EXPLORING MODEL UNITS AND TRAINING STRATEGIES FOR END-TO-END SPEECH RECOGNITION

Mingkun Huang[1], *Yizhou Lu*[1], *Lan Wang*[2], *Yanmin Qian*[1], *Kai Yu*[1]

[1]MoE Key Lab of Articial Intelligence
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
{mingkunhuang,luyizhou4,yanminqian,kai.yu}@sjtu.edu.cn

ABSTRACT

In this work, we explore end-to-end speech recognition models (CTC, RNN-Transducer and attention-based models) with different model units (character, wordpiece and word) and various training strategies. We show that wordpiece unit outperforms character unit for all end-to-end systems on the Switchboard Hub5'00 benchmark. To improve the performance of end-to-end systems, we propose a multi-stage pretraining strategy, which gives 25.0% and 18.0% relative improvements over training from scratch for attention and RNN-T models respectively with wordpiece units. We achieve state-of-the-art performance on the Switchboard+Fisher-2000h task, outperforming all prior work. Together with other training strategies such as label smoothing and data augmentation, we achieve 5.9%/12.1% WER on the Switchboard/CallHome test set without using any external language models. This is a new performance milestone for a single end-to-end system, and it is also much better than the previous published best hybrid system, which is 6.7%/12.5% on each set individually.

Index Terms— end-to-end, sequence-to-sequence models, speech recognition, word piece

1. INTRODUCTION

Automatic speech recognition (ASR) with deep neural networks commonly operates in a hybrid framework with Hidden Markov Model (HMM). In the inference stage, external lexicons and language models are combined with acoustic model, and all of these models are optimized independently [1]. End-to-end speech recognition is a popular approach that directly transcribes speech to text without requiring predefined alignment between acoustic frames and words [2, 3, 4, 5, 6]. Unlike hybrid approaches, the end-to-end model learns a mapping from acoustic frames to word sequence in one step, which directly optimizes the probability of words given input speech observations.

Recent work on end-to-end speech recognition can be categorized into three main approaches: Connectionist Temporal Classification (CTC) [7, 8], RNN-Transducer [9, 10, 11] and attention-based models [4, 5, 6, 12, 13]. These methods address the problem of variable-length input and output sequences. CTC makes an assumption that predictions are conditional independence given input frames at different time steps. This is not a reasonable assumption for speech recognition. RNN-Transducer and attention models use an auxiliary decoder network to build connections between predictions. Therefore, attention-based encoder decoder network has become popular for both machine translation [14] and speech recognition [15, 16, 17, 18, 13, 19]. Such models are typically trained to output *character-based* units: graphemes, byte-pair encodings [20], or wordpieces [21], which allow the model to directly map the frame-level input audio features to the output word sequence, without using any external pronunciation lexicon. Thus, when using such wordpiece units, end-to-end speech recognition models [6] jointly learn the acoustic model, pronunciation model, and language model within a single neural network. In fact, such models outperform the conventional approach when trained on sufficiently large amounts of data [13]. [11] explored architectures and modeling units for RNN-Transducer and has been successfully applied to the real applications [22].

There are studies comparing these models at scale [17, 15]. Other works have compared modeling units on attention models [23] and RNN-Transducer [11] respectively. However, there are no conclusive studies comparing these models with various output units. Meanwhile, we find that it is still hard to train a state-of-the-art end-to-end system from scratch.

In this work, we investigate training end-to-end speech recognition models with various model units: character, wordpiece and word. And we propose a multi-stage pre-

The corresponding authors are Yanmin Qian and Kai Yu. This work was supported by the China NSFC projects (No. 61603252 and No. U1736202) and joint project from Shenzhen Institutes of Advanced Technology in CAS. Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

training strategy, namely, initialize encoder and decoder network from CTC acoustic model and CE language model respectively. Specifically, for encoder pre-training, we start from CTC phone model, then use this phone model to initialize CTC subword model. Our experiments show that this initialization strategy significantly outperforms training from scratch or even with a CTC phone model initialization. We first conduct our experiments on Switchboard dataset, and show that wordpiece unit outperforms character unit for all end-to-end systems on the Hub5'00 benchmark. To improve the performance of our end-to-end systems, we also investigate other training strategies such as label smoothing and data augmentation. We achieve 5.9%/12.1% WER on the Switchboard/CallHome test set without using any external language models, which are much better than the previous published best hybrid system with 6.7%/12.5% on each set individually.

The paper is organized as follows: in Section 2 we review the various sequence-to-sequence models. Then, in section 3, we describe the modeling units, architectures and training strategies. Section 4 presents our experiments on Switchboard and Fisher datasets and compares the speech recognition performance of various models. We analyze our results in Section 5 and draw a conclusion in Section 6.

2. REVISITING SEQUENCE-TO-SEQUENCE MODELS FOR ASR

In this section, we describe different approaches of sequence-to-sequence modeling for speech recognition. Given the raw speech waveform, a sequence of input acoustic features $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ are firstly extracted and then fed into the sequence-to-sequence model. The encoder of the sequence-to-sequence model maps acoustic features $\mathbf{x}$ into high level representations $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T'})$ with length T', which can be shorter than T if time-scale downsampling is applied in the encoder. The decoder of the model utilizes the encoded presentation $\mathbf{h}$ and outputs the predicted sequence. We denote the U-length output sequence as $\mathbf{y} = (y_1, y_2, \ldots, y_U)$, where $\mathbf{y}$ consists of characters, wordpieces, or words. Generally, sequence-to-sequence model needs to define the sequence probability of $P(\mathbf{y}|\mathbf{x})$, and the key challenge of sequence-to-sequence modeling is that the whole model acts not only as a classifier, but also an aligner to infer the alignment between $\mathbf{h}$ and target sequence $\mathbf{y}$.

2.1. Connectionist temporal classification (CTC)

CTC criterion [7] considers all possible alignments between inputs and outputs without requiring pre-segmented training data. As long as the overall output sequence is correct, it allows the model to make label predictions at any time in the input sequence. An augmented predicting *blank* unit Ø is introduced to complete the output sequence to the same length as $\mathbf{h}$. A many-to-one function $\mathcal{F}$, which works as first remove all repeating symbols and then remove all Ø units, is defined to determine the output sequence $\mathbf{y}$ from the potential path $\boldsymbol{\pi}$. Its inverse function $\mathcal{F}^{-1}$ is used to map the target sequence $\mathbf{y}$ to a corresponding set of paths.

With the mapping function and conditional independent assumption, CTC defines the posterior probability $P(\mathbf{y}|\mathbf{x})$ of output sequence $\mathbf{y}$ given acoustic sequence $\mathbf{x}$ as:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{F}^{-1}(\mathbf{y})} P(\boldsymbol{\pi}|\mathbf{h}) = \sum_{\boldsymbol{\pi}} \prod_{t=1}^{T'} P(\pi_t|\mathbf{h}_t) \quad (1)$$

A single softmax layer, which serves as a simple decoder in CTC model, is used to estimate the conditional probability $P(\pi_t|\mathbf{h}_t)$. Through forward-backward dynamic programming algorithm, the sequence probability in equation 1 can be calculated efficiently. CTC model is usually optimized to minimize the negative log probability of $P(\mathbf{y}|\mathbf{x})$ for all the training samples. A greedy algorithm can be used to decode CTC model by simply picking the most probable symbols at each timestep.

2.2. RNN-Transducer (RNN-T)

To model the dependencies between outputs symbols, RNN-Transducer [9, 10] augments CTC model with an additional prediction network. Denote Ø as an empty symbol, the prediction network takes all previous label predictions $(\text{Ø}, y_1, y_2, \ldots, y_{u-1})$ as input to compute the vector $\mathbf{p}_u$ for next prediction. This linguistic score $\mathbf{p}_u$ together with acoustic score $\mathbf{h}_t$ are fed into a joint network [9] to compute the transition probability $P(k|t,u)$, which can be computed as follows:

$$\mathbf{z}_{t,u} = tanh(\mathbf{W}_h \mathbf{h}_t + \mathbf{W}_p \mathbf{p}_u + \mathbf{b}) \quad (2)$$

$$\mathbf{e}_{t,u} = \mathbf{W}_z \mathbf{z}_{t,u} + \mathbf{d} \quad (3)$$

$$P(k|t,u) = \frac{exp(e^k_{t,u})}{\sum_{k'} exp(e^{k'}_{t,u})} \quad (4)$$

where $\mathbf{W}_h, \mathbf{W}_p, \mathbf{W}_z, \mathbf{b}, \mathbf{d}$ are parameters of the model. The k^{th} element of $\mathbf{e}_{t,u}$ is denoted as $e^k_{t,u}$ and $P(k|t,u)$ defines a distribution over all symbol units plus Ø.

The conditional probability $P(\mathbf{y}|\mathbf{x})$ can be efficiently calculated using a dynamic programming algorithm, which is similar to CTC criterion. For RNN-T model, the prediction relies on not only acoustic representations but also previous predictions. Thus the label context dependency can be better modeled even without an external language model.

2.3. Attention-based models

Attention-based models [5, 6] utilize acoustic representations in the decoder, and make prediction based on all previous in-

ference labels $\mathbf{y}_{<u}$:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{u} P(y_u|\mathbf{h}, \mathbf{y}_{<u}) \quad (5)$$

Unlike RNN-T models where acoustic features and linguistic features are modeled in separate networks and combined with a joint network, attention-based models use attention mechanism to generate weighted coefficients $\boldsymbol{\alpha}_u$ and only use the weighted average vector (attention vector) in the decoder. At each decoding step, only the most relevant features are used for prediction. That is, attention-based models generate a 'soft' alignment through the attention mechanism:

$$\mathbf{s}_u = RNN(\mathbf{s}_{u-1}, y_{u-1}) \quad (6)$$

$$\mathbf{c}_u = Attention(\mathbf{s}_u, \boldsymbol{\alpha}_{u-1}, \mathbf{h}) \quad (7)$$

$$P(y_u|\mathbf{h}, \mathbf{y}_{<u}) = MLP(\mathbf{s}_u, \mathbf{c}_u) \quad (8)$$

where $\mathbf{s}_u$ is the query key of attention and $\mathbf{c}_u$ the result attention vector, the previous predicted symbol y_{u-1} is fed in as one-hot vectors. Several attention mechanisms can be employed, we use *location-aware attention* [5] in our experiment to take account the previous alignment $\boldsymbol{\alpha}_{u-1}$, which can be calculated as follows:

$$\mathbf{c}_u = \sum_{t=1}^{T'} \alpha_{t,u} \mathbf{h}_t \quad (9)$$

$$\mathbf{f}_u = \mathbf{F} * \boldsymbol{\alpha}_{u-1} \quad (10)$$

$$e_{t,u} = \mathbf{w}^{\intercal} tanh(\varphi(\mathbf{h}_t) + \phi(\mathbf{s}_{u-1}) + \theta(\mathbf{f}_{t,u}) + \mathbf{b})) \quad (11)$$

$$\alpha_{t,u} = \frac{exp(e_{t,u})}{\sum_{t'=1}^{T'} exp(e_{t',u})} \quad (12)$$

where $\mathbf{F}$ in equation 10 is used to exact vectors $\mathbf{f}_{t,u}$ at each position t from previous alignment $\boldsymbol{\alpha}_{u-1}$. The overall attention model is trained to optimize cross-entropy on training data.

3. MODEL UNITS AND TRAINING STRATEGIES

We investigate characters, wordpieces and words as modeling units for end-to-end models. The character units include letters ($a-z$), digits ($0-9$), punctuations ($\&.'\%/-$), special transcribed notations ('[laughter]', '[noise]', '[vocalized-noise]') and an additional space symbol $\langle space \rangle$. The space symbol is used for segmenting recognized character sequences to word sequences. We train a statistical wordpiece model [21] with word counts obtained from training set text data for segmenting each word individually into subwords. The wordpiece models may also output any unit that the character model has.

The label lengths for different output units may vary. As shown in [17], the amount of time-scale subsampling in the encoder impacts both model convergence and performance. In this work, we investigate three types of down-sampling methods: stack consecutive frames [24], pyramidal pooling [17] and convolutional max-pooling.

We find that training an end-to-end model from scratch is difficult and tricky. For fast convergence and better performance, we propose a multi-stage pre-training strategy as shown in figure 1. Namely, initializing encoder and decoder network from CTC trained acoustic model and CE trained language model respectively. Firstly, we train a phone model using CTC criterion. Then, we train a CTC subword model initialized from CTC trained phone model. After that, we initialize encoder network from CTC trained subword models to train attention based models and RNN-T.

In recent years, ADAM [25] is a popular method for optimizing sequence-to-sequence tasks. Although [17] shown a promising results on Switchboard and Fisher tasks, it is still very hard to schedule the learning rate when using SGD. To find a simpler and better scheduling method, we compare the performance between ADAM and SGD.

Data augmentation is also explored here. As shown in [18, 26, 27], speed perturbation contributes a lot to end-to-end systems, and [28] proposed a simple data augmentation method for speech recognition called SpecAugment, which is applied directly to the feature inputs of a neural network by masking blocks of frequency channels and time steps. We will also investigate the data augmentation with our proposed model in this work.

4. EXPERIMENTS

We conduct our experiments on both Switchboard-300hrs and Switchboard+Fisher-2000hrs datasets. 40 dimensional log-mel filterbanks are extracted with a step size of 10ms and window size of 25ms, and then globally normalized so that each input spectrogram bin has zero mean and unit variance. We do not use speaker information in any of our models. Evaluation is carried out on the Switchboard (SWBD) and Callhome (CH) subset of the NIST 2000 CTS test set. The language model used for decoding the CTC phoneme model is the swbd+fisher 3-gram LM. For end-to-end systems, we don't use any language models in the process of decoding.

4.1. Model specification and training details

We use 40-dimensional log-mel features as input. To reduce the GPU memory consumption and accelerate training stage, we reduce the input frame rate in the encoder, which is important for successfully training sequence-to-sequence speech models as shown in [23]. Thus, we first explore different subsampling rates on CTC phone models. Then we choose the best subsampling type to do subword experiments. Specifically, our encoder optionally include two layers of 3×3 convolution with 32 channels, which results in a total time reduction factor of 4. On top of the convolutional

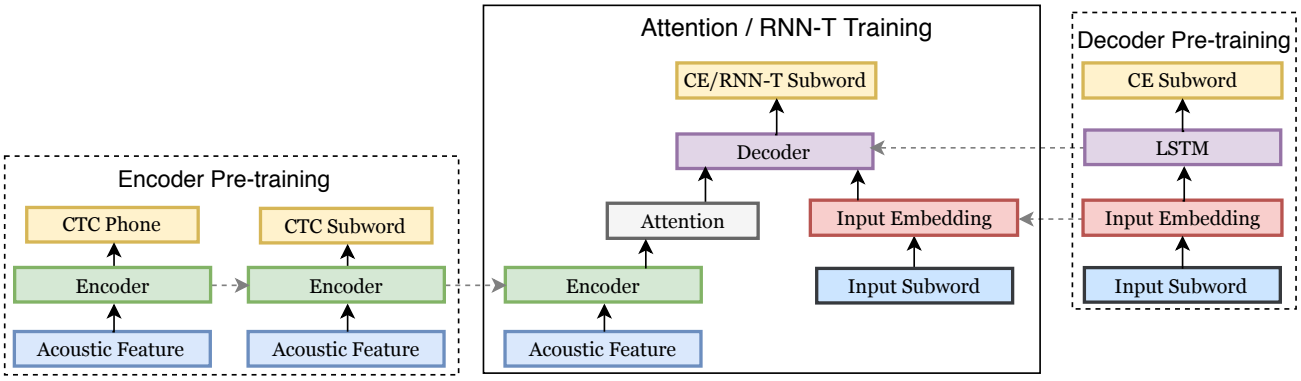


Fig. 1. The multi-stage pre-training of RNN-T and attention-based models. The encoder network is initialized from a CTC model, which is pre-trained from a CTC phone model. The decoder network is trained as a LSTM language model predicting subword units. Then, the RNN-T or attention-based network weights are initialized from the two pre-trained models, indicated by the dashed lines. The attention weights are randomly initialized. For RNN-T model, we don't use attention mechanism. Finally, the entire network is optimized using the RNN-T or cross-entropy loss for RNN-T or attention-based models respectively.

layers, the encoder contains 4 or 5 layers of bi-directional LSTMs [29], with 512 cells in each layer. The decoder consists of 2 LSTM [30] layers with 1024 hidden units, and uses *location-aware attention* as described in [5]. Besides, we compare two different optimizers, SGD and ADAM. We use 1e-3 as initial learning rate with momentum 0.9 and halve it once the loss on validation set has no improvement. For Switchboard dataset, we train the models on single GPU with a batch of 32 utterances. For Switchboard+Fisher, we use synchronous training on 4 GPUs with a batch size of 20.

4.2. Training strategies exploration

For fast exploring, we first compare SGD and ADAM on CTC models. Table 1 shows ADAM outperforms SGD on both phone and wordpiece (WPM) units. Then we do the further experiments with ADAM optimizer.

Table 1. Performance comparison of SGD and ADAM on CTC models. Both phone and WPM use 4 BLSTM layers.

Model	Optimizer	WER (%)	
		SWBD	CH
CTC Phone	SGD	15.3	28.2
	ADAM	**13.0**	**24.5**
CTC WPM	SGD	25.1	38.3
	ADAM	**17.3**	**30.0**

Previous work [17] has shown that time step reduction in the encoder is an effective way to control both the memory usage and the training time. For fast model convergence and better performance, we first investigate three kinds of subsampling methods, then apply the best configuration for further experiments. Table 2 presents results of three subsampling methods on the CTC phone model: stack every 3 frames and decimation [24], pyramidal pooling between BLSTM layers and convolutional front-end with max-pooling. The last two approaches give 4 times subsampling rate in our setups.

Table 2. Results of different subsampling methods on 4BLSTM CTC phone models.

Subsampling method	WER (%)	
	SWBD	CH
frame stacking and decimating	13.0	24.5
pyramid pooling	12.9	24.4
convolutional max-pooling	**12.8**	**23.4**

Table 2 shows the convolutional front-end is better than other two kinds of subsampling types. The model with convolutional front-end converged in 13 epochs, run time for each epoch is about 23 minutes. Model with the first two methods converged within 16 and 18 epochs respectively, and the run time for both of them is about 32 minutes. Considering better performance and convergence, our later experiments use convolutional front-end as default.

As shown in [18, 26, 27], speed perturbation contributes a lot to end-to-end systems. To further improve the model performance, we compare two kinds of strategies when using perturbed training data. Table 3 shows the results on CTC phone model. Line 1 is the baseline model. Line 2 use 0.9, 1.0 and 1.1 perturbed training data. Line 3 only use 1 fold of perturbed data, which means at each epoch, it randomly select one of the perturbed feature for each utterance [26]. Randomly use 1 fold of perturbed data gives 7.4% improvements. Later in our experiments, we use this random perturbation strategy as default.

With the strategies discussed above, we first conduct our

Table 3. Speed perturbation on 5BLSTM CTC phone model.

Speed Perturb.	WER (%)	
	SWBD	CH
✗	12.1	23.4
✓	11.3	22.9
✓(1 fold)	**11.2**	**21.3**

experiments on wordpiece units. We investigate the influence of various initialization methods. In order to fully exploit the potential of encoder-decoder architectures, we propose a multi-stage pre-training strategy. Namely, we start from CTC phone model, then use this phone model to initialize CTC subword model. For faster convergence, we also initialize the decoder network from a pre-trained language model. Table 4 shows that multi-stage pre-training leads to significant improvements over training from scratch. To be more specific, CTC phone model initialization gives 8.0%, 18.7% and 13.5% relative improvements for CTC, attention and RNN-T wordpiece models respectively. Initialized with the CTC wordpiece model gives further 7.9% and 5.4% relative improvements for attention and RNN-T models.

Table 4. Comparison of sequence-to-sequence models with wordpiece units with different initialization models.

Model	Init.	WER (%)	
		SWBD	CH
CTC	Random	15.0	25.5
	Phone	13.8	24.5
Attention	Random	15.5	26.2
	Phone	12.6	23.2
	CTC-WPM	**11.6**	22.9
RNN-T	Random	14.8	25.6
	Phone	12.8	23.1
	CTC-WPM	12.1	**22.4**

4.3. Comparison of model units

We train our sequence-to-sequence models on three modeling units, namely, character, wordpiece (WPM) and word. For RNN-T model, since we use joint network, we cannot feed even a single utterance into GPU memory when the vocabulary size is 10k. Therefore, we skip the results of RNN-T on large vocabulary. Table 5 shows that even with a pre-trained phoneme encoder, the CTC char model still far from any end-to-end systems. That mainly because the conditional independence assumption made by CTC makes it difficult to model long context information. We observe that word unit gives 8.0% relative improvements over wordpiece unit for CTC model. We find RNN-T and attention models perform similar on SWBD test set. With multi-stage pre-training, they both have a huge performance improvement. And attention model with wordpiece unit outperforms all our systems.

Table 5. Comparison of sequence-to-sequence models with various output units on Switchboard dataset. Multi-stage pre-training is used to improve attention and RNN-T models. All results are reported using **greedy search**.

Unit	Model	WER (%)	
		SWBD	CH
Char	CTC	24.1	40.3
	Attention	12.3	28.8
	RNN-T	12.5	23.0
WPM-1k	CTC	13.8	24.5
	Attention	**11.6**	22.9
	RNN-T	12.1	22.4
WPM-10k	CTC	13.1	23.2
	Attention	11.8	22.0
Word-10k	CTC	12.7	22.0
	Attention	11.8	**21.9**

4.4. Optimization improvements

To further exploit the performance of attention model, we apply label smoothing [31, 32] to our best system on wordpiece-1k units. We smooth the ground-truth label distribution with a unigram distribution over all labels. Table 6 shows unigram label smoothing with uncertainty 0.1, and results are obtained by a beam search with beam size 8.

Table 6. Unigram label smoothing (lsm) with SpecAugment.

Model	WER (%)	
	SWBD	CH
Attention	11.6	22.3
+ lsm 0.1	10.8	21.1
+ SpecAug.	**9.2**	**18.2**

We observe that there is still a huge gap for log-likelihood reported on training and validation set. We find these networks tend to overfit to the training data. To improve generalization ability, we apply SpecAugment [1] [28] to our best system reported above.

Then, we conduct experiments on Switchboard+Fisher dataset using the same training strategies as Switchboard dataset, without speed perturbation. We use synchronous training on 4 GPUs with wordpiece-10k units. The WER of our CTC phone model is 9.4%/17.0%. The performance of

[1] We apply a double consecutive mask with frequency parameter 15 and time parameter 30. They achieve 6.8%/14.1% WER on the Switchboard 300h task with 6 BLSTM, which is 4 fold than our model parameters.

the models is presented in table 7 along with other published results.

Table 7. WER comparison against previous published results on Switchboard+Fisher Hub5'00 benchmark without acoustic model adaptation. We only list results using single model here. All the hybrid systems reported WER using N-gram language models. Our results are reported using greedy search.

Architecture	Unit	WER (%) SWBD	CH
Hybrid			
BLSTM + LF MMI [33]	State	8.5	15.3
LACE + LF MMI [34]		8.3	14.9
CTC + Gram-CTC [35]		7.3	14.7
BLSTM + Feat. fusion [36]		7.2	12.7
Dense CNN-BLSTM [2] [37]		6.7	12.5
End-to-End			
Iterated-CTC + LM [38]	Char	10.2	17.7
Attention [17]		8.6	17.8
Attention [18]		8.3	15.5
RNN-T + LM [17]		8.1	17.5
EE-LF-MMI + LM [39]		8.0	17.6
Our Work			
CTC + SpecAugment	WPM	7.9	14.5
Attention		9.2	19.0
+ Multi-stage pre-training		6.8	13.6
+ SpecAugment		**5.9**	**12.1**

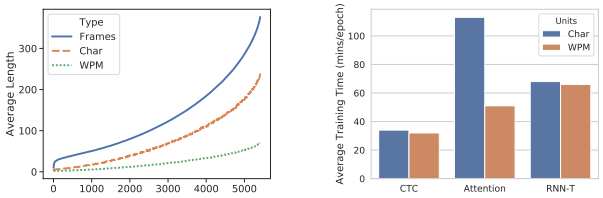
5. ANALYSIS

We observe that a large part of the improvements described in this work are from a reduction in substitution errors as shown in Table 8. Using wordpieces instead of characters results in a relative 5.7% and 20.5% on SWBD and CH test set respectively for our best attention model. While the improvements for RNN-T model is relatively smaller, only about 3% improvements on both SWBD and CH test set. Without an external language model, CTC model with character unit is not comparable with any systems we have.

Table 8. Error analysis on SWBD test set.

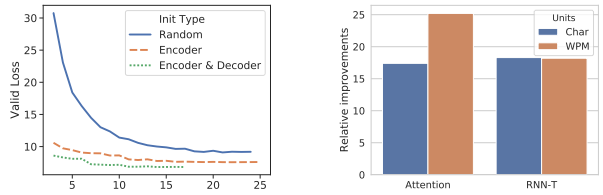
Model	Unit	Sub.	Del.	Ins.	Err.
CTC	Char	17.7	4.6	1.8	24.1
	WPM	**8.6**	3.9	1.4	13.8
Attention	Char	8.0	2.7	1.5	12.3
	WPM	**7.3**	2.9	1.4	11.6
RNN-T	Char	8.1	2.9	1.5	12.5
	WPM	**7.2**	3.7	1.2	12.1

[2]We use their best results without acoustic model adaptation.



(a) Average length / mini-batch. (b) Training time / epoch.

Fig. 2. Average units length in each mini-batch and average training time cost in each epoch.



(a) Learning curves. (b) Relative improvements.

Fig. 3. Comparison of learning curves for various pre-training strategies. The loss on the validation set is reported. The relative improvements of our proposed multi-stage pre-training to randomly initialization is shown in the right.

Figure 2 presents the average utterance length (after subsampling) and the label sequence length of various model units in each mini-batch, along with the training time cost in each epoch. The average character length in each mini-batch is almost two-fold than that of wordpiece. This can dramatically increase the training and inference time with attention models. Figure 3 shows that our proposed multi-stage pre-training can significantly accelerate the training stage and improve the performance of both attention and RNN-T models.

6. CONCLUSION

We present a comprehensive comparison of three popular models on different modeling units for the end-to-end speech recognition task. Considering both performance and training/inference speed, we find that wordpieces outperforms characters and words. We also propose a multi-stage pre-training strategy, with which we can achieve a new state-of-the-art performance on the Switchboard/CallHome test set only with a single speech recognition system.

7. REFERENCES

[1] G.E. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.

[2] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014, pp. 1764–1772.

[3] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, and Zhenyao Zhu, "Deep speech 2: End-to-end speech recognition in english and mandarin," *arXiv preprint arXiv:1512.02595*, 12 2015.

[4] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[5] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: a neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016, pp. 4960–4964.

[7] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

[8] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo, "Direct acoustics-to-word models for english conversational speech recognition," in *Interspeech*, 2017, pp. 959–963.

[9] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013, pp. 6645–6649.

[11] Kanishka Rao, Hasim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *ASRU*, 2017, pp. 193–199.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[13] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP*, 2018, pp. 4774–4778.

[14] Yonghui Wu, Mike Schuster, Zhifeng Chen, and et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," in *ACL*, 2017, p. 339351.

[15] Rohit Prabhavalkar, Kanishka Rao, Tara Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Interspeech*, 2017, pp. 939–943.

[16] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learnin," in *ICASSP*, 2017, p. 48354839.

[17] 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 *ASRU*, 2017, pp. 206–213.

[18] Chao Weng, Jia Cui, Guangsen Wang, Jun Wang, Chengzhu Yu, Dan Su, and Dong Yu, "Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition," in *Interspeech*, 2018, pp. 761–765.

[19] Zhehuai Chen, Mahaveer Jain, Yongqiang Wang, Michael L. Seltzer, and Christian Fuegen, "End-to-end contextual speech recognition using class language models and a token passing decoder," in *ICASSP*, 2018, pp. 6186–6190.

[20] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2016, pp. 1715–1725.

[21] Mike Schuster and Kaisuke Nakajima, "Japanese and korean voice search," in *ICASSP*, 2012, pp. 5149–5152.

[22] Yanzhang He, Tara Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan

Pundak, Khe Chai Sim, Tom Bagby, Shuo-yiin Chang, Kanishka Rao, and Alexander Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*, 2019.

[23] Kazuki Irie, Rohit Prabhavalkar, Anjuli Kannan, Antoine Bruguier, David Rybach, and Patrick Nguyen, "Model unit exploration for sequence-to-sequence speech recognition," *arXiv preprint arXiv:1902.01955*, 2019.

[24] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Interspeech*, 2015.

[25] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[26] Jia Cui, Chao Weng, Guangsen Wang, Jun Wang, Peidong Wang, Chengzhu Yu, Dan Su, and Dong Yu, "Improving attention-based end-to-end asr systems with sequence-based loss functions," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 353–360.

[27] Chengzhu Yu, Chunlei Zhang, Chao Weng, Jia Cui, and Dong Yu, "A multistage training framework for acoustic-to-word model," in *Interspeech*, 2018, pp. 786–790.

[28] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[29] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2673–2681, 1997.

[30] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, November 1997.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jon Shlens, "Rethinking the inception architecture for computer vision," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[32] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Interspeech*, 2017.

[33] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Interspeech*, 2016.

[34] Wayne Xiong, Jasha Droppo, Xinsheng Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, D. H. Yu, and Geoffrey Zweig, "Achieving human parity in conversational speech recognition," *ArXiv*, vol. abs/1610.05256, 2016.

[35] Hairong Liu, Zhenyao Zhu, Xiangang Li, and Sanjeev Satheesh, "Gram-ctc: Automatic unit selection and target decomposition for sequence labelling," in *ICML*, 2017.

[36] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall, "English conversational telephone speech recognition by humans and machines," in *INTERSPEECH*, 2017.

[37] Kyu J. Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian R. Lane, "The capio 2017 conversational speech recognition system," *ArXiv*, vol. abs/1801.00059, 2017.

[38] Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke, "Advances in all-neural speech recognition," 2017, pp. 4805–4809.

[39] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "End-to-end speech recognition using lattice-free mmi," in *Interspeech*, 2018.