INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Tied & Reduced RNN-T Decoder

Rami Botros, Tara N. Sainath, Robert David, Emmanuel Guzman, Wei Li, Yanzhang He

Google Inc., U.S.A
{ramibotros, tsainath, lrdx, emmanuelguzman, mweili, yanzhanghe}@google.com

Abstract

Previous works on the Recurrent Neural Network-Transducer (RNN-T) models have shown that, under some conditions, it is possible to simplify its prediction network with little or no loss in recognition accuracy [1, 2, 3]. This is done by limiting the context size of previous labels and/or using a simpler architecture for its layers instead of LSTMs. The benefits of such changes include reduction in model size, faster inference and power savings, which are all useful for on-device applications.

In this work, we study ways to make the RNN-T decoder (prediction network + joint network) smaller and faster without degradation in recognition performance. Our prediction network performs a simple weighted averaging of the input embeddings, and shares its embedding matrix weights with the joint network's output layer (a.k.a. weight tying, commonly used in language modeling [4]). This simple design, when used in conjunction with additional Edit-based Minimum Bayes Risk (EMBR) training, reduces the RNN-T Decoder from 23M parameters to just 2M, without affecting word-error rate (WER).

Index Terms: end-to-end, speech recognition, on-device, limited memory

1. Introduction

Research on end-to-end (E2E) models has produced promising results on various speech recognition tasks [5, 6, 7, 8, 9, 10, 11]. These models attempt to simultaneously learn the acoustic, pronunciation and language models of conventional speech recognition systems using a single neural network. In addition to being simpler to train, they are usually much smaller in size compared to the conventional systems [12], making them suitable for on-device applications [9, 13, 11]. On-device E2E models have the potential to improve privacy and reduce recognition latency, especially if a streaming model, such as RNN-T, is used.

There are research efforts to bring RNN-T to edge devices. Some relevant challenges are a requirement for low latency, and dealing with memory constraints. In particular, if the complete model cannot completely fit in-memory on hardware accelerators, smaller partitions of it need to be loaded and processed sequentially, with each such step having a high fixed cost. Hence, smaller models can potentially lead to immense speedup during inference. To this end, we focus on studying methods that make the RNN-T decoder, which comprises the prediction network and joint network, as small and computationally cheap as possible without sacrificing WER performance.

There have been studies in the literature on reducing the complexity of the prediction network, or of the length of input that it is given. In [1], the input to the prediction network's LSTM is restricted to two history phonemes without observing WER degradation. It is hence suggested that, after training, the LSTM be converted to a fast lookup table of size $|V|^2$, where V is the output vocabulary[1]. Various other works adopting limited-context prediction networks with 3-4 history tokens are [3, 15] and [14], the latter using a mere causal Conv1D layer over the history embeddings, and thus forgoing token-order information[1]. [2] goes further by just conditioning on a single previous label, making the prediction network function as a stateless embedding layer, and observes no degradation for low-resource languages. Nevertheless, relative WER regressions do occur for languages with large amounts of training data.

These works give clear indications that the size and complexity of the RNN-T encoder are more important for performance than those of the prediction network. This is supported by the findings in [16]: When the prediction network is randomly initialized and frozen (not trained), WER never degrades compared to the fully-trained baseline. In contrast, similar freezing of the encoder layers hurts performance significantly.

In this work, we explore some useful changes in architecture for both the prediction and joint networks, that eliminate the performance gap to the full-context LSTM baseline [16, 3], while using a much smaller RNN-T decoder. First, we design our prediction network to be a weighted averaging of the history-token embeddings, where the weights are determined by a multi-headed attention mechanism that only attends to so-called *position vectors*, without any cross-token attention. In addition, we tie the label embedding matrix to the output layer of the joint network, which is analogous to a common practice in LMs [4]. On a Voice Search task, we find that the proposed tied and reduced RNN-T decoder, at 2M parameters, has no loss in accuracy compared to the 23M parameter LSTM baseline decoder, if word-level Edit-based Minimum Bayes Risk (EMBR) training [17][18][19] is utilized. In addition, the proposed architecture is non-recurrent, fast and accelerator-friendly. The proposed decoder improves inference speed up to 3.7x on CPU.

2. RNN-T and Embedding Decoders

The RNN-T is a streaming E2E model [5], illustrated in Figure 1. Its task is to produce a label sequence $\mathbf{y}$ (wordpieces in this work), given a stream of acoustic frames. At each time step t, the model receives a new acoustic frame x_t and outputs a probability distribution over $\hat{y}_t \in V \cup \{<b>\}$, V being the wordpiece vocabulary, and $<b>$ a blank symbol. The blank symbol is needed since the length of the label sequence is usually much shorter than the total number of acoustic frames. Hence, during training, the model attempts to learn to output sequences $\hat{\mathbf{y}}$ that would be identical to $\mathbf{y}$ if all $<b>$ outputs are removed.

RNN-T consists of 3 main parts: an encoder, a prediction network (PN) and a joint network. At each time step, the encoder (here a series of Conformer [20] layers) produces an encoding f_t, while attending to a series of T most recent frames received so far $\mathbf{x}_{t-T+1:t}$. Every time the model outputs a non-

[1] Unlike our work, [1, 14] use phoneme labels instead of wordpieces, and an n-gram LM, as well as a lexicon, during first-pass decoding.

http://dx.doi.org/10.21437/Interspeech.2021-212

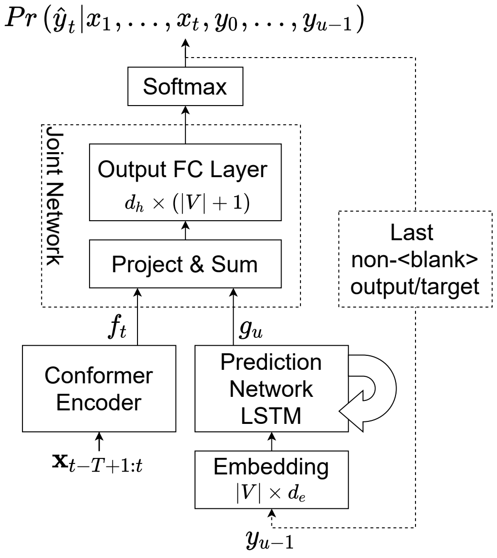

Figure 1: *RNN-T baseline model with some weight dimensions.*

blank token, it is fed back into the PN to be embedded and passed through a series of recurrent LSTM layers. The PN thus produces a representation g_u, which can be conditioned on all previous non-blank outputs $y_0 \ldots y_{u-1}$, and is used for subsequent time steps until the next PN update. Inside the joint network, both f_t and g_u are projected to having the same dimensionality, and their sum goes through a fully-connected layer to produce the output logits.

2.1. Limited-Context RNN-T

In vanilla RNN-T models, the PN is a recurrent model conditioned on all previous non-blank predictions. On the other hand, a limited-history PN is only conditioned on the last N such predictions. Works such that [3, 15] use LSTMs or Transformers with truncated history. Our study adopts limiting the history, but also focuses on ways to replace the PN architecture itself with smaller, non-recurrent and on-device friendly layers.

2.2. Embedding Decoders

[1] suggests that, after training, the PN's LSTM conditioning history can be restricted to a size $N = 2$ without loss in accuracy, and can thus be converted to a lookup table. A variation on this can be to train the lookup table directly. Since that work uses phonemes, size considerations would be different in our case: With thousands of wordpieces, the V^2 lookup table can be in the order 10^8 parameters, i.e. too large for our purposes. One solution is to embed each token with a shared embedding matrix and concatenate the embeddings. We call decoders with such prediction networks Embedding Decoders, since their size and complexity are dominated by the embedding operation.

3. Reduced Embedding Decoders

A downside of the Embedding Decoders from Section 2.2 is that, for longer histories N, concatenated embeddings require larger subsequent projection layers. Moreover, a mere embedding operation might not be expressive enough, esp. for small embedding dimensions. We design a more expressive layer that also cut down overall model size.

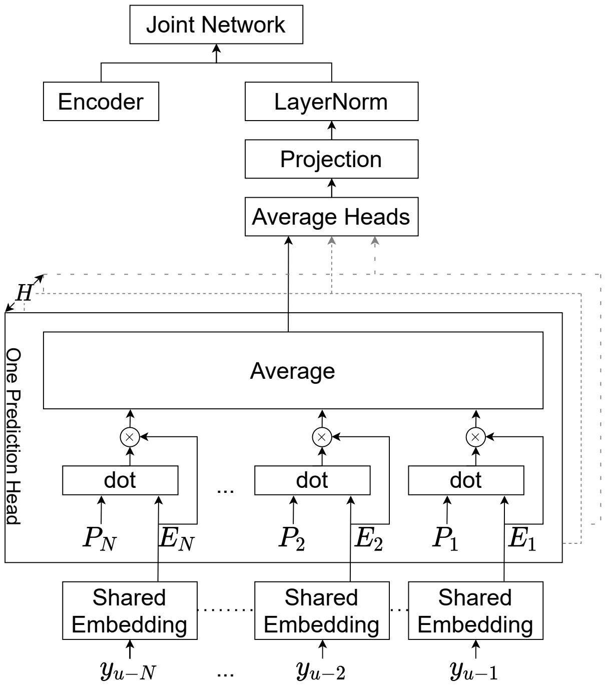

Figure 2: *Proposed prediction network with* N *previous tokens.*

3.1. Prediction Network Design

Figure 2 illustrates our prediction network. It uses a shared embedding matrix to embed each previous label y_{u-i} into an embedding vector E_i. The aim of our design is to average rather than concatenate the embeddings, hence the name *reduced*, and get a slimmer output. In order to still retain information about token order, we use so-called position vectors P: For each position in the history of our N previous labels, we create a P with the same size as the embedding. These are independent vectors, that can either be learned or set to some constant values. The output of our PN is the weighted average of all embeddings, where the averaging weight for each embedding is the dot product between itself and its position's P. This can be interpreted as attending over each embedding in proportion to its relevancy to the P of its respective position. Thus, for

$$\text{Embeddings } \mathbf{E} \in \mathbb{R}^{N \times d_e}, \text{ Position Vectors } \mathbf{P} \in \mathbb{R}^{N \times d_e},$$

$$\text{Prediction}(\mathbf{E}, \mathbf{P}) = \frac{1}{N} \sum_{n} \left[\mathbf{E}_n \cdot \sum_{e} E_{n,e} \cdot P_{n,e} \right],$$

where n and e respectively index the history positions and the embedding dimension. The interaction with $\mathbf{P}$ can potentially preserve position information, without needing recurrence or cross-token attention. Early experiments show that $\mathbf{P}$ don't need to be trainable, so we fix them to random values.

Inspired by [21], we find that expanding this to a multi-headed design improves performance significantly. Each head has its own set of position vectors, but utilizes the same shared embedding matrix, costing just a small parameter increase overall. We average the outputs from all heads. For H heads, indexed by h, and $\mathbf{P}' \in \mathbb{R}^{H \times N \times d_e}$, we get

$$\text{MultiHeadPred}(\mathbf{E}, \mathbf{P}') = \frac{1}{H \cdot N} \sum_{h,n} \left[\mathbf{E}_n \cdot \sum_{e} E_{n,e} \cdot P'_{h,n,e} \right]$$

To further improve model expressiveness, we add a cheap projection layer, stabilized with LayerNorm [22], followed by a Swish non-linearity [23]. We apply those to the averaging output before passing it to the joint network, as shown in Figure 2. The extra projection layer does not alter the output dimension.

Overall, we get a PN with an output dimension equal to one embedding vector, achieving a slim architecture for the whole decoder. It remains small for any context size: For any increase in N, we just need some additional P vectors. This has enabled us to experiment with longer histories $N > 2$.

4. RNN-T Decoder with Tied Embeddings

The works cited above have optimized the size of the prediction network without reducing the size of the joint network, which has a constant cost of 2-3M parameters. To further reduce the RNN-T decoder as a whole, we explore parameter sharing between the joint and the PN. As shown in Figure 1, the dimensions of the PN's embedding matrix are $|V| \times d_e$. Meanwhile, if the joint network's last hidden layer is of size d_h, the feed-forward projection weights from there to the output logits will have a shape of $d_h \times |V + 1|$, with 1 extra output for the blank token. If we restrict our model to have $d_e = d_h$, these two matrices can share their weights for all non-blank tokens, after utilizing a simple transpose transformation. This is analogous to the weight tying practice in LMs [4], and saves $d_h \cdot |V|$ parameters, while potentially regularizing the model.

5. Experimental Details

5.1. Data Sets

The training set used for experiments is the same data from [24]. The multi-domain data covers domains of search, farfield, telephony and YouTube. All datasets are anonymized and hand-transcribed; the transcription for YouTube utterances is done in a semi-supervised fashion [25][26]. To increase data diversity, we augment our dataset with multi-condition training (MTR) [27] and random 8kHz down-sampling [28].

The test set is also anonymized, hand-transcribed, and is representative of Google's Voice Search traffic. It consists of around 12K utterances with an average duration of 5.5 seconds. For long-form test sets, we use the YouTube and the TTS-Audiobook test sets from [24]. They have an average utterance length of 319 and 66 seconds, respectively.

5.2. RNN-T Model Specifications

All experiments use 128-dimensional log-Mel features, computed with a 32ms window and shifted every 10ms. Similar to [24], features for each frame are stacked with 3 frames to the left and then downsampled by three to a 30ms frame rate. Our model is similar to the one described in [29]. Thus, we use a 12-layer causal Conformer encoder [20] with 113M parameters.

We apply Specaugment [30] in the manner described in [31]. The model predicts 4,096 wordpiece units, including the end-of-sentence token. Training is done on 8x8 Cloud TPU using the Tensorflow Lingvo toolkit [32]. Finally, the 1st-pass output is rescored with a maximum-entropy LM [33] in the second pass. When used, EMBR training is performed after normal training, for around 1⁄10 the number of training steps .

5.3. RNN-T Decoders in Comparison

Table 1 gives some details about the different RNN-T decoders that we use in our experiments. The Size column shows the number of parameter for the prediction and joint networks put together. The LSTM baseline is the largest and only recurrent model. It has 2 LSTM layers with 2,048 hidden units and a 640-dimensional projection per layer. Stateless1Emb is the simplest Embedding Decoder, from [2], where the PN is a mere embedding of one token. Concat2Emb is the Embedding Decoder from Section 2.2. Those 3 baselines are compared to our proposed decoders ReducedLarge and ReducedSmall, which only differ in embedding dimension d_e and length of conditioning history N. Both have number of heads $H = 4$.

Table 1: *Specifications for RNN-T decoders in our experiments.*

Decoder Name	Embedding Dimension (d_e)	History Length (N)	Size
LSTM	128	∞	23M
Stateless1Emb	640	1	6.0M
Concat2Emb	640	2	6.4M
ReducedLarge	1280	2	9.2M
ReducedSmall	320	5	**1.9M**

6. Results

6.1. Result Overview

Table 2 compares the pre- and post-EMBR WER for all decoders. We confirm that the Concat2Emb, inspired by [1], has competitive performance, including when the V^2 lookup table is trained directly. Stateless1Emb is significantly worse, but gains a substantial improvement after EMBR training. Our larger setup, ReducedLarge reaches the baseline WER of 6.1%, even without EMBR. Our small setup, ReducedSmall with 1.9M parameters, benefits from an expansion of history size to 5 and reaches the baseline LSTM WER after EMBR. Below, a series of ablation studies show how different parameter choices effect the performance of the reduced models.

Table 2: *WER for baselines and our Reduced models.*

Decoder Name	Size	Pre-EMBR WER	Post-EMBR WER
LSTM	23M	**6.1%**	**6.1%**
Stateless1Emb	6.0M	6.6%	6.2%
Concat2Emb	6.4M	6.2%	6.2%
ReducedLarge	9.2M	**6.1%**	**6.1%**
ReducedSmall	**1.9M**	6.4%	**6.1%**

6.2. Ablation Studies

6.2.1. *Effect of Tying and Decoder Size*

Figure 3 shows how pre-EMBR WER varies for our reduced decoder with different sizes, with and without the tied embeddings from Section 4. The sweep over model size is done by varying the embedding dimension d_e. Note that, in the non-tied case, the last hidden layer of the joint network is always set to be $d_h = 640$. In the tied case, $d_h = d_e$, so the overall model size varies more with d_e. Moreover, however small d_e becomes, the non-tied decoder will always have a size above 2.5M.

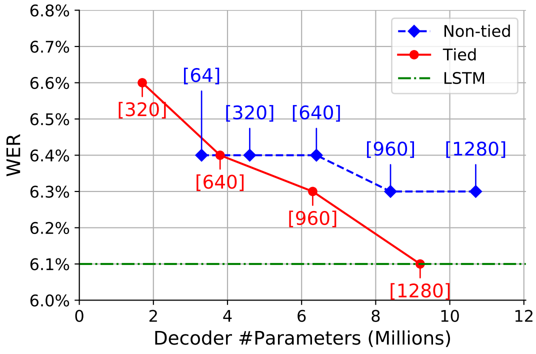

Figure 3: *Pre-EMBR WER vs. size, for tied vs. non-tied reduced decoders. Embedding dimension* (d_e) *are shown. All models here have 4 heads (*H*) and 2 history tokens (*N*).*

The results indicate that, starting from a decoder size of around 4M parameters, the weight-tying technique achieves better performance per model size. Moreover, for large-enough models, it reaches the same performance as the LSTM decoder, which the non-tied model never does. This might be a regularization effect. Below 4M parameters, the tied model rapidly loses performance. In the next subsection, we show how this can be mitigated by increasing the number of history tokens N.

6.2.2. *Varying The Number of History Tokens*

Table 3 shows the pre-EMBR WER for `ReducedSmall` with different history lengths N, which hardly changes model size. Extending N to 5 seems to offset the model's limited capacity.

Table 3: *Pre-EMBR WER vs. number of history tokens (*N*) on* `ReducedSmall`

History length (N)	3	4	**5**	6
WER	6.6%	6.5%	**6.4%**	6.4%

6.2.3. *Varying Number of heads*

Table 4 shows Pre-EMBR WER when varying the number of heads for `ReducedLarge`. We observe that the extra heads seem to successfully augment our model with specialized position vectors. We choose $H = 4$, since that works best with our other reduced models and is in agreement with the number of heads used in the encoder-decoder architecture in [8].

Table 4: *Pre-EMBR WER vs. #heads (*H*) on* `ReducedLarge`

Heads	1	2	3	**4**	5	6
WER	6.2%	6.2%	6.1%	**6.1%**	6.1%	6.3%

6.3. Long-Form Testsets

Now that the parameters for `ReducedSmall` have been better understood, we take a closer look on how the various decoders perform on long-utterance test sets, post EMBR, as shown in Table 5. All Embedding Decoders perform significantly better than the `LSTM` baseline with its unbound context. Our proposed `ReducedSmall` decoder gives a further performance improvement over the baseline Embedding Decoders.

The LSTM has a long recurrent state dependency, and might be suffering from state saturation with the long utterances [34]. While the `Stateless1Emb` and `Concat2Emb` decoders do not have this recurrency problem, they have a limited history of 1-2 tokens and do not scale gracefully in size with a larger history like `ReducedSmall`.

Table 5: *Post-EMBR WER on long-utterance test sets*

Decoder Name	Audiobook WER	YouTube WER
`LSTM`	5.0 %	10.5%
`Stateless1Emb`	4.5%	10.3%
`Concat2Emb`	4.4%	10.3%
`ReducedSmall`	**4.1%**	**10.1%**

6.4. Speedup measurement

Table 6 compares the runtime statistics for the `LSTM` and `ReducedSmall` decoders on Pixel 4's Kyro 485 CPUs. We use 100 utterances (2.5 ± 1 s), running each 100 times. Our smaller decoder achieves 2.7-3.7x inference speedup.

Table 6: *Average Runtime Comparison for 10K runs*

CPU Core	`LSTM`	`ReducedSmall`
A55, 1.78 GHz	19.2 ± 0.36 ms	**5.21 ± 0.15 ms**
A76, 2.42 GHz	2.23 ± 0.65 ms	**0.83 ± 0.43 ms**
A76, 2.84 GHz	2.03 ± 0.95 ms	**0.66 ± 0.03 ms**

7. Conclusions

In this paper, we presented a design for a small, fast RNN-T decoder. Reducing the vanilla RNN-T decoder size by around 90%, our 2M-parameter design is 3-4 times faster, and can completely fit in-memory on many on-device accelerator chips. We found that EMBR training is highly beneficial for small Embedding Decoders, helping our smallest model match or surpass baseline WER. Overall, our work presents further evidence that the size of the encoder is more important than that of the prediction network. We also demonstrated that the common embedding-to-output weight sharing from language modeling is useful for RNN-T. It saved millions of parameters and achieved better WER than our best non-tied models. Future research can investigate whether the technique can also improve RNN-T with complex LSTM and/or Transformer decoders. We also recommend utilizing smaller decoders for complex beam search to further improve WER. Another direction enabled by our work is delivering multiple prediction networks on device, perhaps for model specialization or for multilingual models.

8. Acknowledgements

The authors thank Trevor Strohman, Ehsan Variani and Shankar Kumar for helpful discussions on embedding decoders.

9. References

[1] E. Variani, D. Rybach, C. Allauzen, and M. Riley, “Hybrid Autoregressive Transducer (HAT),” in *Proc. ICASSP*, 2020.

[2] Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein, “Rnn-transducer with stateless prediction network,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7049–7053.

[3] Rohit Prabhavalkar, Yanzhang He, David Rybach, Sean Campbell, Arun Narayanan, Trevor Strohman, and Tara N Sainath, “Less is more: Improved rnn-t decoding using limited label context and path merging,” *arXiv preprint arXiv:2012.06749*, 2020.

[4] Hakan Inan, Khashayar Khosravi, and Richard Socher, “Tying word vectors and word classifiers: A loss framework for language modeling,” *arXiv preprint arXiv:1611.01462*, 2016.

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

[6] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals, “Listen, attend and spell,” *arXiv preprint arXiv:1508.01211*, 2015.

[7] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[8] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., “State-of-the-art speech recognition with sequence-to-sequence models,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[9] 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.

[10] Jinyu Li, Yu Wu, Yashesh Gaur, Chengyi Wang, Rui Zhao, and Shujie Liu, “On the comparison of popular end-to-end models for large scale speech recognition,” *arXiv preprint arXiv:2005.14327*, 2020.

[11] Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al., “A streaming on-device end-to-end model surpassing server-side conventional model quality and latency,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6059–6063.

[12] Golan Pundak and Tara Sainath, “Lower frame rate neural network acoustic models,” 2016.

[13] Jinyu Li, Rui Zhao, Hu Hu, and Yifan Gong, “Improving rnn transducer modeling for end-to-end speech recognition,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 114–121.

[14] Yuekai Zhang, Sining Sun, and Long Ma, “Tiny transducer: A highly-efficient speech recognition model on edge devices,” *arXiv preprint arXiv:2101.06856*, 2021.

[15] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, “Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

[16] Harsh Shrivastava, Ankush Garg, Yuan Cao, Yu Zhang, and Tara Sainath, “Echo state speech recognition,” *arXiv preprint arXiv:2102.09114*, 2021.

[17] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C. C. Chiu, and A. Kannan, “Minimum Word Error Rate Training for Attention-based Sequence-to-sequence Models,” in *Proc. ICASSP*, 2018.

[18] Chao Weng, Chengzhu Yu, Jia Cui, Chunlei Zhang, and Dong Yu, “Minimum bayes risk training of rnn-transducer for end-to-end speech recognition,” *arXiv preprint arXiv:1911.12487*, 2019.

[19] Jinxi Guo, Gautam Tiwari, Jasha Droppo, Maarten Van Segbroeck, Che-Wei Huang, Andreas Stolcke, and Roland Maas, “Efficient minimum word error rate training of rnn-transducer for end-to-end speech recognition,” *arXiv preprint arXiv:2007.13802*, 2020.

[20] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.

[21] 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.

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

[23] Prajit Ramachandran, Barret Zoph, and Quoc V Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.

[24] Arun Narayanan, Rohit Prabhavalkar, Chung-Cheng Chiu, David Rybach, Tara N Sainath, and Trevor Strohman, “Recognizing long-form speech using streaming end-to-end models,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 920–927.

[25] Hank Liao, Erik McDermott, and Andrew Senior, “Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription,” in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 368–373.

[26] Hagen Soltau, Hank Liao, and Hasim Sak, “Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition,” *arXiv preprint arXiv:1610.09975*, 2016.

[27] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara Sainath, and Michiel Bacchiani, “Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home,” 2017.

[28] Jinyu Li, Dong Yu, Jui-Ting Huang, and Yifan Gong, “Improving wideband speech recognition using mixed-bandwidth training data in cd-dnn-hmm,” in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 131–136.

[29] B. Li, A. Gulati, J. Yu, et al., “A Better and Faster End-to-End Model for Streaming ASR,” in *Proc. ICASSP*, 2021.

[30] 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.

[31] Daniel S Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V Le, and Yonghui Wu, “Specaugment on large scale datasets,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6879–6883.

[32] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al., “Lingvo: a modular and scalable framework for sequence-to-sequence modeling,” *arXiv preprint arXiv:1902.08295*, 2019.

[33] Fadi Biadsy, Mohammadreza Ghodsi, and Diamantino Caseiro, “Effectively building tera scale maxent language models incorporating non-linguistic signals,” 2017.

[34] Alex Graves, “Supervised sequence labelling,” in *Supervised sequence labelling with recurrent neural networks*, pp. 5–13. Springer, 2012.