TRANSFORMER-TRANSDUCER: END-TO-END SPEECH RECOGNITION WITH SELF-ATTENTION

Ching-Feng Yeh ⋆, Jay Mahadeokar ⋆, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, Michael L. Seltzer

Facebook AI, USA

ABSTRACT

We explore options to use Transformer networks in neural transducer for end-to-end speech recognition. Transformer networks use self-attention for sequence modeling and comes with advantages in parallel computation and capturing contexts. We propose 1) using VGGNet with causal convolution to incorporate positional information and reduce frame rate for efficient inference 2) using truncated self-attention to enable streaming for Transformer and reduce computational complexity. All experiments are conducted on the public LibriSpeech corpus. The proposed Transformer-Transducer outperforms neural transducer with LSTM/BLSTM networks and achieved word error rates of 6.37 % on the `test-clean` set and 15.30 % on the `test-other` set, while remaining streamable, compact with 45.7M parameters for the entire system, and computationally efficient with complexity of $O(T)$, where T is input sequence length.

Index Terms— transformer, transducer, end-to-end, self-attention, speech recognition

1. INTRODUCTION

There has been significant progress on automatic speech recognition (ASR) technologies over the past few years due to the adoption of deep neural networks [1]. Conventionally, speech recognition systems involve individual components for explicit modeling on different levels of signal transformation: acoustic models for audio to acoustic units, pronunciation model for acoustic units to words and language model for words to sentences. This framework is often referred to as the "traditional" hybrid system. Conventionally, individual components in the hybrid system can be optimized separately. For example, CD-DNN-HMM [1] focuses on maximizing the likelihood between acoustic signals and acoustic models with frame-level alignments. For language modeling, both statistical n-gram models [2] and more recently, neural-network-based models [3] aim to model purely the connection between word tokens.

Hybrid systems achieved significant success [4] but also present challenges. For example, hybrid system requires more human intervention in the building process, including the design of acoustic units, the vocabulary, the pronunciation model and more. In addition, an accurate hybrid system often comes with the cost of higher computational complexity and memory consumption, thus increasing the difficulty of deploying hybrid systems in resource-limited scenarios such as on-device speech recognition. Given the challenges, the interests in end-to-end approaches for speech recognition have surged recently [5–12]. Different from hybrid systems, end-to-end approaches aim to model the transformation from audio signal to word tokens directly, therefore the model becomes simpler and requires less human intervention. In addition to the simplicity of training process, end-to-end systems also demonstrated promising recognition accuracy [11]. Among many end-to-end approaches, recurrent neural network transducer (RNN-T) [5, 6] provides promising potential on footprint, accuracy and efficiency. In this work, we explore options for further improvements based on RNN-T.

Recurrent neural networks (RNNs) such as long-short term memory (LSTM) [13] networks are good at sequence modeling and widely adopted for speech recognition. RNNs rely on the recurrent connection from the previous state $\boldsymbol{h}_{t-1}$ to the current state $\boldsymbol{h}_t$ to propagate contextual information. This recurrent connection is effective but also presents challenges. For example, since $\boldsymbol{h}_t$ depends on $\boldsymbol{h}_{t-1}$, RNNs are difficult to compute in parallel. In addition, $\boldsymbol{h}_t$ is usually of fixed dimensions, which means all historical information is condensed into a fixed-length vector and makes capturing long contexts also difficult. The attention mechanism [14, 15] was introduced recently as an alternative for sequence modeling. Compared with RNNs, the attention mechanism is non-recurrent and can compute in parallel easily. In addition, the attention mechanism can "attend" to longer contexts explicitly. With the attention mechanism, the Transformer model [14] achieved state-of-the-art performance in many sequence-to-sequence tasks [15, 16].

In this paper, we explore options to apply Transformer networks in the neural transducer framework. VGG networks [17] with causal convolution are adopted to incorporate contextual information into the Transformer networks and reduce the frame rate for efficient inference. In addition, we use truncated self-attention to enable streaming inference and reduce computational complexity .

2. NEURAL TRANSDUCER (RNN-T)

In nature, speech recognition is a sequence-to-sequence (audio-to-text) task in which the lengths of input and output sequences can vary. As an end-to-end approach, connectionist temporal classification (CTC) [9] was introduced before RNN-T to model such sequence-to-sequence transformation. Given input sequence $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$, where $\boldsymbol{x}_t \in \mathcal{R}^d$ and T is the input sequence length, output sequence $\mathbf{y} = (y_1, y_2, \cdots, y_u)$, where $y_u \in \mathcal{Z}$ represent output symbols and U is the output sequence length, CTC introduces an additional "blank" label b and models the posterior probability of $\mathbf{y}$ given $\mathbf{x}$ by:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \boldsymbol{H}_{\text{ctc}}(\mathbf{x},\mathbf{y})} \prod_{t=1}^{T} P(\hat{y}_t|\boldsymbol{x}_1 \cdots \boldsymbol{x}_T) \tag{1}$$

where $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots \hat{y}_T) \in \boldsymbol{H}_{\text{ctc}}(\mathbf{x}, \mathbf{y}) \subset \{\mathcal{Z} \cup b\}^T$ correspond to any possible paths such that after removing b and repeated consecutive symbols of $\hat{\mathbf{y}}$ yields $\mathbf{y}$.

⋆ Equal contribution.

The formulation of CTC assumes that symbols in the output sequence are conditionally independent of one another given the input sequence. The RNN-T model improves upon CTC by making the output symbol distribution at each step dependent on the input sequence and previous non-blank output symbols in the history:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \boldsymbol{H}_{\text{rnnt}}(\mathbf{x},\mathbf{y})} \prod_{i=1}^{T+U} P(\hat{y}_i|\boldsymbol{x}_1 \cdots \boldsymbol{x}_{t_i}, y_1 \cdots y_{u_i-1}) \quad (2)$$

where $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots \hat{y}_{T+U}) \in \boldsymbol{H}_{\text{rnnt}}(\mathbf{x}, \mathbf{y}) \subset \{\mathcal{Z} \cup b\}^{T+U}$ correspond to any possible paths such that after removing b and repeated consecutive symbols of $\hat{\mathbf{y}}$ yields $\mathbf{y}$. By explicitly conditioning the current output on the history, RNN-T outperforms CTC when no external language model is present [6, 7]. RNN-T can be implemented in the encoder-decoder framework, as illustrated in Fig. 1. The encoder encodes the input acoustic sequence $\mathbf{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$ to $\mathbf{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_{T'})$ with potential subsampling $T' \leq T$. And the decoder contains a predictor to encode the previous non-blank output symbol y_{u-1} for the logits $\boldsymbol{z}_{t,u}$ to condition on. It's worth noting that only when the most probable symbol $\hat{y}_u$ is non-blank the input to predictor y_{u-1} will be updated, so that the conditioning encoding $\boldsymbol{p}_u$ only changes when non-blank output symbols are observed. From the illustration, we see that RNN-T incorporates a language model of output symbols internally in the decoder.

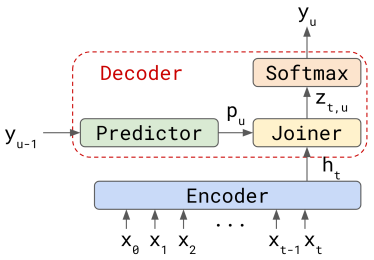

Fig. 1: Neural Transducer.

There are many architectures that can be used as encoders and predictors. The functionality of these blocks is to take a sequence and find a higher-order representations. Recurrent neural networks (RNNs) such as LSTM [13] have been successfully used for such functionality. In this paper, we explore Transformer [14, 15] as an alternative for sequence encoding in RNN-T. Since Transformer is not recurrent in nature, we refer to the architecture illustrated in Fig. 1 as simply "neural transducer" [18] for the rest of the paper.

3. TRANSFORMER

The attention mechanism [19] is one of the core ideas of Transformer [15]. It was proposed to model correlation between contextual signals and produced state-of-the-art performance in many domains including machine translation [15] and natural language processing [14]. Similar to RNNs, attention mechanism aims to encode the input sequence to a higher-level representation by formulating the encoding function into the relationship between queries Q, keys K and values V and describing the similarities between them with:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (3)$$

where $Q \in \mathcal{R}^{T_q \times d_k}$, $K \in \mathcal{R}^{T_k \times d_k}$ and $V \in \mathcal{R}^{T_k \times d_v}$. This mechanism becomes "self-attention" when $Q = K = V = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$. A self-attention block encodes the input $\mathbf{x}$ to a higher-level representation $\mathbf{h}$, just like RNNs but without recurrence. Compared with RNNs where $\boldsymbol{h}_t$ depends on $\boldsymbol{h}_{t-1}$, self-attention has no recurrent connections between time steps in the encoding $\mathbf{h}$, therefore it can generate encoding efficiently in parallel. In addition, compared with RNNs where contexts are condensed into fixed-length states for the next time step to condition on, self-attention "pays attention" to all available contexts to better model the context within the input sequence.

3.1. Multi-Head Self-Attention

The attention mechanism can be further extended to multi-head attention, in which 1) dimensions of input sequences are split into multiple chunks with multiple projections 2) each chunk goes through independent attention mechanisms 3) encodings from each chunks are concatenated then projected to produce the output encodings, as described with:

$$MultiHeadAttention(Q, K, V) = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{\text{H}}]W^o$$
$$where \quad \boldsymbol{e}_{\text{i}} = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (4)$$

where H is the number of heads, d_{in} is the dimension of input sequence, $d_k = d_{in}/H$, $\boldsymbol{e}_{\text{i}}$ is the encoding generated by head i, $W^o \in \mathcal{R}^{d_{in} \times d_{in}}$, $W_i^Q \in \mathcal{R}^{d_{in} \times d_k}$, $W_i^K \in \mathcal{R}^{d_{in} \times d_k}$ and $W_i^V \in \mathcal{R}^{d_{in} \times d_v}$. Multi-head attention integrates encodings generated from multiple subspaces to higher-dimensional representations [15].

3.2. Transformer Encoder

The Transformer [14] is also a sequence-to-sequence model. The architecture of the Transformer encoder contains three main blocks: 1) attention block, 2) feed-forward block and 3) layer norm [20] as shown in Fig. 2(a). The attention block contains the core multi-head self-attention component. The feed-forward block projects the input dimension d_{in} to another feature space d_{ff} and then back to d_{in} (usually $d_{ff} \geq d_{in}$) for learning feature representation. The final layer normalization and other additional components including layer norm and dropout in the first two blocks are added to stabilize the model training and prevent overfitting. Furthermore, we use VGGNets to incorporate positional information into the Transformer as illustrated in Fig. 2(b). More details are given in section 4.1.

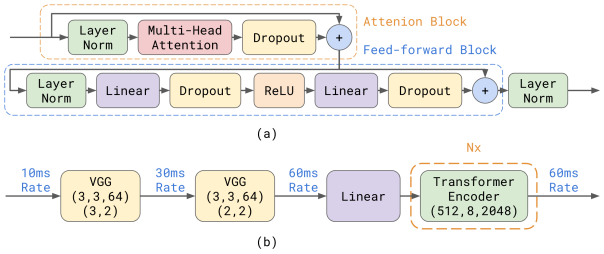

Fig. 2: (a) Transformer Encoder (b) VGG-Transformer.

4. TRANSFORMER-TRANSDUCER

Given the success of the Transformer, we explore the options of applying Transformer in neural transducer. For further improvement, we propose 1) using causal convolution for context modeling and frame rate reduction and 2) using truncated self-attention to reduce the computational complexity and enable streaming for Transformer.

4.1. Context Modeling with Causal Convolution

Transformer relies on multi-head self-attention to model the contextual information. However, attention mechanism is non-recurrent and non-convolutive, therefore risks losing the order or positional information in the input sequence [15, 21], which could harm the performance especially for the case of language modeling. To incorporate the positional information into Transformer, a simple way is adding positional encoding [15] but convolutional approaches [8] demonstrated superior performance. In this paper we adopt the convolutional approach in [8] with modification.

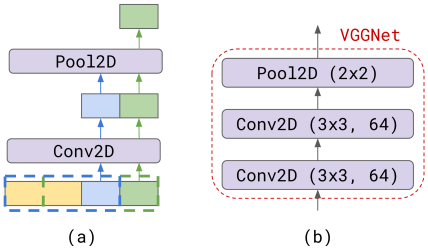

Fig. 3: (a) Causal Convolution (b) VGGNet

Convolution networks model contexts by using kernels to convolve blocks of features. If we treat the input sequence (for example: acoustic features) as a two-dimensional image $X \in \mathcal{R}^{T \times D}$, in common practice for a $N \times K$ kernel the convolution would cover from $X(i - \frac{N-1}{2}, j - \frac{K-1}{2})$ to $X(i + \frac{N-1}{2}, j + \frac{K-1}{2})$ to produce the convolved output $Y(i, j)$. Therefore the convolution would need "future" information to generate the encoding for the current time step. For acoustic modeling this introduces additional look ahead and latency, but introducing future information is impractical for language modeling since the next symbol is unknown during inference.

To prevent future information from leaking into the computation at the current time step, we use causal convolution in which all contexts required are pushed to the history, as illustrated in Fig. 3(a). With causal convolution, for a $N \times K$ kernel the convolution covers from $X(i - N + 1, j - \frac{K-1}{2})$ to $X(i, j + \frac{K-1}{2})$ to produce the convolved output $Y(i, j)$, therefore ensuring the convolution is purely "causal". Similar to [8], we also adopt the VGGNet [17] structure, as illustrated in Fig. 3(b), where two two-dimensional convolution layers are stacked sequentially followed by a two-dimensional max-pooling layer. We use layers of the causal VGGNet to incorporate positional information and propagate to the succeeding Transformer encoder layers. We refer to this network as "VGG-Transformer" and illustrate the architecture used for the encoder in neural transducer in Fig. 3(b), where the first two VGGNet layers are used to incorporate positional information and reduce the frame rate for efficient inference, followed by a linear layer for dimension reduction and multiple Transformer encoder layers for generating higher-level representations.

4.2. Truncated Self-Attention

Unlimited self-attention attends to the whole input sequence and poses two issues: 1) streaming inference is disabled and 2) computational complexity is high. As illustrated in Fig. 4(a), for unlimited self-attention, the output $\boldsymbol{h}_t$ at time step t depends on the entire input sequence $\mathbf{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$, meaning the inference can only begin after the final length T is known. In addition, $\boldsymbol{h}_t$ depends on the similarity pairs $(\boldsymbol{x}_t, \boldsymbol{x}_1), (\boldsymbol{x}_t, \boldsymbol{x}_2) \cdots (\boldsymbol{x}_t, \boldsymbol{x}_T)$, giving complexity $O(T^2)$ for computing $(\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T)$. These issues are critical for self-attention to work in scenarios demanding low-latency and low-computation such as on-device speech recognition [6].

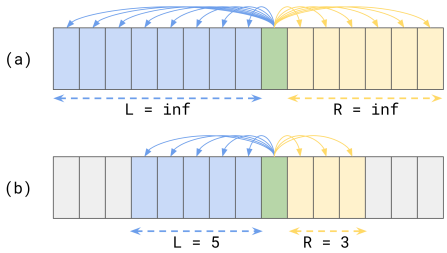

Fig. 4: Self-Attention: (a) Unlimited (b) Truncated.

To reduce both the latency and computational cost, we replace the unlimited self-attention by truncated self-attention, as illustrated in Fig. 4(b). Similar to time-delayed neural network (TDNN) [22, 23], we limit the contexts available for self-attention so that output $\boldsymbol{h}_t$ at time t only depends on $(\boldsymbol{x}_{t-L} \cdots \boldsymbol{x}_{t+R})$. Compared with unlimited self-attention, truncated self-attention is both streamable and computationally efficient. The look-ahead is the right context R and the computational complexity reduces from $O(T^2)$ to $O(T)$. However, it also comes with potential performance degradation and is investigated further in experiments.

5. EXPERIMENTS

5.1. Corpus and Setup

We use the publicly-available, widely-used LibriSpeech corpus [24] for experiments. LibriSpeech comes with 960 hours of read speech data for training, and 4 sets {`dev`, `test`}-{`clean`,`other`} for fine-tuning and evaluations. The `clean` sets contain high quality utterances where as the `other` sets are more acoustically challenging. We use `dev`-{`clean`,`other`} sets to fine-tune parameters for beam search and report results on `test`-{`clean`,`other`} results. We extract 80-dimensional log Mel-filter bank features every 10ms as acoustic features and normalize them with global mean computed from the training set. We also apply SpecAugment [25] with policy "LD" for data distortion. A sentence piece model [26] with 256 symbols is trained from transcriptions of the training set and serves as the output symbols. For each model, we use a learnable embedding layer to convert symbols to 128-dimensional vectors just before the predictor. The experiments are done using PyTorch [27] and Fairseq [28] All models are trained on 32 GPUs with distributed data parallel (DDP) mechanism. We use standard beam search with beam size of 10 for decoding. The decoded sentence pieces are then concatenated into hypotheses to be compared with ground truth transcription for word error rate (WER) evaluation.

5.2. Model Architectures and Details

We compare architectures with roughly the same number of parameters in total. For the encoder in neural transducer, we evaluate options including 1) `BLSTM 4x640`: bidirectional LSTM with 4 layers of 640 hidden units in each direction, 2) `LSTM 5x1024`: LSTM with 5 layers of 1024 hidden units and 3) `Transformer 12x`: VGG-Transformer with 2 layers of VGGNets and 12 Transformer encoder layers. Each VGGNet layer contain 2 layers of two-dimension convolution of 64 kernels of size 3x3. Each Transformer

encoder layer takes 512-dimensional inputs, with 8 heads for multi-head self-attention and 2048 as the feed-forward dimension. For efficient inference, all encoders generate output encodings every 60ms. For LSTM/BLSTM this is achieved with low frame rate [29] in which every three consecutive frames are stacked and subsampled to form the new frame, and apply subsampling of factor 2 to the output of the second LSTM/BLSTM layer [6]. For VGG-Transformer we set the max-pooling on time dimension to 3 for the 1st VGGNet and 2 for the 2nd VGGNet, as illustrated in Fig. 2(b).

For the predictor in neural transducer, we evaluate options including 1) `LSTM 2x700`: LSTM with 2 layers of 700 hidden units and 2) `Transformer 6x`: VGG-Transformer with 1 layer of VGGNet and 6 Transformer encoder layers. Both the VGGNet layer and the Transformer encoder layers share the same configuration with the the encoder case, with the exception that max-pooling is removed in the VGGNet. In addition, the right context R for these the Transformer encoders is 0 for preventing future information leakage.

For the joiner in neural transducer, outputs from the encoder $\boldsymbol{h}_t$ and the predictor $\boldsymbol{p}_u$ are joined with:

$$\boldsymbol{z}_{t,u} = \phi(\boldsymbol{h}_t W^h + \boldsymbol{p}_u W^p) W^o \tag{5}$$

where $W^h \in \mathcal{R}^{d_h \times d_J}$ and $W^p \in \mathcal{R}^{d_p \times d_J}$ project $\boldsymbol{h}_t$ and $\boldsymbol{p}_u$ to a common feature space of dimension d_J, $\phi()$ is an activation function and $W^o \in \mathcal{R}^{d_J \times d_o}$ generates the logits $\boldsymbol{z}_{t,u}$. We use $d_J = 640$, $\phi = ReLU$ and $d_o = 256$ consistently for all experiments.

5.3. Results on Transformer/LSTM Combinations

We experimented with combinations of Transformer and LSTM networks for neural transducer. The results are summarized in Table 1. For the encoder, we use `LSTM 5x1024` as the streamable baseline, `BLSTM 5x640` as the non-streamable baseline and `Transformer 12x` as the novel replacement for the two. For the predictor, we use `LSTM 2x700` and `Transformer 6x` described in section 5.2 as the two options.

Table 1: Neural Transducer with (B)LSTM / Transformer.

encoder	predictor	# params	test-clean	test-other
(1) LSTM 5x1024	LSTM 2x700	50.5 M	12.31	23.16
(2) BLSTM 4x640	LSTM 2x700	48.3 M	6.85	16.90
(3) Transformer 12x	LSTM 2x700	45.7 M	**6.08**	**13.89**
(4) LSTM 5x1024	Transformer 6x	67.1 M	15.76	26.67
(5) BLSTM 4x640	Transformer 6x	64.9 M	7.20	16.67
(6) Transformer 12x	Transformer 6x	62.3 M	7.11	15.62

From Table 1, given the same configuration for the predictor we see that it is difficult for the LSTM network as encoder to perform well given the constraint on number of parameters. The bidirectional LSTM (BLSTM) network however can compensate the performance and remain compact in size at the cost of being non-streamable. The VGG-Transformer with unlimited self-attention outperforms BLSTM significantly as the encoder and is also non-streamable. For the predictor, for all encoder configurations we see the LSTM network still gives better results than the VGG-Transformer and is smaller in size. As a result we keep `LSTM 2x700` as the predictor for the experiments in section 5.4. It is worth noting that the VGG-Transformer loses the advantage of parallel computation as the predictor, as during beam search the hypothesis also extends a token at one search step.

5.4. Results on Truncated Self-Attention

We evaluated the impact of the contexts (L, R) in truncated self-attention on recognition accuracy for the VGG-Transformer. As summarized in section 5.3, we find the VGG-Transformer performs well as the encoder but not as the predictor. Therefore we keep `LSTM 2x700` as the predictor for the experiments in truncated self-attention. The results are summarized in Table 2, where (L, R) are used for truncated self-attention in the VGG-Transformer per layer and aggregate through layers.

Table 2: Transformer with Truncated Self-Attention.

Model Architecture	L	R	test-clean	test-other
(1) LSTM 5x1024 + LSTM 2x700	inf	0	12.31	23.16
(2) BLSTM 4x640 + LSTM 2x700	inf	inf	6.85	16.90
(3) Transformer 12x + LSTM 2x700	inf	inf	**6.08**	**13.89**
(4) Transformer 12x + LSTM 2x700	inf	0	12.32	23.08
(5) Transformer 12x + LSTM 2x700	inf	1	6.99	16.88
(6) Transformer 12x + LSTM 2x700	inf	2	6.47	15.79
(7) Transformer 12x + LSTM 2x700	inf	4	**6.14**	**14.86**
(8) Transformer 12x + LSTM 2x700	inf	8	5.99	14.17
(9) Transformer 12x + LSTM 2x700	4	4	6.84	17.38
(10) Transformer 12x + LSTM 2x700	8	4	6.69	16.79
(11) Transformer 12x + LSTM 2x700	16	4	6.57	15.92
(12) Transformer 12x + LSTM 2x700	32	4	**6.37**	**15.30**

Since the right context R introduces algorithmic latency and has major impact on the recognition accuracy, to find optimal parameters for truncated self-attention, we search for the right context R first while keeping the left context L unlimited and then reduce the left context L given the selected right context R. From Table 2 we see both L and R have significant impact on the performance, especially when $R = 0$ when the VGG-Transformer becomes purely causal. However, as R increases, the WERs gradually recover and come close to the case of unlimited self-attention when $R = 8$. With limited right context R, the VGG-Transformer becomes streamable but still is $O(T^2)$ in computational complexity due to the unlimited left context L. To keep reasonable performance while minimizing latency at the same time, we selected right context $R = 4$ and evaluate different left contexts L. Similar to right context R, we see the WER is also sensitive to left context L. With $(L, R) = (16, 4)$ we see the VGG-Transformer with truncated self-attention gives better WER than both LSTM/BLSTM baselines. With $(L, R) = (32, 4)$ we only lose 4.7 % on `test-clean` and 10.1 % on `test-other` relatively compared with the case of umlimited self-attention, but the system becomes streamable and efficient with computational complexity $O(T)$.

6. CONCLUSION

In this paper, we explore options for using the Transformer networks in neural transducer for end-to-end speech recognition. The Transformer network uses self-attention for sequence modeling and can compute in parallel. With causal convolution and truncated self-attention, the neural transducer with the proposed VGG-Transformer as the encoder achieved 6.37 % on the `test-clean` set and 15.30 % on the `test-other` set of the public corpus LibriSpeech with a small footprint of 45.7 M parameters for the entire system. The proposed Transformer-Transducer is accurate, streamable, compact and efficient, therefore a promising option for resource-limited scenarios such as on-device speech recognition.

7. REFERENCES

[1] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2011.

[2] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[3] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[4] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

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

[6] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[7] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 193–199.

[8] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer, "Transformers with convolutional context for asr," *arXiv preprint arXiv:1904.11660*, 2019.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[10] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International conference on machine learning*, 2014, pp. 1764–1772.

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

[12] Linhao Dong, Feng Wang, and Bo Xu, "Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5656–5660.

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

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] 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.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

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

[18] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

[19] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[20] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[21] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[22] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang, "Phoneme recognition using time-delay neural networks," *Backpropagation: Theory, Architectures and Applications*, pp. 35–61, 1995.

[23] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[25] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[26] Taku Kudo and John Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[28] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[29] Golan Pundak and Tara N Sainath, "Lower frame rate neural network acoustic models," *Interspeech 2016*, pp. 22–26, 2016.