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

An Efficient Streaming Non-Recurrent On-Device End-to-End Model with Improvements to Rare-Word Modeling

Tara N. Sainath[*], *Yanzhang He*[*], *Arun Narayanan*[*], *Rami Botros, Ruoming Pang, David Rybach, Cyril Allauzen, Ehsan Variani, James Qin, Quoc-Nam Le-The, Shuo-Yiin Chang, Bo Li, Anmol Gulati, Jiahui Yu, Chung-Cheng Chiu, Diamantino Caseiro, Wei Li, Qiao Liang, Pat Rondon*

Google Inc., U.S.A
{tsainath, yanzhanghe, arunnt}@google.com

Abstract

On-device end-to-end (E2E) models have shown improvements over a conventional model on Search test sets in both quality, as measured by Word Error Rate (WER) [1], and latency [2], measured by the time the result is finalized after the user stops speaking. However, the E2E model is trained on a small fraction of audio-text pairs compared to the 100 billion text utterances that a conventional language model (LM) is trained with. Thus E2E models perform poorly on rare words and phrases. In this paper, building upon the two-pass streaming Cascaded Encoder E2E model [3], we explore using a Hybrid Autoregressive Transducer (HAT) [4] factorization to better integrate an on-device neural LM trained on text-only data. Furthermore, to further improve decoder latency we introduce a non-recurrent embedding decoder, in place of the typical LSTM decoder, into the Cascaded Encoder model. Overall, we present a streaming on-device model that incorporates an external neural LM and outperforms the conventional model in both search and rare-word quality, as well as latency, and is 318X smaller.

Index Terms: on-device end-to-end models

1. Introduction

End-to-end (E2E) models have become a popular technique to replace the acoustic, pronunciation and language models of a conventional ASR system [5] with a neural-network. Over the past few years, developing an on-device E2E model that can achieve quality and latency comparable to a conventional cloud-based ASR model [5] has become an active area of research [6, 7, 8, 9, 10, 11] across many research groups.

On the quality side, in prior work, we found that a 1st-pass RNN-T model [1] and a 2nd-pass LAS rescoring model [12] allowed E2E to be on par with conventional on both general search and proper noun test sets using contextual biasing [13]. However, thus far, in the absence of contextual biasing, we have not been able to achieve similar performance to conventional when tested with rare words and phrases. This is a challenging problem since E2E models are trained with a small fraction of audio-text pairs compared to the 100-billion text-only utterances that a conventional model is trained with. In addition, the over-confident peaky posteriors of E2E models [14] and constrained beam size during decoding exacerbates this problem.

On the latency side, our goal has been to have an E2E model that can display words on the screen as they are spoken with minimum latency [15] and can endpoint quicker [16]. Recently, we proposed a system [2] that uses FastEmit [15] to encourage the model to emit words and endpoint quickly. This model was shown to achieve these latency goals, while still maintaining voice search and biasing quality to be on-par or better than the conventional model.

In this paper, we address one of the remaining challenges of on-device E2E models compared to conventional models: performance on rare words and phrases. Our baseline system builds on our previous works [7, 1, 6]. Specifically, we use a 2-pass cascaded encoder model [3] trained with FastEmit [15]. To improve performance on the rare words, we first introduce a Hybrid Autoregressive Transducer (HAT) [4] factorization into the model, which allows us to better incorporate an external language model. Next, we train a conformer [17] language model on text utterances, and integrate this with the cascaded encoder model as a rescorer to minimize on-device computation. Finally, to further minimize on-device computation, we replace the LSTM layers, commonly used in transducer prediction networks, with a simple look-up embedding table which reduces decoder parameters by 12X.

2. Modeling Improvements

2.1. Base Architecture: Cascaded Encoders

As first presented in [3], the baseline model we consider in this work is a *Cascaded Encoders* model, as shown in Figure 1. In this model, the 1st-pass system consists of a causal encoder followed by an RNN-T decoder (prediction network + joint layer). In the second pass, the additional non-causal layers take in both left and right context of the 1st-pass encoder outputs, and are fed to the same decoder. A single RNN-T decoder is shared between the 1st and 2nd-passes for better long-form performance, and smaller model size and on-device benefits. Improvements to this model will be discussed below.

2.2. Embedding Prediction Network

The broad success of neural networks in machine learning has brought customized hardware, both in servers and in consumer devices, that can process these networks efficiently. However, these devices are less efficient at processing auto-regressive or recurrent models. Therefore, in this work we explore making the auto-regressive prediction + joint layers in Figure 1 as small and efficient as possible. Specifically, we replace the LSTM prediction network with an embedding lookup table, and share the parameters of the embedding table with the softmax layer. Below we briefly describe the embedding network, shown in Figure 2, and refer the reader to [18] for more details.

With the embedding network, given the 5 previous non-

[*]Equal Contribution. Additional authors: Robert David, Alex Gruenstein, Emmanuel Guzman, W. Ronny Huang, Sepand Mavandadi, Ian McGraw, Charles Peyser, Michael Riley, Trevor Strohman, Yonghui Wu, Yu Zhang.

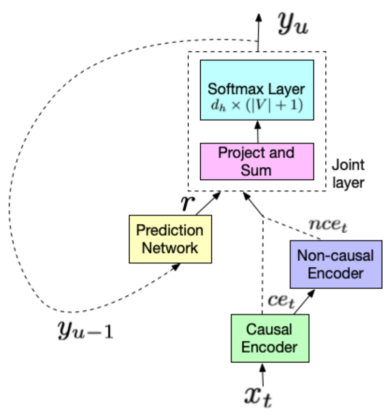
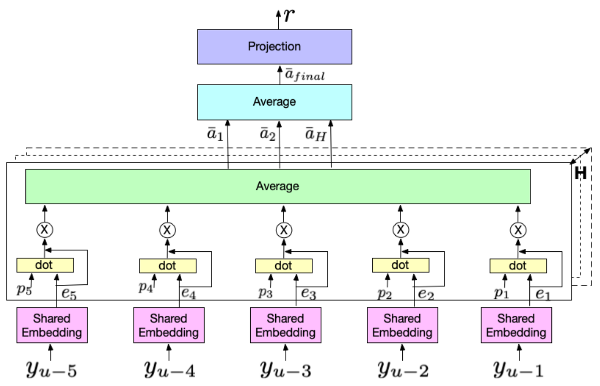


Figure 1: *A block diagram of the Cascaded Encoders model.*



r
Projection
$\bar{a}_{final}$
Average
$\bar{a}_1$
$\bar{a}_2$
$\bar{a}_H$
H
Average
dot
p_5 e_5 p_4 e_4 p_3 e_3 p_2 e_2 p_1 e_1
Shared Embedding
y_{u-5} y_{u-4} y_{u-3} y_{u-2} y_{u-1}

Figure 2: *Block diagram of Embedding Prediction Network.*

blank predictions from the model, which we denote as $Y_{prev} = \{y_{u-5}, \ldots, y_{u-1}\}$, we look up the embedding of each of these outputs, denoted as $\{e_5, \ldots, e_1\}$. The embedding table is of size $|V| \times d_e$, where $|V|$ is the vocabulary size (i.e. wordpieces) and d_e is the embedding dimension. Thus each e_i is of size d_e. For each position in the label history Y_{prev}, we create a random, constant position vector $PV = \{p_5, \ldots, p_1\}$ and take a dot product of each p_i with embedding e_i, and average this, giving output $\bar{a}_1$. The intuition for the PV is that it allows for embeddings to be attended over in proportion to how relevant they are to their respective positions. Motivated by [19], the operation from embeddings $\{e_5, \ldots, e_1\}$ to embedding averaging $\bar{a}_1$ is replicated across multiple heads H, which improves performance at the cost of a small parameter increase. Each head shares the same embedding table, but has unique PVs. The embeddings across each head are also averaged together to get the final embedding vector $\bar{a}_{final} = \frac{1}{H}\sum_i^H \bar{a}_i$.

Neural networks can model a more complex function with additional linear-layer + non-linearity. Since the embedding layer is a simple lookup table, we improved modeling expressiveness, again at the cost of a small parameter increase, by passing the final averaged embedding vector, $\bar{a}_{final}$, through a projection layer and Swish non-linearity, to get output r.

Finally, the joint layer, as shown in Figure 1, takes the output of the prediction network, r, and encoder network, ce_t (causal) or nce_t (non-causal), projects this to dimension d_h and then passes it to a softmax layer, which is of size $d_h \times (|V|+1)$, with the extra output for the blank token. By setting $d_e = d_h$, we share the weights between the embedding and softmax matrices for all non-blank tokens, to further reduce decoder size. This is similar to weight tying used in language modeling [20].

2.3. HAT Factorization

An E2E model directly estimates the posterior of word sequence, y, conditioned on the acoustic features, x, i.e., $p(y|x)$. A common way to integrate an E2E model with an external LM that estimates $p_{LM}(y)$ is to perform shallow fusion (Eq 1).

$$y^* = \arg\max_y [\log p(y|x) + \lambda_1 \log p_{LM}(y)] \tag{1}$$

Here, λ_1 is the weight given to the LM. Since E2E models tend to be over-confident in their posterior estimates, weighting the two terms becomes difficult and can often lead to high deletions [14]. To address this issue, Hybrid Autoregressive Transducer (HAT) proposes a way to factor out an internal LM score, $p_{ILM}(y)$, so that the likelihood score from the E2E model, $p(x|y)$, can be approximated by Equation 2:

$$\log p(x|y) \approx \log p(y|x) - \log p_{ILM}(y) \tag{2}$$

The HAT factorization makes it mathematically more stable to then integrate an E2E model with an external LM, without needing additional coverage penalties:

$$y^* = \arg\max_y [\log p(y|x) - \lambda_2 \log p_{ILM}(y) + \lambda_1 \log p_{LM}(y)] \tag{3}$$

Here, λ_1 and λ_2 are the weights on the LM and ILM, respectively. We train the cascaded encoder with a HAT factorization to allow for better integration with a neural LM and biasing LM.

2.4. Neural Language Model

Our large-scale text-data corpus has ~100 billion utterances, and spans domains of Maps, News, Google Play, Web and YouTube. In contrast, the supervised audio-text pairs used to train the E2E model is much smaller (300 million) and includes domains such as Search, Dictation, YouTube and Telephony [21]. Challenges related to domain-mismatch and data scale make it difficult to train neural LMs using both these datasets.

N-gram LMs attempt to handle the scale and domain-mismatch of the data by training a separate model on each individual domain and then performing a Bayesian interpolation [22] to optimize performance on a target domain of interest. Maximum entropy training, used to train such models, incorporates various optimization strategies to speed up model training, resulting in training on billions of examples taking less than one day [23]. Furthermore, to address domain-mismatch, [24] proposes fine-tuning just on the supervised set after pre-training on all the data. In this work, we address these challenges by simply mixing the LM data and transcripts form the supervised training data with equal probability.

Because of the on-device latency limitations, we explore using a conformer LM to *rescore* hypotheses from the 2nd-pass non-causal decoder. One of the benefits of transformer/conformer LMs is that there is no self-recurrence in the model. This means that model outputs can be computed for multiple inputs in parallel. This is particularly effective in rescoring when the inputs to the LM is known ahead of time. Transformer LMs have shown promising results in the literature [25, 26, 27]. In this work, given our promising results with conformer encoder [17], we explore using a unidirectional conformer LM, where we only attend on previous tokens.

2.5. Biasing

Contextual biasing [28] increases the likelihood of the ASR system recognizing contextually-relevant phrases, such as names from the user's contact list. The typical approach for biasing is to interpolate scores from the E2E model $\log p(y|x)$ and a contextual LM $\log p_C(y)$, represented by a weighted finite-state transducer (FST), via shallow fusion *during* the beam search [13]. We explore how biasing performance is improved with HAT, as shown in Equation 4. When contextual biasing is applied during beam search to the causal and non-causal decoders, the weight of the neural LM $\lambda_1 = 0$. Note that $\lambda_1 \neq 0$ during rescoring with the neural LM, and the contextual LM is reapplied so as to not undo the effects of biasing when applied during beam search. β is the cost on the contextual LM.

$$y^* = \arg\max_y [\log p(y|x) - \lambda_2 \log p_{ILM}(y) + \lambda_1 \log p_{LM}(y) + \beta \log p_C(y)] \tag{4}$$

2.6. Latency

To decode the cascade model in the streaming system, we run the 1st-pass *causal decoder* and the 2nd-pass *non-causal decoder* in parallel. The causal decoder is faster while the non-causal decoder is more accurate. Hence, to reduce latency, the causal decoder is responsible for emitting the partial recognition results [15] and endpointing [16] quickly. Once the endpointing decision is declared, the system then closes the microphone and sends the final recognition result decoded thus far by the non-causal decoder to the server.

An additional way to reduce latency is to fetch results even before the final recognition is ready, which is referred to as prefetching [29]. If the partial result matches the final recognition result, the response fetched for the partial result can be delivered to the user instantly and save the execution latency that typically happens after recognition is completed.

In the cascade model, partial results and final results are emitted by separate decoders. The mismatch can increase prefetch miss, which makes the prefetched results unusable, thereby increasing latency. To reduce the mismatch, we propose to prefetch results based on the combination of both decoders, referred to as *hybrid prefetching*. To merge the prefetch decision from the causal and non-casual decoders, the system first generates prefetch candidates from both decoders independently and then discards the results if they are stale. A prefetch candidate is marked as stale if the top hypotheses is a substring of the latest valid prefetch. For example, "navigate" is a stale result as compared to "navigate to home" so that prefetch is not overridden by the late hypothesis. In the hybrid scenario, the quicker prefetch by the causal decoder would dominate while more accurate prefetch by the non-causal decoder could still revert incorrect prefetching for corner cases.

3. Experiments

3.1. Datasets

Similar to [1], all E2E models are trained on multidomain audio-text pairs [21]. All domains are anonymized and hand-transcribed, except for YouTube where the transcription is done in a semi-supervised fashion [30]. In addition to the diverse training sets, multi-condition training (MTR) [31], random data down-sampling to 8kHz [32] and SpecAug [33] are also used to further increase data diversity.

The text-only data, used to train the conformer language model, consists of more than 100B utterances across domains described in Section 2.4. Since the LM data spans across domains different than the multidomain data, each mini-batch in training uses 50% multi-domain and 50% text-only data, so as to not degrade Search quality. This is standard practice done for both n-gram [22] and maximum-entropy [24] models.

The *Search* test set includes around 12K Voice Search utterances with an average length of 5.5 seconds. They are anonymized and hand-transcribed, and are representative of Google's Voice Search traffic. In addition, to measure the tail, we create a synthetic *Rare-word* test set, described in more detail in [34]. Specifically, we look for words in the LM training data that are rare (i.e., occurs less than 5 times) in the multi-domain data. In addition, we also look at words that have surprising pronunciations given their spellings, with rarity measured as those words with a low grapheme-to-phoneme (G2P) score. Words are selected across all 5 LM domains, and overall we synthesize [35] around 19K utterances to create a rare-word set.

3.2. Modeling

The cascaded encoder model architecture is very similar to [2]. Specifically, all models are trained on 400k hours of data, a 128D log-mel feature frontend with a 16-D one-hot domain-id vector appended to it [21]. Causal convolution and left-context attention layers are used for the Conformer layer to strictly restrict the model to use no future inputs. 8-head attention is used in the self-attention layer and the Convolution kernel size used is 15. The encoder consists of 12 Conformer layers. Five cascaded encoder layers process input 900 milliseconds into the future, and this feeds into the decoder.

The RNN-T decoder consists of a prediction network and a joint network with a single feed-forward layer with 640 units. The baseline system is with 2 LSTM layers with 2,048 units projected down to 640 output units (23.4M params). The embedding prediction network [18] from Section 2.2, uses an embedding dimension of 320, and has 1.96M parameters, $\sim$12X smaller than the LSTM. The baseline cascaded models do not use HAT factorization [2]. All remaining E2E models are trained with the HAT factorization to predict 4,096 word pieces [36]. The cascaded encoder with embedding decoder is 0.15G in model size.

The conformer LM, at 0.13G, has a look back attention context of 31 left tokens for each output wordpiece model we want to predict. The LM is 12 layers, where each layer has a model dimension of 768 and a feedforward layer dim of 2048. Overall the number of attention heads is 6. During inference, the conformer LM is used to rescore the lattice after the non-causal CascEnc. HAT LM weights in Equation 4 (λ_1, λ_2, β) are optimized to minimize WER on small subset of the language model, search and biasing test data.

The conventional model is a low-frame-rate [5] LSTM acoustic model (0.1GB) trained on 20k hours of data, to predict context-dependent phonemes [37]. It uses a phonetic lexicon (2.2GB), a 4.9GB 1st-pass FST LM [22], and a 2nd-pass 82GB Maximum Entropy rescoring LM [23].

4. Results

4.1. Modeling Improvements

First, Table 1 shows that the embedding decoder (E0) does not degrade performance over the LSTM decoder (B1) on the

Search set, while being 12X smaller. Next, exploring LM integration, E1 without HAT (i.e. setting $\lambda_2 = 0$ in Equation 3), results in a degradation on Search, but a small win on the Rare Words sets. However, when we use HAT with $\lambda_2 \neq 0$ (i.e., E2), we see improvements on both Search and rare words sets, compared to cascaded encoder with no LM (E0).

Table 1: *WER on Search and Rare words Sets*

Exp	Model	Search	Rare
B0	CascEnc, LSTM dec-causal	6.7	-
B1	CascEnc, LSTM dec-noncausal	5.6	-
E0	CascEnc, emb dec-noncausal	5.6	32.8
E1	E0 + no HAT, LM Resc	5.8	28.2
E2	E0 + LM Resc	5.5	26.8

4.2. Comparison to Conventional

4.2.1. *Latency*

Table 2 compares the WER, endpointer and prefetch latencies of conventional and CascEnc+LM on the Search test set. The Endpointer latency (median latency: EP50 and 90th latency: EP90) measures (in ms) the time between when the user stops speaking and the model declares the microphone to be closed. The prefetch latency (median latency: PF50 and 90th latency: PF90) measures (in ms) the latency of a correct prefetch relative to the end of the speech. When using CascEnc+LM (E2) with hybrid prefetching (HP), the PF90 is reduced by 130 ms. Overall, the on-device model has better WER, faster endpointer latency, on par prefetch latency, and is more than 318 times smaller than conventional. Note we do not report rare word performance with conventional since it uses the same pronunciation lexicon as the one used to synthesize the test set. Instead, this performance will be explored on held-out logs data in Section 4.2.3.

Table 2: *WER, Latency and Model Size on Search*

Model	Search	EP50	EP90	PF50	PF90	Size(G)
Conv	6.7	460	870	90	190	89.2
E2	5.5	350	700	130	380	0.28
+ HP	5.5	350	700	110	250	0.28

4.2.2. *Contextual Biasing*

We evaluate contextual biasing, described in Section 2.5, when biasing towards the names in a user's contact list. We biased towards a list of artificial contact names and measured performance against two test sets consisting of anonymized utterances from a Search application: an "in-context" set of utterances with intent to start a communication action such as "call $contact" or "message $contact," and an "anti-context" set of general voice search utterances in which communication action queries are rare. The in-context test set shows biasing performance when the context is relevant, and the anti-context set shows us how much general recognition degrades when we optimize for a biasing use case.

The results are shown in Table 3, where the "(Base)" value indicates the WER without biasing applied. The CascEnc+LM with HAT (E2) is observed to show improvement over the CascEnc+LM no HAT (E1). A large portion of the improvement here comes from the change to using HAT. We see improvement in both the in-context and anti-context sets for both the biased and no-bias variants. The relative WER improvements from biasing in the in-context set decreases and the relative WER regression in the anti-context set increases when using HAT compared to without HAT, but this is likely due to the no-bias variant performing better with HAT.

Table 3: *WER on Communication Test Sets*

Model	In-Context (Base)	Anti-Context (Base)
Conventional	8.7 (16.7)	6.9 (6.9)
E1 (no HAT)	5.3 (14.4)	5.9 (5.8)
E2 (HAT)	5.0 (13.2)	5.8 (5.4)

4.2.3. *Rare Word Modeling With SXS*

To better compare performance of our proposed CascEnc+LM model to conventional, we perform a "side-by-side" (SxS) on previously unseen utterances. We collect 500 utterances where the transcription differs between the two models, and send these utterances to be rated by two human transcribers. Each transcript is rated as either a win by CascEnc+LM over conventional (only CascEnc+LM is correct), a loss in CascEnc+LM over conventional (only the conventional model is correct), or neutral (both models are correct or incorrect). Note there is a neutral category since two models can output different text (Main St. *vs* Main Street) that are semantically similar.

We report five statistics to evaluate the SxS. *Changed* is % of utterances in which the two models produced different hypotheses. *Wins* is the # of utts the two-pass hypothesis is correct and conventional model is incorrect. *Losses* is the # of utts the two-pass hypothesis is incorrect and conventional model is correct. *Neutral* is the # of utts the two-pass and conventional model are both correct or incorrect. Finally, *p-Value* is the statistical significance of WER change with cascaded encoder+LM compared to conventional model.

Table 4 shows that the cascaded-encoder+LM model changes about 18% of traffic. The cascaded-encoder+LM model has significantly more wins (130) than losses (36) compared to the conventional model. Overall, the p-Value of $<$ 0.1% shows the performance difference between the two models is statistically *very* significant.

Table 4: *SxS: Conventional vs. Cascaded-Encoder+LM*

Changed (%)	Win	Loss	Neutral	p-Value
18.4	130	36	334	<0.1%

Table 5 shows the wins and losses of the model. The E2E model has some endpointer (EP) deletion losses since it operates at a faster latency. This seems to be the majority of losses in the SXS. There are a few lexicon(Lex) losses as well since conventional uses a lexicon, but the examples are few. The SXS shows very few losses in rare words, which indicates the E2E model is now addressing this issue. In comparison, the E2E model has many more wins in the LM area.

Table 5: *Error Analysis of Conventional vs. CascEnc+LM*

	Type	Conventional	CascEnc+LM
Loss	EP	cookies from brownie mix	cookies from
	Lex	play idealism phosphenes	play idealism phosphines
Win	LM	kyle parkway sears in tupelo, mississippi	call parkway terrace in tupelo mississippi
	LM	what is celtic stitching used for	what is shell tuck stitching used for

5. References

[1] T. N. Sainath, Y. He, B. Li, et al., "A Streaming On-Device End-To-End Model Surpassing Server-Side Conventional Model Quality and Latency," in *Proc. ICASSP*, 2020.

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

[3] A. Narayanan, T. N. Sainath, R. Pang, et al., "Cascaded encoders for unifying streaming and non-streaming ASR," in *Proc. ICASSP*, 2021.

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

[5] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *Proc. Interspeech*, 2016.

[6] J. Li, Y. Wu, Y. Gaur, et al., "On the Comparison of Popular End-to-End Models for Large Scale Speech Recognition," in *Proc. Interspeech*, 2020.

[7] Y. He, T. N. Sainath, R. Prabhavalkar, et al., "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[8] 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.

[9] A. Graves, "Sequence Transduction with Recurrent Neural Networks," *CoRR*, vol. abs/1211.3711, 2012.

[10] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *CoRR*, vol. abs/1508.01211, 2015.

[11] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017.

[12] T. N. Sainath, R. Pang, D. Rybach, et al., "Two-Pass End-to-End Speech Recognition," in *Proc. Interspeech*, 2019.

[13] D. Zhao, T. N. Sainath, D. Rybach, et al., "Shallow-Fusion End-to-End Contextual Biasing," in *Proc. Interspeech*, 2019.

[14] J. Chorowski and N. Jaitly, "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models," in *Proc. Interspeech*, 2017.

[15] J. Yu, C.-C. Chiu, B. Li, et al., "FastEmit: Low-latency Streaming ASR with Sequence-level Emission Regularization," in *Proc. ICASSP*, 2021.

[16] B. Li, S.-Y. Chang, T. N. Sainath, et al., "Towards fast and accurate streaming end-to-end ASR," in *Proc. ICASSP*, 2020.

[17] A. Gulati, J. Qin, C.-C. Chiu, et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, 2020.

[18] R. Botros and T.N. Sainath, "Tied & reduced rnn-t decoder," in *Proc. Interspeech*, 2021.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017.

[20] 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.

[21] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, et al., "Recognizing Long-Form Speech Using Streaming End-to-End Models," in *Proc. ASRU*, 2019.

[22] C. Allauzen and M. Riley, "Bayesian Language Model Interpolation for Mobile Speech Input," in *Proc. Interspeech*, 2011.

[23] F. Biadsy, M. Ghodsi, and D. Caseiro, "Effectively Building Tera Scale MaxEnt Language Models Incorporating Non-Linguistic Signals," in *Proc. Interspeech*, 2017.

[24] F. Biadsy, K. Hall, P.J. Moreno, and B. Roark, "Hybrid Autoregressive Transducer (HAT)," in *Proc. Interspeech*, 2014.

[25] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *CoRR*, vol. abs/1810.04805, 2018.

[26] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," *CoRR*, vol. abs/1901.02860, 2019.

[27] N. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A Conditional Transformer Language Model for Controllable Generation," *CoRR*, vol. abs/1909.05858, 2019.

[28] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to Google speech recognition," in *Proc. Interspeech*, 2015.

[29] Shuo-Yiin Chang, Bo Li, David Rybach, et al., "Low latency speech recognition using end-to-end prefetching," in *Proc. Interspeech*, 2020.

[30] H. Liao, E. McDermott, and A. Senior, "Large Scale Deep Neural Network Acoustic Modeling with Semi-supervised Training Data for YouTube Video Transcription," in *Proc. ASRU*, 2013.

[31] C. Kim, A. Misra, K. Chin, et al., "Generation of Large-Scale Simulated Utterances in Virtual Rooms to Train Deep-Neural Networks for Far-Field Speech Recognition in Google Home," in *Proc. Interspeech*, 2017.

[32] J. Li, D. Yu, J. Huang, and Y. Gong, "Improving Wideband Speech Rcognition using Mixed-bandwidth Training Data in CD-DNN-HMM," in *Proc. SLT*, 2012.

[33] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E.D. Cubuk, and Q.V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019.

[34] T.N. Sainath J. Apfel R. Pang S. Kumar C. Peyser, S. Mavandadi, "Improving Tail Performance of a Deliberation E2E ASR Model Using a Large Text Corpus," in *Proc.Interspeech*, 2020.

[35] X. Gonzalvo, S. Tazari, C. Chan, M. Becker, A. Gutkin, and H. Silen, "Recent Advances in Google Real-time HMM-driven Unit Selection Synthesizer," in *Proc. Interspeech*, 2016.

[36] M. Schuster and K. Nakajima, "Japanese and Korean voice search," in *Proc. ICASSP*, 2012.

[37] E. Variani, T. Bagby, E. McDermott, and M. Bacchiani, "End-to-End Training of Acoustic Models for Large Vocabulary Continuous Speech Recognition with TensorFlow," in *Proc. Interspeech*, 2017.