AN ATTENTION-BASED JOINT ACOUSTIC AND TEXT ON-DEVICE END-TO-END MODEL

Tara N. Sainath, Ruoming Pang, Ron J. Weiss, Yanzhang He, Chung-cheng Chiu, Trevor Strohman

Google, Inc., USA

{tsainath, rpang, ronw, yanzhanghe, chungchengc, strohman}@google.com

ABSTRACT

Recently, we introduced a two-pass on-device end-to-end (E2E) speech recognition model, which runs RNN-T in the first-pass and then rescores/redecodes the result using a noncausal Listen, Attend and Spell (LAS) decoder. This on-device model obtained similar performance to a state-of-the-art conventional model. However, like many E2E models, it suffers from being trained only on supervised audio-text pairs and thus performs poorly on rare words compared to a conventional model which incorporates a language model trained on a much larger text corpus. In this work, we introduce a joint acoustic and text decoder (JATD) into the LAS decoder, which makes it possible to incorporate a much larger text corpus into training. We find that the JATD model obtains in a 3-10% relative improvement in WER compared to a LAS decoder trained only on supervised audio-text pairs across a variety of proper noun test sets.

1. INTRODUCTION

E2E models [1, 2, 3, 4, 5, 6, 7, 8] combine the acoustic (AM), pronunciation (PM) and language models (LM) of a conventional speech recognition system into a single neural network. Such models are a fraction of the size of conventional models and are therefore ideal for embedded on-device ASR applications [1]. While these models have shown competitive performance to conventional models [2, 9], their performance still lags behind on rare words, such as long-tail named entities. One reason for this performance gap is that E2E models are often trained only on supervised audio-text pairs, i.e. the training set for a conventional acoustic model, which is a small fraction of the size of the text data used to train a conventional LM.

Numerous research ideas have been explored for training E2E models using a large amount of unpaired data, either with text-only data or unsupervised audio-text data. For example, training an RNN-LM on text-only data and fusing this into the E2E model, via techniques such as shallow, cold and deep fusion, has shown improvements on long-tail queries [10, 11, 12]. However, these techniques are not amenable to run on-device since the external LM significantly increases model size and computational cost [1]. Another approach uses an RNN-LM, trained on text-only data, as the teacher model to regularize the label distribution of the E2E model while training on paired data [13]. While this does not increase the model size during inference, it still requires training a separate RNN-LM as a first stage before training the E2E model.

Training the E2E model on unsupervised audio-text pairs, known as "weak distillation" [14], has also shown benefits for long-tail proper noun queries, without increasing model size. However, the common recipe for training with this data – first pre-training with supervised-only data, and then fine-tuning on supervised and unsupervised data – significantly increases training time [14]. Furthermore, weak distillation is vulnerable to errors from the teacher model.

Synthesizing speech from text-only data and then using it to train the E2E decoder model, a technique known as "back-translation" [15] in the machine translation literature, has had limited success in ASR when evaluated on real audio sets [14, 16]. Alternatively, "cycle-consistency" [17, 18] has also been explored to train E2E models on unpaired speech and text data. This technique also requires model retraining-steps, first supservised and then supervised + unsupservied.

We propose a new method for training the *decoder* of the E2E model on unpaired data. Our goal is to develop a technique that can leverage a large amount of unpaired data when training *on-device* models without significantly increasing model size, training time and inference cost. Our technique, which we call a joint acoustic/text decoder (JATD), is motivated by past work in multilingual/multi-dialect E2E models [19, 20, 21]. In that work, a domain identifier, or "domain ID", is appended to the input to indicate the language or dialect of the speech. With this added bit of information, a single model can perform well on many languages or dialects. In this work, we adpot a similar technique, using a domain ID to indicate whether a training example corresponds to a supervised audio-text pair or an audio-text example generated from unpaired data. For unpaired data, we can synthesize the missing half of the pair: if we have text data, we can use TTS to synthesize corresponding audio; if we have audio data, we can use ASR to hypothesize a text transcription. During training with a paired example, an acoustic context vector computed from the encoder is fed into the decoder, representing a paired data domain ID. However, for unpaired examples, a fixed but learnable context vector domain ID is used instead, thus bypassing the encoder network. This technique allows the decoder to be trained simultaneously on both paired and unpaired data without increasing model size.

A key benefit of this approach is that it does not increase inference latency significantly, and is thus friendly for on-device deployment. Here we take advantage of the fact that end-user devices are increasingly becoming equipped with processors (GPUs, NPUs, or TPUs) that can cheaply run large batches of computation against a single set of model parameters. Since our technique does not increase the overall number of model parameters significantly, it will work well on such systems. In addition, it does not require additional training stages, unlike past work with unsupervised data [14]. The proposed approach is similar in spirit to [22] which explored a multi-modal E2E decoder that took both acoustic and symbolic input. A main difference is that [22] shares attention and decoder parameters across all modes, only changing the parameters of the input passed to the encoder depending on the input source. In contrast, in our approach, only decoder parameters are shared. Different attention context parameters are used, depending on the source.

We evaluate the JATD model within the RNN-T + LAS two-pass decoding framework [9], which is effective for resource constrained on-device applications. Specifically, the RNN-T decoder emits first-pass hypothesis transcripts, which the LAS decoder is used to rescore

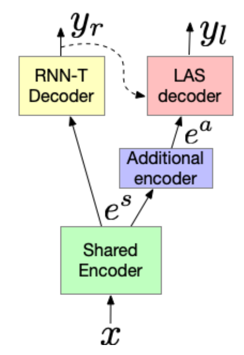

Fig. 1: Two-Pass Architecture

or refine. In this work, only the LAS decoder is modified according to the JATD framework. We evaluate JATD on a variety of proper noun and rare-word test sets, and find that it gives a 3-10% relative reduction in word error rate (WER), when used for rescoring/beam-search, compared to a LAS decoder trained only on paired data.

2. JOINT ACOUSTIC AND TEXT MODEL

2.1. Two-Pass E2E Architecture

The two-pass architecture is illustrated in Figure 1. The input acoustic frames are denoted as $\mathbf{x} = (\mathbf{x}_1 \ldots \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^{128}$ is a frame of log-mel filterbank energies and T is the number of frames. We denote the ground-truth label sequence of length U as $\mathbf{y} = (y_1, \ldots, y_U)$, where $y_u \in \mathcal{Z}$ and $\mathcal{Z}$ corresponds to the set of word-piece [23] output units.

Training takes place in two stages. First, the shared encoder and RNN-T decoder are trained to maximize $P(\mathbf{y}_r = \mathbf{y}|\mathbf{x})$. Next, the shared encoder is held fixed and the additional encoder (AE) + LAS decoder (including the attention context) are trained to maximize $P(\mathbf{y}_l = \mathbf{y}|\mathbf{x})$. We add the AE since it is found to be useful to adapt the encoder output to be more suitable for LAS 2nd-pass models. Decoding is performed in two passes. In the first pass each acoustic frame $\mathbf{x}_t$ is passed through a shared encoder, consisting of a multi-layer LSTM, to get output $\mathbf{e}_t^s$, which is passed to the RNN-T decoder which emits $\mathbf{y}_r$ streaming fashion. In the second pass, the shared encoder output for all frames, $\mathbf{e}^s = (\mathbf{e}_1^s \ldots \mathbf{e}_T^s)$ is passed to a small AE to generate $\mathbf{e}^a = (\mathbf{e}_1^a \ldots \mathbf{e}_T^a)$, which is then passed to a non-causal attention module to compute a context vector $\mathbf{c}_u$ which summarizes the encoder features $\mathbf{e}^a$ for a each output step $u \in U$.

The LAS decoder can run in two modes. When run in *beam-search mode*, the decoder produces output $\mathbf{y}_l$ based on $\mathbf{e}^a$ alone, ignoring $\mathbf{y}_r$, the output of the RNN-T decoder. Alternatively, in *rescoring mode*, the decoder consumes the top-K hypotheses from the RNN-T decoder. The LAS decoder computes a score for each hypothesis by evaluating the network in teacher-forcing mode, with attention on $\mathbf{e}^a$. The resulting score combines the log probability of the sequence and the attention coverage penalty [24]. The sequence with the highest LAS score is chosen as the final output sequence.

2.2. JATD Model Description

Work in multi-dialect/lingual E2E ASR [19, 20, 21] has shown that passing an embedding vector representing the domain of the input data (i.e., dialect or language) is effective in helping the model handle different types of inputs. In this spirit, we explore changing the context vector $\mathbf{c}_u$ passed to the LAS decoder depending on the type of input training data, thus developing a joint acoustic and text decoder. The model is depicted in Figure 2.

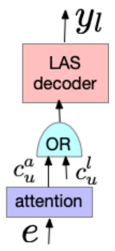

Fig. 2: Joint Acoustic and Text Model. During fprop, either the acoustic context $\mathbf{c}_u^a$ or fixed context $\mathbf{c}_u^l$ is used.

Specifically, if the input example is a supervised audio-text pair, we pass an acoustic context vector $\mathbf{c}_u^a$ to the decoder. However, if the input is an unpaired text sequence, we instead pass a fixed but learnable language model context vector $\mathbf{c}_u^l$ into the decoder. Below, we describe changes to the LAS second-pass decoder inference and training procedures to utilize both paired and unpaired data.

2.2.1. *Inference*

The log probabilities computed for LAS during inference are shown in Eq. 1. Given acoustic input $\mathbf{x}$, we can evaluate the LAS decoder using acoustic context vector $\mathbf{c}_u^a$ to compute log probabilities $\log p(\mathbf{y}_u|\mathbf{x}, \mathbf{c}_u^a, \mathbf{y}_{u-1:1})$ at each decoder step u. Here $\mathbf{y}_{u-1:1} = \{y_{u-1}, \ldots, y_1\}$ indicates all previous decoded labels of a single hypothesis during inference. Alternatively, the text context vector $\mathbf{c}_u^l$ can be used to compute log probabilities $\log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1})$. Like an RNN-LM, this formulation predicts labels based on previous labels alone, ignoring audio features entirely. We can think of these two distributions as corresponding to AM and LM scores, respectively, since each is generated from either acoustic or text context vectors. For each autoregressive decoder step u, we interpolate the two log probabilities with mixing weight λ. Note that this inference is used for each step u, when LAS is run as a rescorer or beam search. Another advantage of the JATD model is that the RNN-LM-like component can be run and cached during the first-pass RNN-T inference, so JATD does not increase 2nd-pass latency compared to standard LAS.

$$\lambda \log p(\mathbf{y}_u|\mathbf{x}, \mathbf{c}_u^a, \mathbf{y}_{u-1:1}) + (1-\lambda) \log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1}) \quad (1)$$

2.2.2. *Training*

In all cases, the RNN-T model is first trained with supervised audio-text paired data. Then we experiment with two strategies for training the LAS model with paired and unpaired data. In the *individual* strategy if audio-text paired data is used, as shown in Eq. 2, we evaluate $\log p(\mathbf{y}_u|\mathbf{x}, \mathbf{c}_u^a, \mathbf{y}_{u-1:1})$ with the acoustic context vector $\mathbf{c}_u^a$ and update the LAS decoder and acoustic context vector parameters. However, if unpaired data is used, the training loss reduces to the cross-entropy loss computed from $\log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1})$, where $\mathbf{c}_u^l$ is the trainable context vector. In this case, only the LAS decoder and context vector are updated. In Section 4, we show results varying the amount of paired and unpaired data seen in training.

$$\mathcal{L} = \begin{cases} \log p(\mathbf{y}_u|\mathbf{x}, \mathbf{c}_u^a, \mathbf{y}_{u-1:1}), & \texttt{if paired example} \\ \log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1}), & \texttt{if unpaired example} \end{cases} \quad (2)$$

Alternatively, in the *joint* strategy we define the training loss by interpolating log probabilities generated from acoustic and text context vectors (Eq. 3), as is done in inference. We denote supervised audio data as $\mathbf{x}^a \in \mathbf{x}$. Thus, for examples which contain supervised audio-text pairs, we interpolate with both $\log p(\mathbf{y}_u|\mathbf{x}, \mathbf{c}_u^a, \mathbf{y}_{u-1:1})$ and $\log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1})$, updating only the LAS decoder and acoustic attention parameters. However, for unpaired data, since we need to compute log probabilities with an acoustic context vector, an audio-text pair needs to be used, where we denote $\mathbf{x}^l \in \mathbf{x}$ as the "created audio". There are two options that can be explored. First, we can take real audio and use a conventional model to generate hypothesize text, similar to [14]. Using untranscribed audio in this way is akin to model distillation from a conventional recognizer. Alternatively, we can take the given text and synthesize an acoustic signal. In this work, we explore only the former. By creating x^l, we can interpolate the following in training $\log p(\mathbf{y}_u|\mathbf{x}^l, \mathbf{c}_u^a, \mathbf{y}_{u-1:1})$ and $\log p(\mathbf{y}_u|\mathbf{c}^l, \mathbf{y}_{u-1:1})$, but we only update the LAS decoder and fixed context vector parameters, in order to avoid biasing the acoustic attention parameters toward unpaired data. $\lambda = 0.1$ was found to work well in training.

$$\mathcal{L} = \begin{cases} \lambda \log p(\mathbf{y}_u|\mathbf{x}^a, \mathbf{c}_u^a, \mathbf{y}_{u-1:1}) + (1-\lambda)\log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1}), \\ \texttt{if paired example} \\ \lambda \log p(\mathbf{y}_u|\mathbf{x}^l, \mathbf{c}_u^a, \mathbf{y}_{u-1:1}) + (1-\lambda)\log p(\mathbf{y}_u|\mathbf{c}_u^l, \mathbf{y}_{u-1:1}), \\ \texttt{if unpaired example} \end{cases} \tag{3}$$

3. EXPERIMENTAL DETAILS

The paired audio-text training set used for experiments, the same as [25], consists of multi-domain utterances spanning domains of search, farfield, telephony and YouTube English utterances. The search and farfield utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation such that the overall SNR is between 0dB and 30dB, with an average SNR of 12dB [26]. The noise sources are from YouTube and noisy environmental recordings. The unpaired data is collected by mining anonymized utterances from voice search traffic. These utterances are decoded by a state-of-the-art conventional ASR model [27], and thus the transcription is not ground-truth. The data used in these experiments is similar to that described in [19].

Our primary evaluation set is broken into a set of ~13K short utterances (*SU*) less than 5.5 seconds long and another set of ~14K long utterances (*LU*) with duration longer than 5.5 seconds, both extracted from Google traffic and also anonymized.

To evaluate improvements on long-tail entities, we also evaluate on 4 sets. The *Corr* (Corrections) dataset covers 5K utterances where the user typed a query immediately after speaking, which likely means that the recognition result was not correct. In addition, the *SXS* set contains a set of 1K utterances where the quality of the E2E model (without an LM) transcription has more errors than a state-of-the-art conventional model [27]. It is useful for measuring how including text-only training data can improve some of these errors.

As described in [16], the *Songs* and *Apps* sets are created by synthesizing sentences in each of these categories using a Parallel WaveNet TTS model [28], The *Songs* test set contains 15K media requests (e.g. `play rihanna music`) with phrases containing popular songs and artist names in US English. The *Apps* test set contains 16K requests to interact with an app (e.g. `open trivia game`) with phrases containing popular app names. Noise is artificially added to the synthetic data, similar to [26]. .

3.1. Modeling

All experiments use 128-dimensional log-Mel features, computed with a 32ms window and shifted every 10ms. Similar to [25], features for each frame are stacked with 3 frames to the left and then downsampled by three to a 30ms frame rate.

The encoder network follows [9], consisting of 8 LSTM layers, where each layer has 2,048 hidden units followed by a 640-dimensional projection layer. We insert a time-reduction layer (by a factor of 2) after the second encoder LSTM layer. The RNN-T decoder contists of a prediction network and a joint network. The prediction network has 2 LSTM layers of 2,048 hidden units and a 640-dimensional projection per layer as well as an embedding layer of 128 units. The outputs of encoder and prediction network are fed to a 640 hidden unit joint network. The additional LAS-specific AE consists of 2 LSTM layers. The LAS decoder consists of multi-head attention [29] with four attention heads, computing context vectors which are fed into 2 LSTM layers of 2,048 hidden units with 640-dimensional projection. It has an embedding layer of 96 units. Both decoders are trained to predict 4,096 word pieces [23], which are derived from a large corpus of text transcripts.

The total size of the RNN-T model is 114M parameters, and the second-pass LAS decoder is 33M parameters. All models are trained in Tensorflow [30] using the Lingvo [31] toolkit on a v2-128 Cloud TPU slice with a global batch size of 4,096. Finally, during inference the interpolation weight λ in Equation 1 is swept on each model to optimize performance for SU and LU.

4. RESULTS

4.1. Model Analysis

In this section, we analyze the behavior of the JATD model with respect to training and inference. Note that in the tables below, boldface represents the model with the best overall performance.

4.1.1. Individual Log-Probability

First, we explore behavior when the training log-probability comes from only the acoustic or text data, as in Eq. 2. Table 1 shows performance as a function of α, which is the percentage of paired data seen during each epoch of training. For example $\alpha = 0.75$ means that 75% of the time paired data is seen in training, and 25% of the time unpaired data is seen. $\alpha = 1.0$ means that the model is trained only on paired data, similar to [9], and therefore comprises a baseline, $B1$. RNN-T only, without any rescoring, is also shown as baseline $B0$. As expected, LAS rescoring $B1$ improves over RNN-T only ($B0$). However, across all data-sets, including more unpaired data in the JATD model ($E0 - E2$) does not improve WER.

ID	Model	SU	LU	SxS	Corr	App	Songs
B0	RNN-T	6.9	4.9	31.8	15.3	8.9	15.4
B1	$\alpha = 1.0$	**5.9**	**3.7**	**29.6**	**14.3**	**8.7**	**13.2**
E0	$\alpha = 0.9$	6.0	3.6	29.4	14.1	8.7	12.8
E1	$\alpha = 0.75$	6.1	3.7	29.5	14.2	8.6	12.8
E2	$\alpha = 0.5$	6.2	3.7	29.6	14.3	8.6	12.8

Table 1: LAS Rescoring WER, Individual Training Log-Prob.

4.1.2. *Joint Log-Probability*

Next, Table 2 shows performance when the training loss interpolates between acoustic and text logits, as in Eq. 3. For inference $\lambda = 0.05$ was found to be optimal for all models. Decreasing α to 0.75 (E4) results in between 3-10% relative improvement across the proper noun sets, without degrading SU and LU. However, including too much unpaired data ($\alpha < 0.75$) naturally degrades WER. Comparing the results to those in Table 1, it is clear that joint interpolation during training (Eq. 3) is a better approach to training the JATD model compared to using individual log-probs (Eq. 2).

ID	Model	SU	LU	SxS	Corr	App	Songs
E3	$\alpha = 0.9$	6.0	**3.6**	28.6	14.1	8.3	12.4
E4	$\alpha = 0.75$	**6.0**	3.7	**28.2**	**13.9**	**8.2**	**11.9**
E5	$\alpha = 0.5$	6.1	3.7	28.5	14	8.2	12.2
E6	$\alpha = 0.25$	6.3	3.8	28.6	14.2	8.1	12.1

Table 2: LAS Rescoring WER, Joint Log-Prob.

4.1.3. *Increasing Model Capacity*

Because the LAS decoder is now used to model both paired and unpaired data, we evaluate whether increasing the model capacity can further improve performance. Table 3 shows the performance when doubling the number of layers in the LAS decoder to 4 layers ($\sim$ 66M parameters). Baselines $B1$ and $B2$ compare performance with $\alpha = 1.0$ for both a small (S) and large (L) models respectively. The table indicates that the larger model gives very little performance improvement when only using paired data. However, when making use of unpaired data, much larger improvements are seen, particularly in the SXS and Songs test sets, compared to the best smaller model ($E4$). In addition, much less degradation is seen now when training with more unpaired data ($\alpha < 0.75$) compared to smaller models in Table 2. Overall $E8$ with $\alpha = 0.5$ performs best, resulting in 3-10% relative improvement over baseline $B2$.

4.2. Rescoring vs. Beam Search

Using the JATD model as a rescorer handicaps the model, as it is only able to rescore hypotheses emitted by the first-pass RNN-T decoder, which is not trained with the unpaired data. To further study the potential benefit of the unpaired data with the JATD model, we explore running the second-pass LAS model within the beam search. Table 4 shows the results for JATD as a function of α. Baseline $B4$ uses no unpaired data. As α decreases the WER continues to improve in the proper noun sets. Overall the best performing model uses $\alpha = 0.5$ ($E11$), which gives the best tradeoff between proper-noun WER without hurting performance on LU and SU. Overall, $E11$ results in 3-12% relative improvement over baseline $B4$.

4.3. Analysis

4.3.1. *Comparison to Other Techniques*

Finally, we compare JATD to other techniques that incorporate unpaired training data without increasing model size in Table 5. Shallow-fusion results are not reported as this increases model size. We explored training the LAS decoder with paired and unpaired data from scratch (B5) with a 50%-50% mix, just as in JADT, rather than using multiple training stages as in [14]. When training in this manner, the WER degrades significantly as is expected since the unpaired data has a mistmatch between the audio and text. A similar trend

ID	Model	SU	LU	SxS	Corr	App	Songs
B1	$\alpha = 1.0$, S	5.9	3.7	29.6	14.3	8.7	13.2
B2	$\alpha = 1.0$, L	5.8	3.6	29.2	14.3	8.7	13.2
E4	$\alpha = 0.75$, S	6.0	3.7	28.2	13.9	8.2	11.9
E7	$\alpha = 0.75$, L	5.8	3.7	27.3	13.9	8.1	11.9
E8	$\alpha = 0.5$, L	**5.8**	**3.6**	**27.4**	**13.8**	**8.0**	**11.9**
E9	$\alpha = 0.25$, L	6.0	3.6	27.4	14.1	8.1	11.9

Table 3: LAS Rescoring WER, Joint Log-Prob, larger model size.

ID	Model	SU	LU	SxS	Corr	App	Songs
B0	RNN-T	6.9	4.9	31.8	15.3	8.9	15.4
B4	$\alpha = 1.0$, L	5.6	3.5	26.5	14.4	8.6	12.5
E10	$\alpha = 0.75$, L	5.7	3.5	24.3	14.0	8.1	11.3
E11	$\alpha = 0.5$, L	**5.6**	**3.5**	**23.2**	**14.0**	**7.9**	**11.1**
E12	$\alpha = 0.25$, L	5.8	3.7	22.8	14.0	7.9	11.3

Table 4: LAS Beam Search WER, Joint Log-Prob, larger model size.

was seen with cycle-consistency (CC). Overall, JADT offers the best tradeoff between WER improvements without increasing training time or model size.

ID	Model	SU	LU	SxS	Corr	App	Songs
B4	LAS, only [9]	**5.6**	**3.5**	26.5	14.4	8.6	12.5
B5	Unsup [14]	6.5	6.3	26.4	15	8.9	14.7
E11	JATD	**5.6**	**3.5**	23.2	**14.0**	**7.9**	**11.1**

Table 5: LAS Beam Search Baselines

4.3.2. *Proper Noun Analysis*

Table 6 shows some example decodes from the JATD model ($E8$) which improve over the baseline LAS decoder ($B1$). The top two rows show language-modeling errors which are be corrected by the JATD model, and the bottom two rows demonstrate corrections to proper noun errors are.

LAS	JATD
How do you **bake** a hook for bass fishing	How do you bait a hook for bass fishing
peanut butter cookies **and** scratch	peanut butter cookies from scratch
Houston **Astro** cap	Houston Astros cap
What is **ligonberry**	What is lingonberry

Table 6: JATD wins. LAS errors indicated in **red**.

To be a bit more quantitative, we also explore the errors to due rare words across the sets sets. Here a word is defined as *rare* if it has less than a count of 10 in the training data. Table 7 shows that by including unsupervised data, the % of errors due to rare-words decreases by more than 20% relative.

ID	Model	SU	LU	SxS	Corr	App	Songs
B2	LAS	6.4	4.9	7.6	17.6	10.0	6.3
E8	JATD	4.5	3.9	5.0	11.7	4.7	5.1

Table 7: % of Errors due to Rare Words

5. ACKNOWLEDGEMENTS

The authors would like to thank Rohit Prabhavalkar for helpful discussions and comments regarding this work.

6. REFERENCES

[1] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[2] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, N. Jaitly, B. Li, and J. Chorowski, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[3] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[4] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep neural networks," in *Proc. ICASSP*, 2012.

[5] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

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

[7] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[8] C.-C. Chiu and C. Raffel, "Monotonic chunkwise alignments," in *Proc. ICLR*, 2017.

[9] T.N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu, I. McGraw, and C.C Chiu, "Two-Pass End-to-End Speech Recognition," in *Proc. Interspeech*, 2019.

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

[11] A. Sriram, H. Jun, S. Sateesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *CoRR*, vol. abs/1708.06426, 2017.

[12] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, 2018.

[13] Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengkun Tian, and Zhengqi Wen, "Learn Spelling from Teachers: Transferring Knowledge from Language Models to Sequence-to-Sequence Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 3795–3799.

[14] B. Li, T.N. Sainath, R. Pang, and Z. Wu, "Semi-supervised Training for End-to-End Models Via Weak Distillation," in *Proc. ICASSP*, 2019.

[15] R. Sennrich, B. Haddow, and A. Birch, "Improving Neural Machine Translation Models with Monolingual Data," in *ACL*, 2016.

[16] D. Zhao, T. N. Sainath, D. Rybach, D. Bhatia, B. Li, and R. Pang, "Shallow-Fusion End-to-End Contextual Biasing," in *submitted to Proc. Interspeech*, 2019.

[17] T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux, "Cycle-Consistency Training for End-to-End Speech Recognition," in *Proc. ICASSP*, May 2019, pp. 6271–6275.

[18] Yi Ren, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, "Almost unsupervised text to speech and automatic speech recognition," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 5410–5419.

[19] B. Li, T. N. Sainath, K. Chai Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao, "Multi-Dialect Speech Recognition With A Single Sequence-To-Sequence Model," in *ICASSP*, 2018.

[20] S. Toshniwal, T. Sainath, R. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual Speech Recognition with a Single End-to-End Model," in *ICASSP*, 2018.

[21] S. Kim and M. Seltzer, "Towards Language-Universal End-to-End Speech Recognition, booktitle = Proc. ICASSP, year = 2018,," .

[22] A. Renduchintala, S. Ding, M. Wiesner, and S. Watanabe, "Multi-Modal Data Augmentation for End-to-end ASR," in *Proc. Interspeech*, 2018.

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

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

[25] A. Narayanan, R. Prabhavalkar, C.C. Chiu, D. Rybach, T.N. Sainath, and T. Strohman, "Recognizing Long-Form Speech Using Streaming End-to-End Models," in *to appear in Proc. ASRU*, 2019.

[26] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated 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.

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

[28] A. van den Oord, Y. Li, and I. Babuschkin et. al., "Parallel wavenet: Fast high-fidelity speech synthesis," Tech. Rep., Google Deepmind, 2017.

[29] 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.

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

[31] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, et al., "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," 2019.