INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Streaming on-device end-to-end ASR system for privacy-sensitive voice-typing

Abhinav Garg, Gowtham Vadisetti, Dhananjaya Gowda,
Sichen Jin, Aditya Jayasimha, Youngho Han, Jiyeon Kim, Junmo Park, Kwangyoun Kim,
Sooyeon Kim, Youngyoon Lee, Kyungbo Min, Chanwoo Kim
Samsung Research, Korea

Abstract

In this paper, we present our streaming on-device end-to-end speech recognition solution for a privacy sensitive voice-typing application which primarily involves typing user private details and passwords. We highlight challenges specific to voice-typing scenario in the Korean language and propose solutions to these problems within the framework of a streaming attention-based speech recognition system. Some important challenges in voice-typing are the choice of output units, coupling of multiple characters into longer byte-pair encoded units, lack of sufficient training data. Apart from customizing a high accuracy open domain streaming speech recognition model for voice-typing applications, we retain the performance of the model for open domain tasks without significant degradation. We also explore domain biasing using a shallow fusion with a weighted finite state transducer (WFST). We obtain approximately 13 % relative word error rate (WER) improvement on our internal Korean voice-typing dataset without a WFST and about 30% additional WER improvement with a WFST fusion.

1. Introduction

Recent advances in the field of end-to-end neural network based speech recognition have brought the performance of a small foot-print low-latency on-device ASR almost on par with the larger server-side ASR systems [1, 2, 3, 4]. One of the key advantages of an on-device ASR is that it is privacy preserving, apart from saving huge server-side costs for the service provider. However, on-device speech recognition models still need significant engineering to make this work for privacy sensitive voice-typing. voice-typing can potentially serve as an efficient way to input especially for devices without a touch or physical keyboard such as smart television and wearable devices. The scenario of privacy sensitive voice-typing can be defined as one in which a user speaks his email ID, password, pin code, etc., character by character [5, 6]. While the transcription is typically restricted to English characters, numbers, special symbols and commands the spoken speech depends upon the native language or dialect of the speaker and the pronunciations in the corresponding language. Also, in many languages such as Korean the same output symbol can have multiple native as well as non-native spoken forms.

Recently, attention-based encoder-decoder models have gained popularity for developing end-to-end ASR systems [7, 8, 9, 10, 11, 12, 13]. These systems are capable of learning both the acoustic as well as linguistic information, unlike conventional systems which required separate modules for each of these tasks. Moreover, the end-to-end models are more suitable to be compressed since they do not need separate phonetic dictionaries and language models and hence making them one of the best candidates for on-device ASR system [2].

Attention-based models have been shown to perform better than other end-to-end models, namely, connectionist temporal classification (CTC) and recurrent neural network transducer (RNN-T) models [14]. With enough amount of data, they can even outperform the conventional models [7, 9]. However, most of these attention models use bidirectional long short-term memory (BLSTM) units and offline attention techniques which are not suitable for streaming decoding. Recently, there have been many efforts that make attention models streaming capable [15, 16, 17]. In [16], a hard monotonic attention based model was proposed for streaming decoding with acceptable accuracy degradation. A much improved soft monotonic chunk-wise attention (MoChA) based model was proposed in [17].

In this work, we propose MoChA based ASR system for the task of streaming on-device voice-typing. We aim at building a minimalistic ASR system that can run on-device under resource constrained environment of smart devices. While being under the umbrella of general ASR, voice-typing can be significantly different in terms of the challenges involved [6]. A voice-typing model primarily needs to run on-device for privacy issues, performance and needs to be small and efficient. Also, the average length of input utterance is much smaller for voice-typing, it has also been observed that online end-to-end model has delayed attention [2, 18] which can cause problems such as no output being generated for very short utterances and/or delayed output for others. Obtaining a large domain specific dataset is also a labor intensive task.

In this work, we address some of the above mentioned challenges. Due to unavailability of large scale domain specific data, we leverage the existing open domain dataset by training our baseline model on it. We then use data-augmentation techniques to adapt our model to the voice-typing domain. Although our main focus is to improve our performance for voice-typing, we want our system to perform well even for open domain usage. Finally, we also explore voice-typing specific post-processing techniques such as inverse text normalizer (ITN) and shallow fusion with a weighted finite state transducer (WFST). We achieve approximately 13% relative word error rate (WER) improvement on our internal Korean voice-typing test set without significant degradation in open domain WER.

2. Streaming attention encoder-decoder model for voice-typing

In this section, we define the model architecture and the training process for our baseline models similar to our previous work [2]. We also explore post processing techniques that we use to bias output for voice-typing.

2.1. Model architecture

We use an attention based encoder-decoder (AED) architecture which is composed of an encoder, a decoder, and an attention block. Our AED system is constructed using a `Tensorflow` [19] 2.0 Keras model. This model is trained using an *in-house* trainer built using `Tensorflow` 2.0-Keras APIs and `tf.data`. The encoder converts the input sequence into a sequence of hidden vector representations called encoder embeddings. The decoder is a generative module and predicts the target labels which are byte-pair encodings(BPE) in this work. Attention is used to learn how heavily a particular hidden vector sequence influences the current target label prediction.

http://dx.doi.org/10.21437/Interspeech.2020-3172

The encoder consists of several unidirectional LSTM (ULSTM) layers. It converts an input audio feature sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{T}$ of length T into a sequence of hidden encoder embeddings $\boldsymbol{h} = \{h_j\}_{j=1}^{T_e}$ of length T_e. Due to difference in input and output lengths, encoder-decoder models have difficulties in convergence. Hence, we use max-pooling layers having reduction factor two after the first three ULSTM layers.

The attention block computes a context vector c_l based on the encoder embeddings $\{h_{1:k}\}$ and the previous decoder state s_{l-1}. Where l is the decoder output step. In this paper, we use MoChA for the attention block to ensure online or streaming decoding [17]. The decoder consists of one or more ULSTM layers followed by a few fully connected layers and a softmax layer. The decoder uses the attention context vector c_l, the previous state of the decoder s_{l-1}, and the previous decoded label y_{l-1} to generate the label probabilities $P(y_l|y_{1:l-1}, x_{1:i})$ at the softmax output.

2.2. Training

For training, cross entropy loss is calculated between the final output of the decoder and ground truth labels $\boldsymbol{y^*}$:

$$L_{CE} = -\sum_{l=1}^{L} \log(P(y_l^*|y_{1:l-1}^*, x_{1:i})), \quad (1)$$

here $x_{1:i}$ is used to denote streaming input. We additionally apply a feed forward layer and a softmax layer over encoder embeddings h and compute CTC loss between the encoder outputs and the ground truth labels. We then use joint CTC-CE loss as our final loss objective[20].

As our final recognition accuracy measure was word error rate (WER), we also define a loss on the candidate beams of the beam search which effectively minimizes the expected WER. We use the N-best variant for minimum word error rate(MWER) loss [21] defined as:-

$$L_{MWER} = \sum_{\boldsymbol{y}^i \subset Beam(\boldsymbol{x}, N)} P(\boldsymbol{y}^i|\boldsymbol{x})(\mathcal{W}(\boldsymbol{y}^i, \boldsymbol{y}^*) - \bar{\mathcal{W}}) \quad (2)$$

Where $Beam(\boldsymbol{x}, N)$ is set of top N hypothesis computed using beam search. $P(\boldsymbol{y}^i|\boldsymbol{x})$ is re-normalized distribution over top N hypothesis. $\mathcal{W}(\boldsymbol{y}^i, \boldsymbol{y}^*)$ represents number of word errors in hypothesis $\boldsymbol{y}^i$. $\bar{\mathcal{W}}$ is average number of word error calculated over top N hypothesis.

2.3. Compression and quantization

We use a singular value decomposition based low rank approximation algorithm (LRA) to compress our model. However, for large compression ratio it is difficult for the LRA algorithm to converge as the LRA distortion (the difference between LRA approximated weight matrix and the actual weight matrix) propagates through layers and increases the training loss. To deal with this issue we use the Hyper-LRA algorithm proposed in [22] which uses DeepTwist method to find the optimal LRA weights. To further reduce the memory footprint and increase the speed in order to meet real-time requirements we apply 8-bit quantization using Tensorflow-lite.

2.4. Domain specific post processing during inference

2.4.1. Domain adaptation

To bias the output for voice-typing, we use on-the-fly rescoring approach [23]. Word-level voice-typing phrases are prepared in advance and are compiled into a language model WFST, G. It is then composed with a lexicon FST, B, which spells out BPE combinations for each word instead of the FST L used in conventional HCLG approach [24]. The domain LM is made after determinization and minimization ($P_{LM} = \min(\det(B \circ G))$). The integration between the decoder model and the domain LM is accomplished with shallow fusion [25]:

$$y_l = \arg\max_y (P(y_l|y_{1:l-1}, \boldsymbol{x}) + \lambda P_{LM}(y_l|y_{1:l-1})) \quad (3)$$

2.4.2. Inverse Text Normalization (ITN)

As the final step in our inference process, we use a rule-based ITN to convert our model output to user-displayed results. The rules consist of one-to-many mappings between numbers, alphabets, special characters to their corresponding texts in target language. For cases where we have multiple matches we select the candidate having the maximum string length. While building ITN for Korean we also resolve the ambiguity in Korean to English translation using output context (Eg:- *ㅇ](e)* in Korean can refer to English alphabet *e* as well as Korean pronunciation for number *2*. If output proceeding *ㅇ](e)* is a number we choose *2* else we choose *e*).

3. Challenges for voice-typing system

Apart from being privacy preserving, on-device ASR solution also reduces the network latency and improves the responsiveness. An on-device ASR solution needs to be both memory efficient and computationally inexpensive to be able to perform under the resource constraint environments of smart devices. Apart from the resource constraints, on-device voice-typing ASR also presents other distinctive challenges and choices. In this section, we discuss some such challenges and their proposed solutions.

3.1. Choice of output units

The modeling unit for AED models has been shown to have a large impact on the overall performance [8, 26, 27, 28]. While conventional speech recognition systems used phoneme-based models, it was shown recently that word-piece models for AED models consistently outperform phoneme-based models [28]. Using phoneme or grapheme for voice-typing has the following advantages: (i) As the current output has little to no dependency on the previous output, there might be no potential benefit from the dependence captured by word-piece units. Hence, for voice-typing grapheme represents the output most closely. (ii) Lower vocabulary size would mean a lower memory footprint.

However, these advantages disappear for open domain usage. Even with a low memory footprint, the total number of decoder runs required by grapheme based models is high leading to a higher overall time when compared to word-piece based models.

As we want our system to perform well on open domain as well, we choose to use word-piece as our modeling unit. Most word-piece based ASR models use a vocabulary size of 10K-15K [8, 10, 26, 9] for best performance. However, for voice-typing using a high vocab size makes the final memory footprint of the model bigger and has output coupling problem to be discussed in section 3.2.

3.2. Output coupling and delayed attention

For a real-time ASR system, latency is also an important metric along with recognition accuracy. For online attention models, it has been observed that attention alignment is delayed as compared to full-attention models [2, 18] as they cannot use future information. While this does not affect the recognition accuracy of an ASR model, it affects the latency of the output.

In voice-typing, using a larger vocabulary size makes the latency problem more severe. As the output units now consist of coupled characters as well, the ASR system is waiting for the next character to be spoken in order to output a composite BPE unit having both characters together

To solve the above problems of the choice of output units, output coupling and delayed attention, we propose to use a smaller BPE vocabulary consisting of about 2K BPE units. Also, to make sure that the effect of output coupling is not severe, we build our vocabulary from a text corpus consisting of 60% voice-typing data and 40% open domain data. Fig 1 shows an example utterance having speech signal as Korean pronunciation for $abcd$. While, the model with 10K vocabulary runs for 3 decoder steps, and outputs cd coupled. Model with 2K vocabulary runs for 4 steps and outputs each character separately.

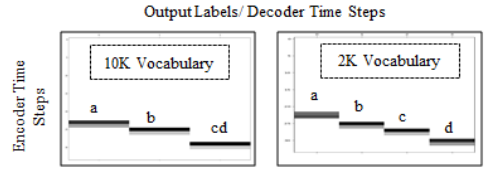

Figure 1: *Alignments produced by 10K and 2K vocabulary models*

3.3. No or delayed attention

The delayed attention phenomenon discussed above causes a severe problem for short utterances having transcripts consisting of about 1-2 BPE units. For such utterances, on multiple occasions, we observe empty output from the MoChA model. In order to address this issue, we append a short segment of silence or dummy audio at the end of input audio if the length of input audio is smaller than 2 sec. This improves the system performance significantly on short utterances.

3.4. Model size

For our network architecture, the encoder is the major contributor to model size. While using a small cell size for ULSTM units can lead to smaller memory footprint and better decoding speeds, it has been observed [10] that using bigger cell size gives significantly better recognition accuracy for AED models. Hence, we decide to use a bigger cell size for our experiments. As mentioned in Section 2.2 we use compression and quantization techniques to compress our model without significant loss in recognition accuracy.

3.5. WFST biasing issues

For calculating domain LM probability, $P_{LM}(y_l|y_{1:l-1}))$, we proceed a decoding step with all possible y_l from the vocabulary. Since we are using word level language model G, lexicon B, WFST paths for some y_l might be absent. For such y_l, they would get pruned even though the E2E model gives a high probability for them. We introduce two techniques to prevent this: (i) Giving a low default LM weight to the BPE sequences which were not present in the WFST. (ii) Using all possible BPE expressions of a word when creating the lexicon FST B. While the second technique provides significant improvements in recognition accuracy it also leads to exponential growth in the size of WFST for longer words. We explore splitting long phrases into shorter words to keep the FST size small.

4. Model adaptation and data augmentation

An on-device voice-typing system for smart device should be capable of running on a wide variety of user devices such as TV, smartwatch, AI speaker. Software wise these systems can be quite similar, but their hardware (processors, memory, mic, etc) varies significantly from device to device. Hence, apart from being compact and fast, an on-device voice-typing system should be trained on speech data from variety of environments. Recently, many large scale datasets have been proposed, fueling the development of end-to-end ASR systems [29, 30]. However, most of these datasets are open domain. While they can be used to train an open domain ASR, we must use data augmentation techniques to adapt our model for a limited domain. For this purpose, we use text-to-speech (TTS) generated utterances for voice-typing. We also collect a small dataset of about 6 hours consisting of voice-typing data. To up sample our recorded dataset and make our model robust, we use acoustic room simulator [31] and adapt our model obtained from Section 2 to the voice-typing domain.

4.1. TTS data augmentation

As an initial augmentation method, we use Deep Convolution TTS(DCTTS [32]) to generate voice-typing data. We randomly generate transcriptions for voice-typing which contains a mixture of alphabet, numbers and email formatted sentences. In Korean, the pronunciations of numbers depend on them being ordinal or cardinal numbers, and we randomly sample from the two pronunciations. For the alphabet, we consider both capital and small letters. Regarding symbols, we convert them to their pronunciations in Korean. With all these variants, we generate about 41,563 speech utterances for each male and female professional voice actor.

4.2. Acoustic Simulator

To augment our dataset with simulated audio of different acoustic environments, we use the *acoustic simulator* as described in [31]. Using our *acoustic simulator*, we generate simulated speech utterances for various room dimensions, a wide distribution of reverberation time and signal-to-noise(SNR) ratio [31]. We also add a small bias to the reverberation time and SNR distributions for simulating near-field audio more frequently.

5. Experiments

We evaluated our baseline approaches for Korean and English. For training our baseline models we used anonymized open domain corpus consisting of around 10K hours of transcribed speech for Korean and English each [2]. Out of which we sampled one hour data for our test and validation sets for the open domain. We use the 40-dimensional power mel-frequency filterbank features which are computed by power operation of mel-spectrogram with $\frac{1}{15}$ as the power function coefficient [33], which were then combined into variable batch sized chunks so as to keep the number of frames processed per training step constant. We use 6 ULSTM layers for our encoder each with 1536 units. Our decoder consisted of a single ULSTM layer with 1000 units. For Korean voice-typing domain adaptation, we used 6 hours of recorded data and 120 hours of TTS generated data as our training set. We used separate 1 hour recorded data for our voice-typing test set.

To leverage open domain information we started with a model having the same setup and training procedure to [2] and having a vocabulary size of 10K. We then adapted it for voice-typing by reducing the vocabulary size to 2K and retraining it with a combined dataset consisting of open domain dataset, TTS generated voice-typing dataset, recorded voice-typing dataset in ratio 3:3:4 respectively. Recorded voice-typing dataset size was up-sampled using acoustic simulator [31]. We used tensorflow 2.0 APIs [26, 34] for performing all our training experiments.

For optimal performance, we implemented our inference code in C++. We used `tflite` [35] along with `openfst` for implementing our inference code. We do not employ any processor specific optimizations such as neon other than those already implemented in `tflite`. For generating output hypothesis, we used beam search with a beam size of 4 for all our ex-

Table 1: *WER(%) for our baseline models* [2] *having 10K vocabulary size and trained using techniques mentioned in Section 2*

Language	dataset	Baseline Model	MWER + SVD Model	8 Bit Model	8 Bit Model plus WFST
English	Open domain	9.03	8.64	8.88	-
	Voice Typing	-	9.28	10.02	6.60
Korean	Open domain	9.37	9.60	9.80	-
	Voice Typing	-	21.4	22.51	16.76

Table 2: *WER comparison for various Korean voice-typing*

Model	Open domain WER	Voice Typing WER
8-bit baseline	9.80	22.51
+vocab compression	10.1	23.3
+TTS augmentation	10.06	22.00
+Recorded Data Augmentation	10.29	20.1
+Silence concatenation	10.28	19.82
+Dummy concatenation	10.24	19.53

Table 3: *WER on voice-typing using fusion with various WFST G_B*

WFST	fusion weight	WER(%)	Size on Disk(KB)
no wfst	0	19.82	0
WFST *init_one*	0.3	18.68	19
WFST *init_all*	0.3	15.00	101
WFST *final_one*	0.4	19.62	4.9
WFST *final_all*	0.45	13.94	47

periments. All our performance benchmarks were obtained on a smart user device board having Cortex-A72 processor with a clock speed of 1.75Ghz and 2GB memory. Compared to this, most modern smartphones are equipped with an architecture having up to 2.8Ghz clock speed and up to 16GB memory. ITN modules were used before evaluating results for voice-typing domain.

6. Results

Our baseline model performances are shown in Table 1. We evaluated our baseline models on both open domain and voice-typing test set. All models shown in Table 1 use 10K vocabulary size, and were trained with only open domain corpus. Intial baselines were trained using the joint CE-CTC loss. After which SVD was applied to compress the model and MWER was jointly applied to compensate the performance loss. WFST results are reported using WFST *final_all* (described later) for both the languages.

Korean baseline models perform significantly worse on the voice-typing dataset as compared to the open domain. This is since training utterances are mostly monolingual (Korean) whereas the voice-typing dataset containing email Ids, passwords, pin code, etc is mostly bilingual (Korean and English). Obtaining bilingual voice-typing data is a labor intensive task and we leverage the available open domain Korean data for the task of voice-typing.

Our English baseline model performs well on open domain as well as the voice-typing dataset. Also, we observed that output coupling was not a major problem for English models. However, even English model had no attention problem. To evaluate the effect for appending silence/dummy audio at the end of the input speech, we used 4,482 non-silence utterances from the speech command dataset's test set [6] each of length 1 sec. The baseline English models produced no output for 18% of the test sentences. After appending silence of 300ms at the end of each utterance we attained 38% decrease in the number of empty outputs. Further, appending dummy audio of about 300ms achieved 59% additional decrease.

To evaluate the effectiveness of using smaller BPE vocabulary to solve the output coupling problem, we selected some utterances having coupling problem with larger BPE vocabulary and measured average latency between the time a character was spoken and it was predicted by the ASR system. We found that even while the total decoding time for system with larger BPE vocabulary was slightly less than smaller BPE vocabulary system, the average latency of the system with smaller BPE vocabulary was 20% less.

Table 2 shows the improvements in performance for the Korean voice-typing test set using domain specific techniques. Reducing the output BPE vocabulary size to 2K from 10K worsened both open domain and voice-typing domain performance by 3%. However, retraining with TTS data augmented recovered the performance for voice-typing while the performance on the open domain remained almost unchanged. Using recorded data further improved the voice-typing performance by 10% while the open domain performance degraded by just 1.5%. Appending 300ms silence at the end of test utterance gave an additional 5% improvement. Since the number of short utterances (<2sec) in voice-typing test case is only 5-6%, dummy concatenation did not helped much.

In order to bias our models to the voice-typing domain, we created a WFST with a word-level component G and a word-to-BPE speller WFST B. First, we collected the top 1000 most frequent phrases from voice-typing text corpus and created a word-level WFST G, $init$ using all these sentences. We then split these phrases into unique words of about 400 words and created another WFST G, $final$. For the speller WFST B we considered two variations for sub-word sequence. In the first variation, WFST B, one we used the one-best BPE sequence, and in the second variation WFST B, all we used all the possible subword sequences of a word made using symbols from BPE vocabulary. We compare domain biasing using the various combinations of WFST B and G. We performed a linear search for most optimal fusion weights. As shown in 3 WFST B, all worked better than B, one for both WFST G, due to pruning phenomenon discussed in Section 3.5. We achieved about 30% improvement using WFST *final_all* with fusion weight 0.45 and achieved 13.94% WER on Korean voice-typing dataset.

The average output latency of our final models was 252ms and 220ms for Korean and English respectively. The total memory(RAM, Code + Data) usage was 50MB and 70MB respectively for Korean and English. The real time factor (xRT) was 0.62 on Cortex A72 (1.75GHz) and 0.4 on Cortex-A75 (2.31GHz) for both the models.

7. Conclusion

We present our streaming on-device end-to-end ASR solution for the task of voice-typing using MoChA attention. The proposed architecture is compact enough to fit on most smart devices and fast enough to do real time online decoding. Using data augmentation we achieve about 13% improvement on the voice-typing dataset without significant loss in open domain accuracy. Finally, we apply domain biasing with WFST and achieve an additional 30% improvement for voice-typing.

8. References

[1] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[2] K. Kim, K. Lee, D. Gowda, J. Park, S. Kim, S. Jin, Y. Y. Lee, J. Yeo, D. Kim, S. Jung, J. Lee, M. Han, and C. Kim, "Attention based on-device streaming speech recognition with large speech corpus," in *Proc. ASRU*, 2019.

[3] D. Gowda, A. Garg, K. Kim, M. Kumar, and C. Kim, "Multi-task multi-resolution char-to-bpe cross-attention decoder for end-toend speech recognition," 2019.

[4] D. Gowda, A. Kumar, K. Kim, H. Yang, A. Garg, S. Singh, J. Kim, M. Kumar, S. Jin, S. Singh, and C. Kim, "Utterance invariant training for hybrid two-pass end-to-end speech recognition," in *Proc. Interspeech*, 2020.

[5] A. Kumar, T. Paek, and B. Lee, "Voice typing: a new speech interaction model for dictation on touchscreen devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 2277–2286.

[6] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018.

[7] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018, pp. 4774–4778.

[8] A. Zeyer, T. Alkhouli, and H. Ney, "Returnn as a generic flexible neural toolkit with application to translation and speech recognition," in *ACL*, 2018.

[9] D. S. Park, W. Chan, Y. Zhang, C.-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.

[10] A. Garg, D. Gowda, A. Kumar, K. Kim, M. Kumar, and C. Kim, "Improved multi-stage training of online attention-based encoder-decoder models," in *Proc. ASRU*, 2019.

[11] C. Kim, K. Kim, and S. R. Indurthi, "Small energy masking for improved neural network training for end-to-end speech recognition," in *ICASSP*, 2020.

[12] A. Garg, A. Gupta, D. Gowda, S. Singh, and C. Kim, "Hierarchical multi-stage word-to-grapheme named entity corrector for automatic speech recognition," in *INTERSPEECH-2020*, Oct. 2020.

[13] A. Kumar, S. Singh, D. Gowda, A. Garg, S. Singh, and C. Kim, "Utterance confidence measure for end-to-end speech recognition with applications to distributed speech recognition scenarios," in *INTERSPEECH-2020*, Oct. 2020.

[14] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition." in *Proc. Interspeech*, 2017, pp. 939–943.

[15] N. Jaitly, D. Sussillo, Q. V. Le, O. Vinyals, I. Sutskever, and S. Bengio, "A neural transducer," *ArXiv*, vol. abs/1511.04868, 2016.

[16] C. Raffel, T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proc. ICML*, 2017.

[17] C. C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *International Conference on Learning Representations*, 2018.

[18] H. Miao, G. Cheng, P. Zhang, T. Li, and Y. Yan, "Online hybrid ctc/attention architecture for end-to-end speech recognition," *Proc. Interspeech 2019*, pp. 2623–2627, 2019.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[20] T. Hori, S. Watanabe, Y. L. Zhang, and W. Chan, "Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM," in *INTERSPEECH*, 2017.

[21] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C. Chiu, and A. Kannan, "Minimum word error rate training for attention-based sequence-to-sequence models," in *Proc. ICASSP*, 2018, pp. 4839–4843.

[22] D. Lee, P. Kapoor, and B. Kim, "Deeptwist: Learning model compression via occasional weight distortion," *CoRR*, vol. abs/1810.12823, 2018.

[23] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. ICASSP*, 2016.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[25] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *CoRR*, vol. abs/1503.03535, 2015.

[26] C. Kim, S. Kim, K. Kim, M. Kumar, J. Kim, K. Lee, C. Han, A. Garg, E. Kim, M. Shin, S. Singh, L. Heck, and D. Gowda, "End-to-end training of a large vocabulary end-to-end speech recognition system," in *Proc. ASRU* , 2019.

[27] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[28] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, and P. Nguyen, "On the choice of modeling unit for sequence-to-sequence speech recognition," *Proc. Interspeech*, pp. 3800–3804, 2019.

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

[30] Paul, Douglas B and Baker, Janet M, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language*. ACL, 1992, pp. 357–362.

[31] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "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] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *ICASSP 2018*, April 2018, pp. 4784–4788.

[33] C. Kim, M. Shin, A. Garg, and D. Gowda, "Improved vocal tract length perturbation for a state-of-the-art end-to-end speech recognition system," *Proc. Interspeech 2019*, pp. 739–743, 2019.

[34] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.

[35] G. Inc., "Tensorflow Lite," Online documents; https://www.tensorflow.org/lite, 2018.