SPEECH-TRANSFORMER: A NO-RECURRENCE SEQUENCE-TO-SEQUENCE MODEL FOR SPEECH RECOGNITION

Linhao Dong[1,2]*, Shuang Xu*[1]*, Bo Xu*[1]

[1]Institute of Automation, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
{donglinhao2015, shuang.xu, xubo}@ia.ac.cn

ABSTRACT

Recurrent sequence-to-sequence models using encoder-decoder architecture have made great progress in speech recognition task. However, they suffer from the drawback of slow training speed because the internal recurrence limits the training parallelization. In this paper, we present the Speech-Transformer, a no-recurrence sequence-to-sequence model entirely relies on attention mechanisms to learn the positional dependencies, which can be trained faster with more efficiency. We also propose a 2D-Attention mechanism, which can jointly attend to the time and frequency axes of the 2-dimensional speech inputs, thus providing more expressive representations for the Speech-Transformer. Evaluated on the Wall Street Journal (WSJ) speech recognition dataset, our best model achieves competitive word error rate (WER) of 10.9%, while the whole training process only takes 1.2 days on 1 GPU, significantly faster than the published results of recurrent sequence-to-sequence models.

***Index Terms*—** Speech Recognition, Sequence-to-Sequence, Attention, Transformer

1. INTRODUCTION

In speech recognition field, sequence-to-sequence (seq2seq) models have made great strides forward recently [1, 2, 3, 4, 5, 6, 7, 8]. They remove the unreasonable frame-independence assumption made by Hidden Markov Model (HMM) [9] and Connectionist Temporal Classification (CTC) [10] models, enabling themselves to learn an implicit language model and optimize WER more directly [11]. In recent years, seq2seq models have achieved significant WER reduction benefiting from building deeper encoder [5], the effective label smoothing schemes [4] and the latent sequence decomposition methods [6]. In addition, some monotonic attention models [8, 12] have also been explored and developed. These efforts are jointly driving seq2seq model closer to practical application.

Although the seq2seq models have shown success in speech recognition task, they still suffer from a drawback: slow training speed [13]. In most of the proposed seq2seq models, recurrent neural networks (RNNs) [14, 15, 16] play an essential role when generating sequential hidden representations (encoding) and emitting character according to soft alignment at different time (decoding). Unfortunately, the sequential nature of RNNs limits the computation parallelization of training. This dilemma becomes especially severe for speech recognition task since speech sequences are commonly long and it is rather time-consuming when processing recurrence.

This work is supported by the National Key Research and Development Program of China under No.2017YFB1002102 and Beijing Digital Content Engineering Technology Research Center under No.Z171100002217015.

Recently, Vaswani et al. [17] proposed a no-recurrence sequence-to-sequence model, called the Transformer, which achieved state-of-the-art performance on WMT 2014 English-to-French translation task with markedly less training cost. Its fundamental module is self-attention, a mechanism relates all the position-pairs of a sequence to extract a more expressive sequence representation. Since the self-attention can draw the dependencies between different positions through the position-pair computation rather than the position-chain computation of RNNs, it just needs to be calculated once to obtain the transformed representation, rather than be computed one by one in RNNs. Therefore, the Transformer relying solely on attention mechanisms can be trained faster with more parallelization, which is exactly needed by the seq2seq models in automatic speech recognition (ASR).

In this paper, we successfully introduce the Transformer to ASR task and we term our model the Speech-Transformer, a new seq2seq model that transforms speech feature sequences to the corresponding character sequences. Additionally, we propose a 2D-Attention mechanism, which is inspired by the time-frequency LSTM in [18] but replaces the time-frequency recurrence with both the temporal and spectral dependencies captured by attention. By deploying the WSJ speech recognition dataset, we find our proposed 2D-Attention mechanism achieves better performance than the convolutional networks in [5], thus providing a more discriminated representation for the Speech-Transformer. Moreover, our best model obtains 10.92% WER after training 1.2 days on 1 NVIDIA K80, which is a comparable performance with a considerable reduction of training cost compared with most recurrent seq2seq models.

2. MODEL DESCRIPTION

Similarly to previous seq2seq models, the Speech-Transformer is based on the encoder-decoder architecture: the encoder transforms a speech feature sequence $(x_1, \ldots, x_T)$ to a hidden representation $\mathbf{h} = (h_1, \ldots, h_L)$. Given $\mathbf{h}$, the decoder then generates an output sequence $(y_1, \ldots, y_S)$ one character at a time. At each step, the decoder consumes the previously emitted characters as additional inputs when emitting the next character.

However, as a no-recurrence seq2seq model, the Speech-Transformer differs from recurrent seq2seq models mainly on two aspects: Firstly, both the encoder and decoder are composed of multi-head attention and position-wise feed-forward networks as described in section 2.1, rather than RNNs. Secondly, the encoder outputs $\mathbf{h}$ are attended by each decoder block respectively, as shown in Figure 2, replacing the one-step intermediary attention of recurrent seq2seq models. Next, we will describe the details of the Speech-Transformer in the rest of this section:

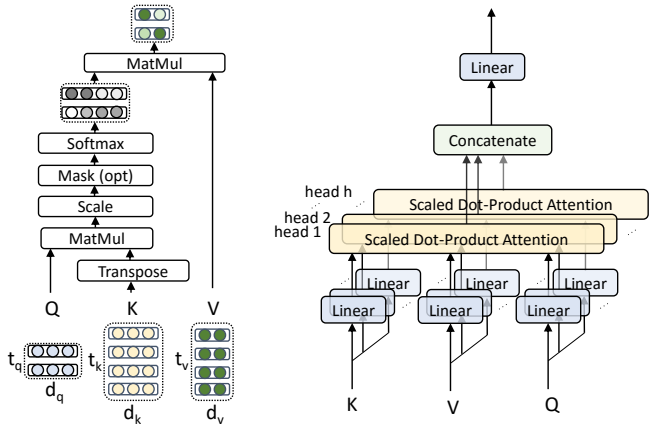

Fig. 1. (left) Scaled Dot-Product Attention with a simple illustration of its running process. (right) Multi-Head Attention consists of multiple Scaled Dot-Product Attention performing in parallel.

2.1. Core Module of the Speech-Transformer

2.1.1. Scaled Dot-Product Attention

Self-attention, a mechanism that relates different positions of input sequences to compute representations for the inputs. Concretely, it has three inputs: queries, keys and values. One query's output is computed as a weighted sum of the values, where each weight of the value is computed by a designed function of the query with the corresponding key. Here, we use Scaled Dot-Product Attention, an effective self-attention mechanism demonstrated in [17]. As shown in the left half of Figure 1 , Let $\mathbf{Q} \in \mathbb{R}^{t_q \times d_q}$ be the queries, $\mathbf{K} \in \mathbb{R}^{t_k \times d_k}$ be the keys and $\mathbf{V} \in \mathbb{R}^{t_v \times d_v}$ be the values, where t_* are the element numbers in different inputs and d_* are the corresponding element dimensions. Normally, $t_k = t_v$, $d_q = d_k$. The outputs of self-attention are computed as:

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

where the scalar $1/\sqrt{d_k}$ is used to prevent softmax function into regions that has very small gradients.

2.1.2. Multi-Head Attention

Multi-head attention, a core module of the Speech-Transformer, is applied to leverage different attending representations jointly. As the right half of Figure 1 shows, multi-head attention calculates h times Scaled Dot-Product Attention, where h means the head number. Before performing each attention, there are three linear projections to transform the queries, keys and values to more discriminated representations respectively. Then, each Scaled Dot-Product Attention is calculated independently, and their outputs are concatenated and fed into another linear projection to obtain the final d_{model}-dimensional outputs:

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

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

In the above equation, since $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ in the Speech-Transformer have the same dimension of d_{model}, the projection matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$. $d_q = d_k = d_v = d_{model}/h$ throughout the paper.

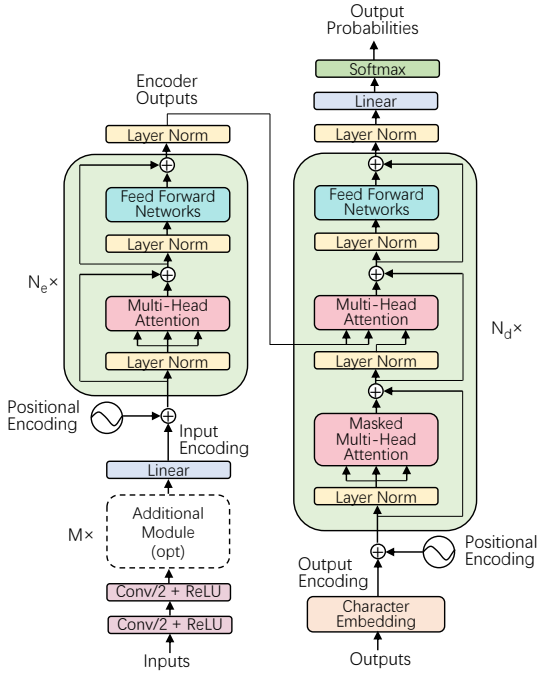

Fig. 2. Model architecture of the Speech-Transformer.

2.1.3. Position-wise Feed-Forward Network

Position-wise feed-forward network is another core module of the Speech-Transformer. It consists of two linear transformations with a ReLU activation in between. The dimensionality of input and output is d_{model}, and the inner layer has dimensionality d_{ff}. Specifically,

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{4}$$

where the weights $\mathbf{W}_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$ and the biases $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$, $\mathbf{b}_2 \in \mathbb{R}^{d_{model}}$. The linear transformations are the same across different positions.

2.2. Model Architecture

The Speech-Transformer aims at transforming the speech feature sequence to the corresponding character sequence. The feature sequence, commonly a few times longer than the character sequence, can be depicted as 2-dimensional spectrograms with time and frequency axes. Therefore, we choose the convolutional networks to exploit the structure locality of spectrograms and mitigate the length mismatch by striding along time. Based on above, we present the model architecture of the Speech-Transformer, and the details of its encoder and decoder are as follows:

The encoder is shown in the left half of Figure 2. We firstly stack two 3×3 CNN layers with stride 2 for both time and frequency dimensions to prevent the GPU memory overflow and produce the approximate hidden representation length with the character length. Then, we can optionally stack M additional modules which are applied to extracting more expressive representations for our Speech-Transformer, which will be detailed in section 4.2. Next, we perform a linear transformation on the flattened feature map outputs to obtain the vectors of dimension d_{model}, which is called input encoding here. Afterwards, in order to enable the model to attend by relative positions, the d_{model}-dimensional positional encoding is added to the input encoding:

$$\mathbf{PE}_{(pos,i)} = \begin{cases} sin(pos/10000^{2i/d_{model}}) & 0 \leq i < d_{model}/2 \\ cos(pos/10000^{2i/d_{model}}) & d_{model}/2 \leq i < d_{model} \end{cases} \quad (5)$$

Where pos represents the position in sequence, i represents the i-th dimension. The positional encoding works because for arbitrary fixed offset k, $\mathbf{PE}_{pos+k}$ can be represented as a linear function of $\mathbf{PE}_{pos}$. We can obtain the final encoded outputs by inputting the sum of input encoding and positional encoding to a stack of N_e encoder-blocks, each of them has two sub-blocks: The first is a multi-head attention whose queries, keys and values come from the outputs of the previous block. And the second is position-wise feed-forward networks. Meanwhile, layer normalization and residual connection are introduced to each sub-block for effective training. Given sub-block inputs $\mathbf{x}$, the corresponding outputs are:

$$\mathbf{x} + \text{SubBlock}(\text{LayerNorm}(\mathbf{x})) \quad (6)$$

The decoder is shown in the right half of Figure 2. We firstly employ a learned character-level embedding to convert the character sequence to the output encoding of dimension d_{model}, which is added with the positional encoding. Then, the sum of them are inputted to a stack of N_d decoder-blocks to obtain the final decoder outputs. Differently from the encoder-block, each decoder-block has three sub-blocks: The first is a masked multi-head attention which has the same queries, keys and values. And the masking is utilized to ensure the predictions for position j can depend only on the known outputs at positions less than j. The second is a multi-head attention whose keys and values come from the encoder outputs and queries come from the previous sub-block outputs. The third is also position-wise feed-forward networks. Like the encoder, layer normalization and residual connection are also performed to each sub-block of the decoder. Finally, the outputs of decoder are transformed to the probabilities of output classes by a linear projection and a subsequent softmax function.

3. PROPOSAL: 2D-ATTENTION MECHANISM

The attention mechanism used in the encoder-block of the Speech-Transformer relates positions on time axis to build the temporal dependencies. Here we call it 1D-Attention. However, speech feature sequence is often transformed to 2-dimensional spectrograms with both time and frequency axes. When reading a spectrogram, a human predicts its pronunciation relying on the varying correlations between different frequencies with time. Therefore, attending to both time and frequency axes may be beneficial to the modeling of the temporal and spectral dynamics in a spectrogram.

Motivated by the analysis above, we propose a 2D-Attention block which is illustrated in Figure 3. Firstly, it performs three convolutional networks on the n-channels spectrograms to extract the representations of queries, keys and values independently, where the output channels of each convolution network are c. Then, it introduces two types of attention to capturing temporal and spectral dependencies respectively: As Figure 3 shows, the bottom one attends to the time axis using c Scaled Dot Product Attentions, each attention handles the queries, keys and values from the corresponding channel. The top one applies c Scaled Dot Product Attentions to the transposed queries, keys, values in order to attend to the frequency axis, and its outputs are then transposed to original size. Finally, the outputs of 2D-Attention are concatenated and fed into another

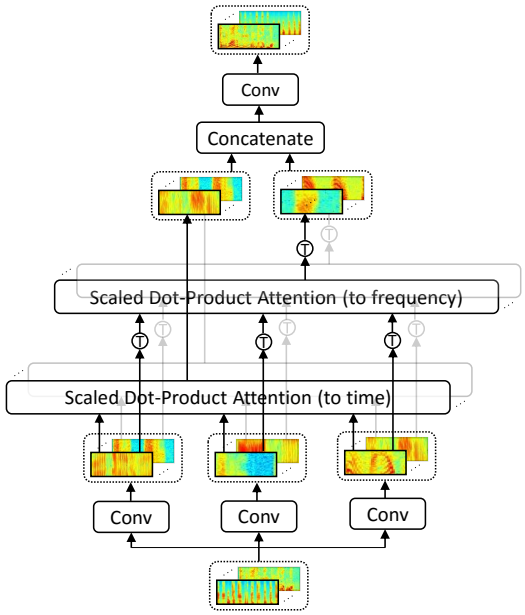

Fig. 3. Illustration of the Proposed 2D-Attention mechanism. The colored rectangles represent spectrograms, and the circle with T inside represents the transposition operation.

convolution network to obtain the final n-channels outputs:

$$\text{2D-Attention}(\mathbf{I}) = \mathbf{W}^O * \text{Concat}(\text{channel}_1^t, \ldots, \text{channel}_c^t, \text{channel}_1^f, \ldots, \text{channel}_c^f) \quad (7)$$

$$\text{where} \quad \text{channel}_i^t = \text{Attention}((\mathbf{W}_i^Q * \mathbf{I}), (\mathbf{W}_i^K * \mathbf{I}), (\mathbf{W}_i^V * \mathbf{I})) \quad (8)$$

$$\text{channel}_i^f = \text{Attention}((\mathbf{W}_i^Q * \mathbf{I})^T, (\mathbf{W}_i^K * \mathbf{I})^T, (\mathbf{W}_i^V * \mathbf{I})^T)^T \quad (9)$$

Where, $\mathbf{I}$ is the n-channels inputs, * represents the convolutional operation, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ represent the filters applied on $\mathbf{I}$ to obtain the queries, keys, values of channel i, respectively. $\mathbf{W}^O$ represents the filters applied on the $2c$ concatenated channels to obtain the final n-channels outputs.

4. EXPERIMENTS

4.1. Experimental Setups

We experimented with the Wall Street Journal (WSJ) dataset, training on si284, validating on dev93 and evaluating on eval92 set. The input acoustic features were 80-dimensional filterbanks extracted with a hop size of 10ms and a window size of 25ms, extended with temporal first and second order differences and per-speaker mean subtraction and variance normalization. The output alphabet of target text consisted of 31 classes, 26 lowercase letters, apostrophe, period, space, noise marker and end-of-sequence tokens.

In the training stage, the samples were batched together by approximate feature sequence length and each training batch contained 20000-frames features. We trained the model on 1 NVIDIA K80 GPU for a total of 100k steps. We used the Adam optimizer [19] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$ and varied the learning rate over the course of training, according to the formula:

$$\text{lrate} = k \cdot d_{model}^{-0.5} \cdot \min(n^{-0.5}, n \cdot warmup\_n^{-1.5}) \quad (10)$$

where n is the step number, k is a tunable scalar, and the learning rate increases linearly for the first $warmup\_n$ training steps and decreases thereafter proportionally to the inverse square root of the step number. We used $warmup\_n = 25000$, $k = 10$ and decreased k to 1 when the model converged. In order to prevent over-fitting, we used the $neighborhood$ smoothing scheme proposed in [4], and the probability of correct label was set to 0.8. Meanwhile, we set both of residual dropout and attention dropout to 0.1, where the residual dropout is applied to each sub-block before adding the residual information, the attention dropout is performed on the softmax activations in each attention. In addition, we encouraged the model attending to closer positions by adding bigger penalty on the attention weights of more distant position-pairs. All the convolutional networks in our model used 64 output channels and each of them was followed by a batch normalization layer for faster convergence.

After training, we averaged the last 10 checkpoints, which were written at 10-minute intervals in TensorFlow framework [20]. Then, we performed decoding using beam search with a beam size of 10 and length penalty $\alpha = 1.0$ [21]. Last but not least, all our WER results in section 4.2 were averaged over two runs.

4.2. Results

We first explore different hyper-parameter combinations for the Speech-Transformer, including the encoder block number N_e, the decoder block number N_d and the feed-forward inner dimension d_{ff}. The model dimension $d_{model} = 256$ and the head number $h = 4$ are kept unchanged during comparison.

Table 1. The performance comparisons of different super-parameter combinations for the Speech-Transformer.

Model	N_e	N_d	d_{ff}	WER
6Enc6Dec (base model)	6	6	1024	12.20
12Enc6Dec-wide (big model)	12	6	2048	**10.92**
4Enc8Dec	4	8	1024	13.26
8Enc4Dec	8	4	1024	11.95
8Enc4Dec-wide	8	4	2048	11.56
10Enc5Dec-wide	10	5	2048	11.01

As Table 1 shows, when the total block numbers of encoder and decoder are kept identical (6Enc6Dec, 4Enc8Dec, 8Enc4Dec), the model with more encoder blocks achieves better performance. We conjecture this is because deeper encoder can extract a more discriminated representation of acoustic information, which is presumably more important than more character spelling knowledge to speech recognition accuracy. In addition, using wider inner dimension (8Enc4Dec, 8Enc4Dec-wide) and adding the model depth (8Enc4Dec-wide, 10Enc5Dec-wide, 12Enc6Dec-wide) are also beneficial to the WER performance. Then, we establish our base model and big model for further investigation.

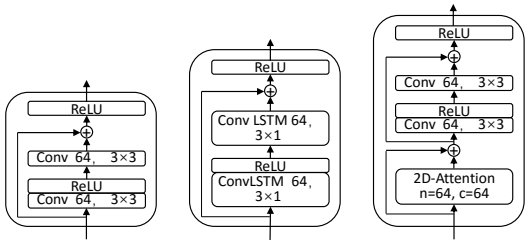

Fig. 4. (left) The ResCNN module. (medium) The ResCNNLSTM module. (right) Our designed 2D-Attention module.

Then, we investigate the performance of adding different additional modules to our base model, including ResCNN, ResCNNLSTM and our designed 2D-Attention module. Their structures are illustrated in Figure 4. Additionally, we set a comparison of adding extra encoder-blocks to our base model and denote it as 1D-Attention. The WER results are listed in Table 2.

Table 2. Results of adding different additional modules to the Speech-Transformer.

Model	WER
base model	12.20
base model + 4×ResCNN	11.90
base model + 4×ResCNNLSTM	12.01
base model + 2×1D-Attention	11.69
base model + 2×2D-Attention	**11.43**
big model	**10.92**
big model + 2×2D-Attention	11.01

It can be observed that the 2D-Attention module obtains the best performance when applied to our base model, even better than the ResCNNLSTM module which performs effectively in recurrent seq2seq models [5]. This can be explained by the fact that the 2D-Attention module can jointly capture the temporal and spectral relations, which is in line with the human spectrogram reading habits. In contrast, although the ResCNNLSTM captures temporal information by recurrent connections and maintains the spectral locality by convolutional networks, it doesn't take advantage of the spectral correlations and provides limited error reduction, neither does the ResCNN. However, when we apply 2D-Attention module to our big model, it shows a little performance reduction. We analyze this is probably due to the redundant extraction of acoustic information caused by excessive encoder components.

We gather the published character-based WSJ results without extra language models in Table 3. Our big model achieves the WER of 10.92% and converges after training 1.2 days on 1 NVIDIA K80, which is a comparable performance with most of the published models while consuming a significantly small fraction of training costs.

Table 3. Character-based results without language model on WSJ.

Model	Training time	eval92
CTC[22]	-	27.3
seq2seq[2]	-	18.6
seq2seq + deep convolutional[5]	5 days on 10 GPUs [1]	10.5
seq2seq + Unigram LS[4]	-	10.6
Speech-Transformer (big)	1.2 days on 1 K80	10.9

5. CONCLUSION

In this work, we presented a no-recurrence seq2seq model, the Speech-Transformer, aiming at mitigating the heavy training costs among most recurrent seq2seq models. We also explored various additional modules and found the best performance was obtained by our proposed 2D-Attention module, which is in line with the human spectrogram reading habits and extracts discriminated representations for the upper encoder-blocks. On the WSJ dataset, our big model converged with considerably small training costs and achieved competitive WER performance, which shows the efficiency and effectiveness of the Speech-Transformer.

[1] Their work didn't mention specific GPU products used for training.

6. REFERENCES

[1] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.

[2] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[3] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.

[4] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv preprint arXiv:1612.02695*, 2016.

[5] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4845–4849.

[6] William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly, "Latent sequence decompositions," *arXiv preprint arXiv:1610.03035*, 2016.

[7] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4835–4839.

[8] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Local monotonic attention mechanism for end-to-end speech recognition," *arXiv preprint arXiv:1705.08091*, 2017.

[9] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[10] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[11] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, et al., "Exploring neural transducers for end-to-end speech recognition," *arXiv preprint arXiv:1707.07413*, 2017.

[12] Colin Raffel, Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck, "Online and linear-time attention by enforcing monotonic alignments," *arXiv preprint arXiv:1704.00784*, 2017.

[13] Dong Yu and Jinyu Li, "Recent progresses in deep learning based acoustic models," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 396–409, 2017.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[15] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[18] Jinyu Li, Abdelrahman Mohamed, Geoffrey Zweig, and Yifan Gong, "Exploring multidimensional lstms for large vocabulary asr," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4940–4944.

[19] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[21] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[22] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.