_t^{\mathsf{T}}, \boldsymbol{x}_{t+1}^{\mathsf{T}}, \cdots, \boldsymbol{x}_{t+n-1}^{\mathsf{T}})^{\mathsf{T}}$. This encodes the *relative positional* information;
- *Convolutional embedding:* inspired by [27], we use 2D convolutional layers to implicitly encode the relative positional information. Convolutional embedding implicitly performs *frame stacking* as well as learns useful short-range spectral-temporal patterns [28].

3.4 Training Deep Transformers

Transformer layers can be stacked many times to form a very deep network. In our initial experiments, we found better accuracies can be obtained by deeper networks. However, after stacking many layers, it becomes difficult to train and often gets stuck in a bad local optimum. To enable training deep transformer, we used iterated loss [29], in which output of some intermediate transformer layers is also used to calculate auxiliary cross entropy losses. These auxiliary losses are interpolated to make the final loss function. Note that intermediate-layer-specific parameters (e.g., the linear transformation before the softmax operation) are discarded after training.

3.5 Relation to Other Works

The original transformer paper [14] proposed to use self-attention and cross-attention to replace the recurrence in encoder and decoder in a sequence-to-sequence model. Since we focus on hybrid speech recognition, we only use self-attention to replace the RNNs in the acoustic encoder in this work.

Self-attention based acoustic modeling has been explored in the past. In [20], self-attention is modified to attend to a fixed number of left and right context frames, and only one attention layer was used. By comparison, in our work attention heads attend to all the past frames, and we use both self-attention and FFN networks with a very deep structure, which is critical to achieve a good model accuracy. In [30], transformers are compared with RNNs in the sequence-to-sequence architecture. In [18], various positional embedding methods were investigated for a sequence-to-sequence model, where it is found that replacing the FFN network with a LSTM layer to make the self-attention layer position aware yielded better performance. Following [27], we use convolution layers as pre-processors for the transformer layer's input and compare it with other positional encoding methods in Section 4.2. In [31], a loss function similar to the iterated loss is used to enable training very deep transformers for character-level LMs; we demonstrate that it is also crucial for training deep transformer-based AMs.

4 Experiments

To evaluate the effectiveness of the proposed transformer-based acoustic model, we first perform experiments on the Librispeech

corpus [32]. This corpus contains about 960 hours of read speech data for training, and 4 development and test sets ({dev, test} - {clean,other}), where other sets are more acoustic challenging. No segmentation is performed for these test sets. The standard 4-gram language model (LM) with a 200K vocabulary is used for all first-pass decoding.

4.1 Experiment Setups

Following [24], we use context- and position-dependent graphemes (i.e., *chenones*) in all experiments. We bootstrap our HMM-GMM system using the standard Kaldi [33] Librispeech recipe. We use 1-state HMM topology with fixed self-loop and forward transition probability (both 0.5). 80-dimensional log Mel-filter bank features are extracted with a 10ms frame shift. A reduced 20ms frame rate is achieved either by stacking-and-striding 2 consecutive frames or by a stride-2 pooling in the convolution layer if it is used. We found that this not only reduces the computation but also slightly improves the recognition accuracy. Speed perturbation [34] and *SpecAugment* [10] (LD policy without time warping) are used. We focus on cross-entropy (CE) trained models and only selectively perform sMBR [35] training on top of the best CE setup.

Neural network training is performed using an in-house developed speech extension of the PyTorch-based *fairseq* [36] toolkit. Adam optimizer [37] is used in all experiments; the learning rate linearly warms up from 1e-5 to 1e-3 in the first 8000 iterations and stays at 1e-3 during the rest of training. We mainly compare full-context transformer with BLSTM in this work though we do have an initial investigation of transformers using limited right context. Dropout is used in all experiments: 0.1 for transformer and 0.2 for BLSTM. To improve training throughput, our batch size is dynamically determined so that we can occupy as much GPU memory as possible. For most of the experiments in this work, a batch contains around 10,000 to 20,000 frames, including padding frames. We train models using 32 Nvidia P100 GPUs for at most 100 epochs; training is usually done within 4 days. We did not perform thorough architecture searches for either transformer or BLSTM. For transformers, we mainly use a 12-layer transformer architecture with $d_i = 768$: per-head dimension is always 64 and the FFN dimension is always set to $4d_i$. This model has about 90M parameters. For BLSTMs, we follow [24] and consider two architectures, a 5-layer BLSTM with 800 units per layer per direction (about 94M parameters), and a 6-layer BLSTM with 1000 units (about 163M parameters)[1].

Training transformers requires some tricks. Due to the quadratically growing computation cost with respect to the input sequence length, we segment the training utterances into segments that are not longer than 10 seconds [2]. Though this creates a mismatch between training and testing, preliminary results show that training on shorter segments not only increases the training throughput but also helps the final WERs. We also found that transformers are more prone to over-fitting, thus require some regularization. We found *SpecAugment* [10] is effective: without it, WER starts to increase after only 3 epochs, while WER continues to improve during training with *SpecAugment*.

A fully-optimized, static 4-gram decoding graph is built using Kaldi. This decoding graph is used for first-pass decoding and n-best generation for neural LM rescoring. Test set WERs are obtained using the best model based on WER on the development set[3]. Following [38], the best checkpoints for test-clean and test-other are selected separately on the corresponding development sets [4]

4.2 Effect of Positional Embedding

In the first set experiment, we investigate the effect of four positional embeddings (PE) methods for transformer-based acoustic models. In the first method, we stack-and-stride every 2 frames: it does not break the permutation invariance in transformers, thus denoted as *None*. In the second method, the *Sinusoid* PE proposed in the original transformer paper [14], which encodes the *absolute* positional information, is used. In the third method, *Frame Stacking*, we stack the current frame and next 8 future frames followed by a stride-2 sampling to form a new input sequence to transformers. Note that since the stacked frames are partially overlapped with its neighboring stacked frames, the permutation invariance no longer holds. This method encodes *relative* positional information. In the fourth method, *Convolution*, we use two VGG blocks [39] beneath transformer layers: each VGG block contains 2 consecutive convolution layers with a 3-by-3 kernel followed by a ReLu non-linearity and a pooling layer; 32 channels are used in the convolution layer of the first VGG block and increase to 64 for the second block. Max-pooling is performed at a 2-by-2 grid, with stride 2 in the first block and 1 in the second block. For an input sequence of 80-dim feature vector at a 10ms rate, this VGG network produces a 2560-dim feature vector sequence at a 20ms rate. Note that the perception field of each feature vector output by the VGG network consists of 80ms left-context and 80ms right context, the same right context length as *Frame Stacking*. A linear projection is used to project the feature vector to the dimension accepted by transformers, in this case, 768.

Table 1*:* Effect of Positional Embeddings (PE) for Transformer.

PE	test-clean	test-other
None	3.11	6.94
Sinusoid	3.13	6.67
Frame Stacking	3.04	6.64
Convolution	2.87	6.46

4.3 Transformer vs. BLSTM

In the second set of experiments, we compare the transformer architecture with BLSTM. For a fair comparison, we try to build transformer and BLSTM-based models using similar number of parameters. First we compare a BLSTM model, *BLSTM(800, 5)*, i.e., 5 layers with 800 hidden units per layer per direction, with the transformer model in row 3, Table 1, dubbed *Trf-FS* since it uses *Frame Stacking*. To be able to compare our best performing transformer-based model with *Convolution* PE, we combine the same VGG blocks in row 4, Table 1 with BLSTM, producing *vggBLSTM(800, 5)*. Lastly, with about 163M parameters, we build the largest vggBLSTM model, *vggBLSTM(1000,6)*. To match the number of parameters of this model, we increase the number of transformer layers from 12 to 20. As shown in Table 2, transformer-based models consistently outperform BLSTM-based models by 2–4% on test-clean and 7–11% on test-other.

4.4 Effect of Iterated Loss

Table 2 shows that simply increasing the depth of transformers to 20 layers, we obtained about 5.5% relative WER reduction (6.10 vs. 6.46). Inspired by this, we try to increase the number of transformer

[1]We did not obtain further WER improvements by increasing number of parameters in BLSTM beyond 163M.

[2]This is achieved by aligning audio against the reference using an existing latency-controlled BLSTM acoustic model.

[3]We also average the last 10 epoch checkpoints to form an extra candidate.

[4]This is only to follow the same experimental protocol set by the prior work in [38] – most of the experimental results on both test sets, including the best WERs we reported in Table 4, are actually achieved by the same model.

Table 2*:* Architecture comparison on the Librispeech benchmark

Model Arch	#Params (M)	test-clean	test-other
BLSTM (800,5)	79	3.11	7.44
Trf-FS (768,12)	91	3.04	6.64
vggBLSTM (800,5)	95	2.99	6.95
vggTrf. (768,12)	93	2.87	6.46
vggBLSTM (1000,6)	163	2.86	6.63
vggTrf. (768, 20)	149	2.77	6.10

layers further. To make the model size manageable, we use a smaller embedding dimension, 512, for deep transformer models. Our initial attempt was not successful; deep transformer models (deeper than 20 layers) often got stuck in training and made little progress for a long time. We solved the problem with the iterated loss used in [29]: the output embeddings of the 6/12/18-th transformer layers are non-linearly transformed (projected to a 256-dimensional space with a linear transformation followed by a Relu non-linearity) and auxiliary CE losses are calculated separately. These additional CE losses are interpolated with the original CE loss with a 0.3 weight. With this iterated loss, we were able to train a 24-layer transformer model with only 81M model parameters in decoding[5] and obtain a 7% and 13% WER reduction on `test-clean` and `test-other`, respectively, over the *vggTrf(768, 12)* baseline.

Table 3*:* Using iterated loss to train deep transformer models.

Model Arch	Iter Loss	test-clean	test-other
vggTrf. (768, 12)	N	2.87	6.46
(Params: 93M)	Y	2.77	6.10
vggTrf. (512, 24)	N	not converged	
(Params: 81M)	Y	2.66	5.64

On top of this *vggTrf(512, 24)* model, we further perform sMBR training and it slightly improves to 2.60% and 5.59% on `test-clean` and `test-other`. We compare our results with some published state-of-the-art systems on Librispeech in Table 4: when the standard 4-gram LM is used in decoding, our system achieves 19% and 26% WER reduction on `test-clean` and `test-other` respectively, over previous best 4-gram only hybrid system [24][6]. We also built a transformer LM similar to the setup in [16] on the 800M text tokens provided by the Librispeech benchmark and performed n-best rescoring on the first pass decoding output. To the best of our knowledge, our final WERs (2.26/4.85) are state-of-the-art results on this widely used benchmark.

Table 4*:* Comparison with previous best results on Librispeech. "4g" means the stand 4-gram LM is used; "NNLM" means a neural LM is used.

Arch.	System	LM	test-clean	test-other
LAS	Park et al. [10]	NNLM + 4g	2.5	5.8
	Karita et al. [30]	NNLM	2.6	5.7
Hybrid	RWTH [38]	4g	3.8	8.8
		+NNLM	2.3	5.0
	Han et al. [41]	4g	2.9	8.3
		+NNLM	**2.2**	5.8
	Le et al. [24]	4g	3.2	7.6
	Ours	4g	**2.60**	**5.59**
		+NNLM	2.26	**4.85**

[5]There are 6M extra parameters only used in training.

[6]Note that [24] used LC-BLSTM [40] instead of full-context BLSTM.

Table 5*:* Forcing transformer models to use limited right context (RC) *per layer* during inference. Given a 12-layer transformer, an RC of 10 frames translates to 2.48 seconds of total lookahead.

RC	test-clean	test-other
∞	2.87	6.45
50	3.01	7.12
20	3.29	8.10
10	3.65	9.01

4.5 Limited Right Context

All the transformer-based experiments so far used full context. To understand to what extent the transformer relies on future frames to derive embeddings for the current frames, we take the *vggTrf(768, 12)* model (row 4, Table 2) and force every layer to attend to a fixed limited right context during inference. Interestingly, though this creates a large mismatch between training and inference, the resultant systems can still yield reasonable WERs if the number of right context frames is large enough. Note that though each layer only requires limited right context frames, the overall right context length is added up by the right context length of every transformer layer, therefore we still end up with a large look-ahead window into the future, which makes it less possible to be used in a streaming ASR application. We will investigate transformer-based acoustic models with the streaming constraint in our future study.

4.6 Large Scale Experiments

Finally, we perform a large scale experiment on one of our internal tasks, *English video ASR*. The training set consists of 13.7K hours of videos (from 941.6K video clips) shared publicly by users; only the audio part of those videos are used in our experiments. These data are completely anonymized; both transcribers and researchers do not have access to any user-identifiable information. Due to the data nature, it is a very diverse and challenging task. About 9 hours (from 620 video clips) data are held out for `dev` set. 3 test sets are used for evaluation purpose: an 8.5-hour `curated` set of carefully select very clean videos, an 19-hour `clean` set and a 18.6-hour `noisy` set. For our initial evaluation purpose, both training and test sets are segmented into maximum 10 second segments.

Due to time limit, we only built *vggTrf(768, 12)* without the iterated loss and *vggBLSTM(800, 5)* on this task. Table 6 shows that on this task, the proposed transformer-based acoustic model outperform vggBLSTM by 4.0-7.6%. We will report more results in our future work.

Table 6*:* Experiment results on our internal *English video ASR* task.

Model	curated	clean	noisy
vggBLSTM(800,5)	10.72	15.97	22.13
vggTrf(768,12)	9.90	15.26	21.25

5 Discussions And Conclusions

In this work, we proposed and evaluated transformer-based acoustic models for hybrid speech recognition. A couple of model modeling choices are discussed and compared. We demonstrated that transformer can significantly outperforms BLSTM and give the best acoustic models on Librispeech benchmark. Initial study on a much larger and more challenging dataset also confirms our findings.

There are many works we are yet to explore. For example, our experiments did not show to what extent transformer's superior performance comes from replacing recurrence with self-attention, while other modeling techniques from transformer can be borrowed to improve RNNs as well [42]. The quadratically growing cost with respect to the length of speech signals is still a major blocker for transformer-based acoustic models to be used in practice. These questions will be studied in our future work.

6 References

[1] G. Hinton, L. Deng, D. Yu, et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011.

[3] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014.

[4] O. Abdel-Hamid, A. Mohamed, H. Jiang, et al., "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

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

[6] S. Zhang, H. Jiang, S. Wei, and L. Dai, "Feedforward sequential memory neural networks without recurrent feedback," *arXiv preprint arXiv:1510.02693*, 2015.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] D. Bahdanau, J. Chorowski, D. Serdyuk, et al., "End-to-end attention-based large vocabulary speech recognition," in *Proc. ICASSP*, 2016.

[9] C.-C. Chiu, T.N. Sainath, Y. Wu, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[10] D. S. Park, W. Chan, Y. Zhang, et al., "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[11] Y. He, T. N. Sainath, R. Prabhavalkar, et al., "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019.

[12] Y. Bengio, P. Simard, et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[13] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[14] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[15] J. Devlin, M.-W. Chang, K. Lee, et al., "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] A. Radford, K. Narasimhan, Tim S., et al., "Improving language understanding by generative pre-training," 2018.

[17] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018.

[18] M. Sperber, J. Niehues, G. Neubig, et al., "Self-attentional acoustic models," *arXiv preprint arXiv:1803.09519*, 2018.

[19] S. Zhou, L. Dong, S. Xu, and B. Xu, "Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin Chinese," *arXiv preprint arXiv:1804.10752*, 2018.

[20] D. Povey, Hossein Hadian, P. Ghahremani, et al., "A time-restricted self-attention layer for asr," in *Proc. ICASSP*, 2018, pp. 5874–5878.

[21] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *Proc. ICASSP*, 2019, pp. 7115–7119.

[22] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer Science & Business Media.

[23] M.-Y. Hwang and X. Huang, "Subphonetic modeling with markov states-senone," in *Proc. ICASSP*, 1992, vol. 1, pp. 33–36.

[24] D. Le, X. Zhang, W. Zheng, et al., "From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition," *arXiv preprint arXiv:1910.01493*, 2019.

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

[26] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[27] A. Mohamed, D. Okhonko, and L. Zettlemoyer, "Transformers with convolutional context for asr," *arXiv preprint arXiv:1904.11660*, 2019.

[28] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. ICASSP*, 2017, pp. 4845–4849.

[29] A. Tjandra, C. Liu, F. Zhang, et al., "Deja-vu: Double feature presentation in deep transformer networks," *Submitted to ICASSP*, 2020.

[30] S. Karita, N. Chen, T. Hayashi, et al., "A Comparative Study on Transformer vs RNN in Speech Applications," *arXiv preprint arXiv:1909.06317*, 2019.

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

[32] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[33] D. Povey, A. Ghoshal, G. Boulianne, et al., "The kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[34] T. Ko, V. Peddinti, D. Povey, et al., "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015.

[35] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks.," in *Proc. Interspeech*, 2013, vol. 2013, pp. 2345–2349.

[36] O. Myle, E. Sergey, B. Alexei, F. Angela, et al., "fairseq: A Fast, Extensible Toolkit for Sequence Modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

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

[38] C. Lüscher, E. Beck, K. Irie, et al., "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention-w/o Data Augmentation," *arXiv preprint arXiv:1905.03072*, 2019.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] Y. Zhang, G. Chen, D. Yu, et al., "Highway long short-term memory RNNs for distant speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 5755–5759.

[41] K. J. Han, R. Prieto, K. Wu, and T. Ma, "State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions," *arXiv preprint arXiv:1910.00716*, 2019.

[42] M. Chen, O. Firat, A. Bapna, et al., "The best of both worlds: Combining recent advances in neural machine translation," *arXiv preprint arXiv:1804.09849*, 2018.