BENCHMARKING LF-MMI, CTC AND RNN-T CRITERIA FOR STREAMING ASR

Xiaohui Zhang⋆, Frank Zhang⋆, Chunxi Liu⋆, Kjell Schubert, Julian Chan, Pradyot Prakash, Jun Liu, Ching-Feng Yeh, Fuchun Peng, Yatharth Saraf, Geoffrey Zweig

Facebook AI, USA

ABSTRACT

In this work, to measure the accuracy and efficiency for a latency-controlled streaming automatic speech recognition (ASR) application, we perform comprehensive evaluations on three popular training criteria: LF-MMI, CTC and RNN-T. In transcribing social media videos of 7 languages with training data 3K - 14K hours, we conduct large-scale controlled experimentation across each criterion using identical datasets and encoder model architecture. We find that RNN-T has consistent wins in ASR accuracy, while CTC models excel at inference efficiency. Moreover, we selectively examine various modeling strategies for different training criteria, including modeling units, encoder architectures, pre-training, etc. Given such large-scale real-world streaming ASR application, to our best knowledge, we present the first comprehensive benchmark on these three widely used training criteria across a great many languages.

Index Terms— LF-MMI, CTC, RNN-T, latency-controlled ASR

1. INTRODUCTION

Thus far there has been a growing interest in speech community to investigate ASR modeling techniques that allow for flat-start (alignment-free) training, e.g., connectionist temporal classification (CTC) [1], recurrent neural network transducer (RNN-T) [2], and attention-based sequence-to-sequence (seq2seq) models [3, 4]. Specifically, CTC criterion learns an encoder-only model, which can be further composed with an n-gram language model (LM) in a standard WFST framework. RNN-T or attention-based seq2seq model jointly learns an encoder with a neural decoder model that can be considered as a neural LM.

There have been extensive ASR benchmarks on the public dataset like LibriSpeech [5]. Recently deep transformer based hybrid models have achieved state-of-the-art results among cross-entropy (CE) [6] and CTC based models [7] respectively. Improving accessibility to social media videos remains an important task, which allows for various applications like automatic video captioning, indexing and retrieval. Transcribing the heterogeneous social media videos of extensively diverse languages is still highly challenging, and prior works [8, 9, 10, 11, 7, 12] have examined various ASR technologies in both real-world high- and low-resource scenarios. In this work, we particularly focus on *streaming* and *long-form* ASR solutions, where each test utterance is up to 45 seconds long. While attention-based seq2seq models have shown difficulty in generalizing well on long utterances given the previous study [10], we thus do not include attention-based seq2seq model in this study.

To understand the performance distinction of different model types, it is essential to examine whether the model architecture or the training criterion is accountable. This is typically a challenging task, since the model architectures for differing training criteria are intrinsically different, which makes the training criteria not directly comparable. However, we note that, for all these training criteria, the majority of model parameters resides in the encoder part. In order to understand the relative performance difference in a relatively fair manner, we can fix the encoder model architecture for each training criterion to exclude the respective encoder impact on ASR performance. Therefore, in this work, we focus on comparing RNN-T and CTC criteria against a strong hybrid LF-MMI [13] baseline, and all cases use the same streamable encoder architecture. We evaluated the modeling performance by word error rate (WER) and decoding efficiency by real-time factor (RTF) of models trained by all three training criteria on video ASR tasks of 7 languages with training data ranging from $3K$ to $14K$ hours, with test sets of three noisy levels for each language. To the best of our knowledge, there has not been such comprehensive study thus far. For example, prior work [14] provided comparisons among CTC, RNN-T, and attention-based seq2seq models, while not including the standard hybrid ASR model. Recent work in [10, 15] has compared RNN-T with attention-based seq2seq models with the same encoder architecture; however, they used different encoder architectures in comparison to hybrid and CTC models. Besides, [16] has compared RNN-T with LF-MMI with the same encoder architecture on English.

Additionally, as we understand that RNN-T relies on a neural decoder - which can be seen as an implicit neural LM - in the first-pass decoding, however, all model types are able to explicitly perform additional online neural LM rescoring in the first-pass decoding (e.g. shallow fusion [17]). So we further exclude any additional LM rescoring effect in this study.

Besides the comprehensive benchmark of three training criteria, we make further contributions by selectively specifying how we reach the best modeling strategies for each criterion. We present studies on: (i) the choice of modeling units, wordpiece v.s. chenone, and (ii) encoder model architectures, latency-controlled bi-directitonal LSTM (LC-BLSTM) v.s. time-depth separable (TDS) convolutions for CTC models of three languages. We also present optimization efforts - reducing memory consumption and model pre-training - for RNN-T training, which lead to improvements in both modeling performance and training efficiency.

2. MODEL TRAINING

For each language we evaluated on, all models were trained on the same data segmented to up-to 10s, which was achieved by force aligning the whole audio against the reference using the same cross-entropy (CE)-trained model. Segmenting training data could substantially improve the training throughput, and slightly improve the accuracy as shown in [6]. LF-MMI models were pre-trained with CE criterion on 10s segments, and then fine-tuned with LF-MMI criterion on 1.5s

⋆ Equal contribution.

segments. CTC models were trained on 10s segments directly. The encoder in RNN-T models were pre-trained with CE criterion [15] and then the whole model was fine-tuned with RNN-T criterion, all on 10s segments. The chunk size used during training is 1.28s for CTC and RNN-T, and 1.5s for LF-MMI. The right context in training is 210ms for LF-MMI and 240ms for CTC and RNN-T[1].

Here we briefly review the three training criteria which are studied in this paper. ASR can be formulated as a sequence-to-sequence problem. Each speech utterance is parameterized as an input acoustic feature vector sequence $\mathbf{x} = \{\mathbf{x}_1 \dots \mathbf{x}_T\} = \mathbf{x}_{1:T}$, where $\mathbf{x}_t \in \mathbb{R}^d$ and T is the number of frames in $\mathbf{x}$. We define a grapheme set or a wordpiece inventory as $\mathcal{Y}$, and the corresponding target sequence of length U as $\mathbf{y} = \{y_1 \dots y_U\} = \mathbf{y}_{1:U}$, where $y_u \in \mathcal{Y}$. Besides, we define $\bar{\mathcal{Y}}$ as $\mathcal{Y} \cup \{\emptyset\}$, where $\emptyset$ is the blank label, and $\bar{\mathcal{Y}}^*$ as the set of all sequences over output space $\bar{\mathcal{Y}}$.

2.1. LF-MMI

The MMI objective can be formulated as:

$$F_{MMI} = log\frac{p(\mathbf{x}|\mathbf{y})}{\sum_{\hat{\mathbf{y}}} p(\mathbf{x}|\hat{\mathbf{y}})} \approx log\frac{p(\mathbf{x}|\mathbb{G}_{num})}{p(\mathbf{x}|\mathbb{G}_{den})} \tag{1}$$

where $\hat{\mathbf{y}}$ represents any possible hypothesis. In LF-MMI [13], a composite HMM graph called "denominator graph" $\mathbb{G}_{den}$ is used to approximate the denominator, which encodes all possible hypothesis, and thereby we have $\sum_{\hat{\mathbf{y}}} p(\mathbf{x}|\hat{\mathbf{y}}) \approx p(\mathbf{x}|\mathbb{G}_{den})$. Efficient computation of the denominator without having to generate lattices is enabled by adopting an n-gram phone/character language model (LM) when generating $\mathbb{G}_{den}$, and doing full forward-backward computation on GPUs. Similarly, the numerator $p(\mathbf{x}|\mathbf{y})$ is approximated by $p(\mathbf{x}|\mathbb{G}_{num})$ where $\mathbb{G}_{num}$ is another composite HMM graph called "numerator graph", encoding all possible sequences of HMM states pertaining to the transcription $\mathbf{y}$. It could be either an acyclic graph encoding pre-computed alignments, giving regular LF-MMI (used in this work), or a graph with self-loops determined solely by reference transcripts, giving flat-start (alignment-free) LF-MMI.

2.2. CTC

As a sequence-level training criterion, for CTC, the log-likelihood of a given target sequence $\mathbf{y}$ can then be found by summing the probabilities of all allowed alignments. Specifically,

$$\log p(\mathbf{y}|\mathbf{x}_{1:T}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{t=1}^{t=T} p(\mathbf{a}_t|\mathbf{x}_t) \tag{2}$$

where $\mathcal{B} : \bar{\mathcal{Y}}^* \rightarrow \mathcal{Y}^*$ is a mapping operation that removes all blank labels and repeating symbols in a given sequence. The encoder is trained to maximize the log-likelihood for each training example and p can be computed efficiently using the forward-backward algorithm.

Note that the underlying assumption in Eq (2) is that probabilities between timestamps are conditional independent. The Transducer criterion introduced in the next section will lift this constraint.

[1]The difference in right context is due to it needs to be divisible by the stride e.g. 3, 4 or 8. Empirically the slight different in right context did not affect final WER and real time factor.

2.3. RNN-T

Excluding the conditional independence assumption made in CTC, RNN-T models the posterior probability as:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} P(\mathbf{a}|\mathbf{x}) \tag{3}$$

where $\mathcal{B} : \bar{\mathcal{Y}}^* \rightarrow \mathcal{Y}^*$ is a mapping operation that removes all blank labels in a given sequence. RNN-T model parameterizes the alignment probability $P(\mathbf{a}|\mathbf{x})$ and computes it with an encoder network (i.e. transcription network in [2]), a prediction network and a joint network. The encoder performs a mapping operation, denoted as f^{enc}, which converts $\mathbf{x}$ into another sequence of representations $\mathbf{h}^{\text{enc}} = \{\mathbf{h}^{\text{enc}}_1 \dots \mathbf{h}^{\text{enc}}_{T'}\}$:

$$\mathbf{h}^{\text{enc}} = f^{\text{enc}}(\mathbf{x}) \tag{4}$$

where T' is equal or shorter than T due to subsampled frame rate. A prediction network f^{pred}, based on RNN or its variants, takes both its state vector and the previous non-blank output label y_{u-1} predicted by the model, to produce the new representation $\mathbf{h}^{\text{pred}}$:

$$\mathbf{h}^{\text{pred}}_{1:u} = f^{\text{pred}}(y_{0:(u-1)}) \tag{5}$$

where u is the output label index and $y_0 = \emptyset$. Finally, the joint network f^{join} is a feed-forward network that combines the encoder output $\mathbf{h}^{\text{enc}}_t$ and prediction network output $\mathbf{h}^{\text{pred}}_u$ to compute logits $\mathbf{z}_{t,u}$, which go through a softmax function and produce a posterior distribution of the next output label over $\bar{\mathcal{Y}}$:

$$\mathbf{z}_{t,u} = f^{\text{join}}(\mathbf{h}^{\text{enc}}_t, \mathbf{h}^{\text{pre}}_u) \tag{6}$$

$$p(y_u|\mathbf{x}_{1:t}, y_{1:(u-1)}) = \text{Softmax}(\mathbf{z}_{t,u}) \tag{7}$$

The encoder can be seen as an acoustic model, and the combination of prediction and joint network as a decoder.

3. MODELING UNITS

For the LF-MMI criterion, we used tied context-dependent grapheme states (i.e. chenones) [18] with a stride (sub-sampling factor) of 3. For CTC criterion, we used wordpiece units with a stride of 8. For RNN-T criterion, we used wordpiece units with a stride of 4. For each training criterion, the choice of modeling units, and stride were tuned separately on validation data to achieve the best balance between WER and inference efficiency. In Section 6.3, we will specify our analysis in the modeling unit options for CTC. Besides, the size of the chenone set (i.e. decision tree size) or wordpiece vocabulary were tuned to optimize WER on each language for each model, also on validation data.

4. MODEL ARCHITECTURE

We keep the encoder architecture fixed when comparing performance across different training criteria. In the main experiment, we used a latency-controlled bi-directitonal LSTM (LC-BLSTM) encoder with 5 layers of 800 hidden units. Sub-sampling along the time dimension by a factor of 3 is applied at the output of the first layer to achieve a stride of 3 for LF-MMI models, and sub-sampling by a factor of 2 is applied at the output of first, second or third layer to achieve a stride of 4 or 8 respectively for RNN-T and CTC models. The encoder alone has around 75M parameters. All the models presented here can run in a streaming fashion, because of the limited right context.

5. MODEL INFERENCE

For LF-MMI and chenone-CTC models, we pre-built decoding graphs $\mathrm{H} \circ \mathrm{C} \circ \mathrm{L}$ and G [2] and dynamically composed them during decoding [19]. For wordpiece-CTC models, we use the same dynamic decoding approach but pre-built $\mathrm{H} \circ \mathrm{L}$ rather than $\mathrm{H} \circ \mathrm{C} \circ \mathrm{L}$ [7]. For RNN-T model, we use standard beam search decoding without any external LM fusion as in [2], since we are evaluating the three training criteria under a "vanilla" single-pass inference setting without any LM fusion/rescoring for all models. All acoustic models were trained in PyTorch and applied post-training INT8 quantization to enable efficient decoding. Decoding hyper-parameters, e.g. beam sizes, were tuned on validation data for each model separately, to achieve a balance between decoding efficiency and WER. To satisfy latency constraints for live captioning use-case, we limit the chunk size to 0.8s for English, Spanish, Hindi and Indic English, and 1.5s for Thai, Vietnamese and Turkish across all models.

6. EXPERIMENTS

6.1. Data

We evaluate all models on our in-house Video ASR datasets, which are sampled from public social media videos and completely de-identified before transcription; both transcribers and researchers do not have access to any user-identifiable information (UII). These videos contain a diverse range of speakers, accents, topics, and acoustic conditions making automatic recognition very challenging. We included a wide variety of languages in this study in order to get a broad understanding of model performance, including: (i) fusional languages Spanish (ES), Hindi (HI) and Indic English (EN-IN), (ii) analytic languages US English (EN-US), Vietnamese (VN) and Thai (TH), and (iii) an agglutinative language Turkish (TR). The training set sizes are shown in Table 1. Note that we combined Hindi and Indic English training data and trained a single model for each criterion, although we evaluate the model on Hindi and Indic English test sets separately. The reason is that due to their similarity in pronunciation and frequent code-switching, it can be hard for a language identification (LID) model to differentiate acoustic inputs from these languages. In addition, some special text processing was applied to Thai: as reference transcripts were un-segmented (no word-level tokenization), we needed to tokenize the transcripts at wordpiece level by training a wordpiece model first, and then construct a lexicon mapping wordpieces to graphemes for data segmentation, model training and decoding. Regarding data augmentation, speed perturbation [20] and *SpecAugment* [21] (LD policy for RNN-T and SM policy for CTC and LF-MMI) are used.

The test sets for each language are composed of `clean`, `noisy` and `extreme` categories, with `extreme` being the most acoustically challenging. The validation set for each language was composed of data from the `noisy` category. The duration of validation and test sets for each category in each language is around 10 to 40 hours. All validation and test data were segmented up to 45s.

Table 1. Training data sizes (in hours).

EN-US	ES	HI & EN-IN	TH	VN	TR
14K	7.2K	6.7K	5.1K	4.2K	3.1K

[2]H transduces HMM states to context-dependent graphemes; C transduces context-dependent graphemes to graphemes; L transduces graphemes to words; G represents the language model.

6.2. Results

In this section, we first present decoding results on all 7 languages in Table 2, with the best overall modeling strategies (modeling unit, stride, and pre-training strategy) chosen for each training criterion. For each language, all models were trained on the same data with the same encoder model architecture (5×800 LC-BLSTM). Later, we will selectively analyze the impact of modeling strategies and encoder model architecture for CTC/RNN-T. We use word error rate (WER), or character error rate (CER) for Thai, to measure modeling performance on `clean`, `noisy` and `extreme` test splits for each language and model. We use real-time factor (RTF) to measure decoding efficiency.

6.3. WP-CTC v.s. chenone-CTC

It is well-studied that LF-MMI works well with chenone units and RNN-T works well with wordpiece units. For CTC training, we have observed that full sequence deep transformer encoder trained with chenone units outperforms wordpieces consistently. However, the trend is different for our streaming application in this work. In Table 3[3], we show results of WP-CTC (with a stride of 8) and chenone-CTC (with a stride of 4) on English, Vietnamese and Turkish, which are the best choices of stride for WP/chenone-CTC respectively, in terms of balancing WER and RTF. We find that for LC-BLSTM models, WP-CTC training consistently outperforms chenone-CTC in WER[4], and thus we decided to adopt WP-CTC when comparing it with other training criteria. One hypothesis is that the LC-BLSTM encoder being a streaming model is less expressive than a full-context deep transformer encoder. Therefore, the LC-BLSTM encoder is not able to fully exploit the richer target representation provided by chenone alignments during training, in a sense that the optimal size of a chenone is usually much larger than a wordpiece set.

6.4. Choices of the encoder model architecture

We also explored an encoder architecture based on time-depth separable (TDS) convolutions [22] as an alternative encoder choice under the CTC setup. Since from recent research [23] TDS encoder has shown its advantage of speed during inference, we want to explore if this architecture generalizes well on more datasets. The TDS architecture in this study is designed to use as many parameters as possible to improve WER given that the RTF is still lower than LC-BLSTM: the TDS encoder consists of 14 TDS blocks, 3 sub-sampling layers, each with a stride of 2, for a total sub-sampling factor of 8. The total right context is 570 ms. Total parameters is 122M and is larger than the LC-BLSTM model (75M parameters). The realized RTF of TDS on English (0.26 [23]) is lower than LC-BLSTM as in Table 2. Results on English, Vietnamese and Turkish languages are presented in Table 4. We can see that TDS WER slightly outperforms LC-BLSTM in English and lags behind in the other two languages. One possible explanation is that the TDS architecture is more data hungry, e.g. for English, there are more than 3 times the training data as Turkish and Vietnamese. There is also evidence that on LibriSpeech which has 1000hrs of training data, LC-BLSTM is outperforming TDS [22, 18]. Therefore, for the overall 7 languages comparisons, we used the LC-BLSTM encoder when comparing different training criteria.

[3]WERs of CTC in Table 3 and Table 4 are different from Table 2 due to some differences in evaluation datasets, however the same data were used consistently within each table.

[4]and also in RTF [7], though we did not measure RTF here.

Table 2. *Performance overview of WER (CER for Thai) and RTF. Average WERR (Word Error Rate Reduction, positive and larger is better) is computed by first computing the WERR on the three test categories individually (using LF-MMI models as a baseline), and then taking the unweighted average.*

Language	US English			Spanish			Hindi			Indic English		
Model	LF-MMI	CTC	RNN-T	LF-MMI	CTC	RNN-T	LF-MMI	CTC	RNN-T	LF-MMI	CTC	RNN-T
clean	10.4	11.3	10.2	10.4	10.2	9.1	20.1	18.9	17.9	26.9	26.7	26.2
noisy	14.4	15.0	14.2	12.7	12.6	11.1	21.7	20.6	19.4	31.6	31.1	31.3
extreme	20.3	20.9	19.8	21.0	20.7	19.2	25.7	26.3	25.0	32.2	32.7	31.3
Avg. WERR	–	-5.3%	1.9%	–	1.4%	11.2 %	–	2.9%	8.1%	–	0.3%	2.1%
RTF	0.46	0.40	0.49	0.50	0.33	0.48	0.44	0.30	0.41	0.44	0.30	0.41

Language	Thai			Vietnamese			Turkish		
Model	LF-MMI	CTC	RNN-T	LF-MMI	CTC	RNN-T	LF-MMI	CTC	RNN-T
clean	9.7	9.9	8.7	11.5	11.7	10.5	19.4	19.6	16.9
noisy	13.7	14.2	12.8	19.3	19.9	19.0	20.2	20.7	18.6
extreme	21.7	22.8	20.2	45.3	46.6	46.3	37.9	39.9	38.4
Avg. WERR	–	-3.6%	7.9%	–	-2.6%	2.6%	–	-2.9%	6.5%
RTF	0.41	0.29	0.40	0.37	0.29	0.44	0.45	0.33	0.43

Table 3. WER of WP-CTC v.s. chenone-CTC

Lang.	EN		VN		TR	
Unit	WP	chenone	WP	chenone	WP	chenone
clean	14.0	15.3	11.6	15.5	19.3	20.7
noisy	20.0	21.3	19.9	23.5	20.4	21.5
extreme	26.1	28.5	46.6	52.2	39.9	40.6

Table 4. WER of LC-BLSTM v.s. TDS encoder for CTC

Lang.	EN		VN		TR	
Unit	LC-BLSTM	TDS	LC-BLSTM	TDS	LC-BLSTM	TDS
clean	14.0	13.7	11.6	12.7	19.3	20.9
noisy	20.0	19.5	19.9	20.9	20.4	22.4
extreme	26.1	25.2	46.6	48.2	39.9	41.8

6.5. (Pre-)training optimization for RNN-T

One of the major challenges of training RNN-T models is the need of enormous memory size, due to the formulation on both embeddings from the encoder $\mathbf{h}_t^{\text{enc}}$ and the predictor $\mathbf{h}_u^{\text{pre}}$ as shown in Eq. 6. Specifically, in order to compute the forward-backward algorithm [2], a joint embedding $\mathbf{z}_{t,u}$ is needed for each position pair (t, u). This translates to a minimum memory usage of $T_i * U_i * D$ floating numbers for the i-th sequence in a sequence of batch size B, where T_i and U_i are sequence lengths of encoder/predictor embeddings and D is the number of sentence pieces as output units. This can in turn lead to $B * \max_i(T_i) * \max_i(U_i) * D$ floating numbers for the entire batch if with the more traditional "broadcasting" implementation, or the reduced $\sum_i T_i * U_i * D$ with optimization [24]. For either cases, the scale of such tensors is often measured in GBs, therefore limits the batch size, which is observed to be highly correlated with the stability of gradients and then the final word error rates.

With the identification of the bottleneck for training RNN-T models, our in-house RNN-T criterion implementation provides additional improvements on training efficiency and word error rate reduction with highly optimized memory consumption. First, *function merging* [24] was adopted to fuse the softmax operation into the RNN-T criterion, this reduces the memory usage by $\simeq 50\%$ (translating to 2x batch size) while the numerical value of gradients stay identical.

Table 5. *Training optimization and CE pre-training effects for RNN-T with varying training mini-batch size. WER results on Turkish (without model quantization and decoding beam sweeping).*

batch size	8		16	
pre-training	N	Y	N	Y
clean	17.7	17.0	17.1	16.8
noisy	19.3	19.0	18.9	18.9

Second, mixed-precision training was implemented in which 16-bit float numbers (fp16) are used instead of 32-bit ones (fp32), which leads to another $\simeq 50\%$ memory usage reduction (another 2x gain on batch size), with some loss on precision but compensated later by larger batch sizes. The combined optimizations improves the batch size by a factor of 4 compared over the vanilla implementation, and a factor of 2 over function merging alone, which leads to not only training speed-up but also performance gain.

With an output wordpiece size 2048, RNN-T training can only use a batch size 8 in each V100 GPU of 16G memory before the above training optimization. In such case, pretraining the RNN-T encoder with the hybrid CE model (i.e. the same model used in LF-MMI systems) has provided consistent performance improvements, as shown in Table 5. After the training optimization enables a batch size 16, we observe noticeable performance improvements without encoder pre-training, while additional pre-training with hybrid CE model only provides minor further gains.

7. DISCUSSIONS AND CONCLUSIONS

In this work, we demonstrated in details that across the 7 languages studied, CTC systems achieved best decoding efficiency while RNN-T systems provided best WER overall. Compared with the LF-MMI baseline, for CTC, the RTF improvement is around 30% with 2 - 5% WER degradation for 4 languages and up to 3% WER improvement for the other 3 languages; for RNN-T, the RTF is about the same as baseline LF-MMI systems for all languages, with significant and consistent 2 - 11% WER improvements.

Overall, CTC systems were able to achieve the best decoding efficiency since they use wordpieces (WP) units (spanning longer temporal space than chenone) with the largest stride (8) among all

systems. This agrees with findings from other literature that CTC can achieve a good balance between WER and decoding inference efficiency when using wordpiece [25, 26, 7] or even whole words [9] as modeling units. RNN-T systems consistently achieve the best WERs across all languages, presumably due to to its expressiveness in explicitly leveraging previous output labels, as shown in Eq. 7.

In future work, we will continue to measure the performance on named entities (i.e. entity error rate), and present studies on the ASR inference latency, i.e., delayed token generation problem [27, 28]. We will also examine various model-specific techniques that can improve a model type in particular [29, 30, 31], and continue to benchmark the best systems across training criteria.

In summary, each system explored in this study - LF-MMI, CTC, or RNN-T - has its own strengths and limits, and accordingly each could be adopted based on different business requirements, e.g. prioritizing run time over WER, or vise versa.

8. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. ICML*, 2006.

[2] Alex Graves, "Sequence Transduction with Recurrent Neural Networks," in *Proc. ICML*, 2012.

[3] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, et al., "Attention-based models for speech recognition," in *Proc. NueralIPS*, 2015.

[4] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016.

[5] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015.

[6] Yongqiang Wang, Abdelrahman Mohamed, Duc Le, Chunxi Liu, et al., "Transformer-based acoustic modeling for hybrid speech recognition," in *Proc. ICASSP*, 2020.

[7] Frank Zhang, Yongqiang Wang, Xiaohui Zhang, Chunxi Liu, et al., "Faster, simpler and more accurate hybrid ASR systems using wordpieces," in *Proc. Interspeech*, 2020.

[8] Hank Liao, Erik McDermott, and Andrew Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Proc. ASRU*, 2013.

[9] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," *Proc. Interspeech*, 2017.

[10] Chung-Cheng Chiu, Wei Han, Yu Zhang, Ruoming Pang, et al., "A comparison of end-to-end models for long-form speech recognition," in *Proc. ASRU*, 2019.

[11] Chunxi Liu, Qiaochu Zhang, Xiaohui Zhang, Kritika Singh, Yatharth Saraf, and Geoffrey Zweig, "Multilingual graphemic hybrid ASR with massive data augmentation," in *Proc. of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, 2020.

[12] Da-Rong Liu, Chunxi Liu, Frank Zhang, Gabriel Synnaeve, Yatharth Saraf, and Geoffrey Zweig, "Contextualizing ASR lattice rescoring with hybrid pointer network language model," in *Proc. Interspeech*, 2020.

[13] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, et al., "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016.

[14] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *Proc. ASRU*, 2017.

[15] Jinyu Li, Yu Wu, Yashesh Gaur, Chengyi Wang, et al., "On the comparison of popular end-to-end models for large scale speech recognition," in *Proc. Interspeech*, 2020.

[16] Mahaveer Jain, Kjell Schubert, Jay Mahadeokar, Ching-Feng Yeh, Kaustubh Kalgaonkar, Anuroop Sriram, Christian Fuegen, and Michael L. Seltzer, "Rnn-t for latency controlled asr with improved beam search," 2019.

[17] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, 2018.

[18] Duc Le, Xiaohui Zhang, Weiyi Zheng, Christian Fügen, et al., "From Senones to Chenones: Tied Context-Dependent Graphemes for Hybrid Speech Recognition," *Proc. ASRU*, 2019.

[19] Jun Liu, Jiedan Zhu, Vishal Kathuria, and Fuchun Peng, "Efficient dynamic wfst decoding for personalized language models," *arXiv preprint arXiv:1910.10670*, 2019.

[20] T. Ko, V. Peddinti, D. Povey, et al., "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015.

[21] D. S. Park, W. Chan, Y. Zhang, et al., "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019.

[22] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," in *Proc. Interspeech*, 2019.

[23] Vineel Pratap, Qiantong Xu, Jacob Kahn, Gilad Avidov, et al., "Scaling Up Online Speech Recognition Using ConvNets," 2020.

[24] Jinyu Li, Rui Zhao, Hu Hu, and Yifan Gong, "Improving RNN Transducer Modeling for End-to-End Speech Recognition," in *Proc. ASRU*. IEEE, 2019.

[25] M. Huang, Y. Lu, L. Wang, Y. Qian, et al., "Exploring Model Units and Training Strategies for End-to-End Speech Recognition," in *Proc. ASRU*, 2019.

[26] Amit Das, Jinyu Li, Guoli Ye, Rui Zhao, et al., "Advancing Acoustic-to-Word CTC Model with Attention and Mixed-Units," *IEEE TASLP*, 2018.

[27] Hirofumi Inaguma, Yashesh Gaur, Liang Lu, Jinyu Li, and Yifan Gong, "Minimum latency training strategies for streaming sequence-to-sequence ASR," in *Proc. ICASSP*, 2020.

[28] Jay Mahadeokar, Yuan Shangguan, Duc Le, Gil Keren, Hang Su, Thong Le, Ching-Feng Yeh, Christian Fuegen, and Michael Seltzer, "Alignment restricted streaming recurrent neural network transducer," in *Proc. SLT*, 2021.

[29] Andros Tjandra, Chunxi Liu, Frank Zhang, Xiaohui Zhang, Yongqiang Wang, Gabriel Synnaeve, Satoshi Nakamura, and Geoffrey Zweig, "Deja-vu: Double feature presentation and iterated loss in deep transformer networks," in *Proc. ICASSP*, 2020.

[30] Chunxi Liu, Frank Zhang, Duc Le, Suyoun Kim, Yatharth Saraf, and Geoffrey Zweig, "Improving RNN transducer based ASR with auxiliary tasks," in *Proc. SLT*, 2021.

[31] Ashutosh Pandey, Chunxi Liu, Yun Wang, and Yatharth Saraf, "Dual application of speech enhancement for automatic speech recognition," in *Proc. SLT*, 2021.