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

Faster, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces

Frank Zhang, Yongqiang Wang, Xiaohui Zhang, Chunxi Liu, Yatharth Saraf, Geoffrey Zweig

Facebook AI, USA

{frankz,yqw,xiaohuizhang,chunxiliu,ysaraf,gzweig}@fb.com

Abstract

In this work, we first show that on the widely used LibriSpeech benchmark, our transformer-based context-dependent connectionist temporal classification (CTC) system produces state-of-the-art results. We then show that using wordpieces as modeling units combined with CTC training, we can greatly simplify the engineering pipeline compared to conventional frame-based cross-entropy training by excluding all the GMM bootstrapping, decision tree building and force alignment steps, while still achieving very competitive word-error-rate. Additionally, using wordpieces as modeling units can significantly improve runtime efficiency since we can use larger stride without losing accuracy. We further confirm these findings on two internal *VideoASR* datasets: German, which is similar to English as a fusional language, and Turkish, which is an agglutinative language.

Index Terms: hybrid speech recognition, CTC, acoustic modeling, wordpiece, transformer, recurrent neural networks

1. Introduction

Deep neural networks have been the de facto architecture for automatic speech recognition (ASR) tasks since they were first introduced [1]. These network architectures have evolved in recent years and can be broadly classified into two categories: 1) Those that support streaming during inference, such as time delay neural network (TDNN) [2], feed-forward sequential memory networks (FSMN) [3], long short-term memory (LSTM) [4], latency-controlled bi-directional LSTM (LC-BLSTM) [5,6] and time-depth separable convolutions (TDS) [7] etc. and, 2) Full sequence architectures when latency is not a concern, e.g. BLSTM [8], which can be used to provide better accuracy since the neural network can take full advantage of future information. Recently, Transformer [9] architectures have shown superior results in ASR tasks compared to BLSTMs [10–13]. In this work, we use LC-BLSTM as a representative for streaming and Transformer for the full sequence use case.

Traditional hybrid DNN/Hidden Markov Model (HMM) approach utilizes a neural network to produce a posterior distribution over tied HMM states [14, 15] for each acoustic frame, usually followed by sequence discriminative training to boost performance [16]. CTC [17] has became an alternative criterion to frame-level cross-entropy (CE) training or sequence-level lattice-free MMI (LF-MMI) training in recent years and has shown promising results [18–22]. Inspired by the rise of end-to-end training in machine translation, encoder-decoder architecture was also introduced for ASR, e.g. Listen, Attend and Spell (LAS) [23]. Most recently, neural transducers [24] have shown great potential for both on-device [25] and server [26] use cases. Here, we consider both CE and CTC trained systems as hybrid since the neural network is solely modeling posteriors distribution over modeling units and a WFST-based decoder was used to produce hypothesis. While the LAS and Transducer based systems are considered end-to-end since there are decoder neural network components that directly produce hypotheses. In this work, our focus will be on CTC-based systems.

The most extensively studied modeling unit of hybrid ASR systems is tied context-dependent (CD) states/phones, i.e. senone [27]. As an alternative, chenone [6] was proposed that has not only shown improvement in accuracy, but also eliminates the need of a phonemic lexicon. Since CTC training does not require alignment labels per frame, graphemes [18], wordpieces (WP) [13, 28–31] or even whole words [32] can be directly modeled.

In the rest of this paper, we first compare performance between chenone and wordpiece as modeling unit. We further study the best striding scheme for both modeling units and the impact of wordpiece vocab sizes. Then we perform neural language model rescoring on top of our best system to produce final WERs on LibriSpeech. With a transformer network trained with chenone-CTC, we achieve state-of-the-art result on LibriSpeech among hybrid systems. On our internal *VideoASR* tasks, a system trained with wordpiece-CTC only slightly lags behind in terms of WER, but is up-to 3x faster during inference compared to systems using chenone due to the larger stride in the neural network.

2. Hybrid Architecture

In hybrid ASR, an *acoustic encoder* is used to encode a sequence of acoustic frames $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T$ to a corresponding sequence of high level embedding vectors $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_T$. A softmax layer is then applied on these embedding vectors to produce a posterior distribution over the chosen modeling unit, e.g. senone or chenone, for each frame. These posterior distributions are then fed into a weighted finite-state transducer (WFST) [33] based decoder with a decoding graph also composed with a lexicon and language model (LM) to find the best hypothesis.

In contrast to CE training that requires pre-computed per frame labels usually through a force alignment process, CTC training can implicitly learn the alignment between the input sequence and target sequence by introducing an additional blank label. The blank label is used to estimate the probability of outputting no label at a given frame. The encoder will be trained to produce probability distribution over all labels including blank. The log-likelihood of a given target sequence $\boldsymbol{y}$ can then be found by summing the probabilities of all allowed alignments. Specifically,

$$\log p(\boldsymbol{y}|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T) = \sum_{\pi \in B^{-1}(\boldsymbol{y})} \prod_{t=1}^{t=T} p(\pi_t|\boldsymbol{x_t}) \quad (1)$$

where B is the mapping operation that removes all blank and

The authors would like to thank Duc Le for helpful discussion about chenone, Jun Liu about CTC decoding and Abdelrahman Mohamed about transformer models

http://dx.doi.org/10.21437/Interspeech.2020-1995

repeating labels in a given sequence. Note that the underlying assumption is probabilities between timestamps are conditional independent, which is ensured since the encoder is non-autoregressive. The network is then trained to maximize the log-likelihood for each training example and p can be computed efficiently using forward-backward algorithm.

3. Acoustic Model Architecture

In this section, we briefly review the neural network architectures we are going to study in this work: Transformer (in Section 3.1) and LC-BLSTM (in Section 3.2).

3.1. Architecture of Transformer

Unlike when the Transformer [9] was first proposed as an encoder-decoder architecture for machine translation task, we only use the encoder part for acoustic modeling. Specifically we follow the setup in [10] and use VGG layers [34] in lieu of the original sinusoid positional encoding, since we have seen that for ASR tasks, convolutional positional encoding performs the best. Iterated loss [35] was also applied to the intermediate embedding of transformer layers to help convergence and improve accuracy. Also different from the original transformer, we apply layer normalization [36] before multi-head attention (MHA) and feed-forward network (FFN) and we have an extra layer normalization operator after the residual connection. This is necessary to prevent bypassing the transformer layer entirely and helps with model convergence.

3.2. Architecture of LC-BLSTM

Unidirectional recurrent neural networks such as LSTMs base their predictions solely on the history they have already seen, hence the prediction accuracy is worse than bi-directional LSTMs that have access to the full context of the input, including future frames. However for streaming applications such as live captioning, we cannot wait for the full context to arrive because the ASR output needs to be made available within a certain latency budget of the audio stream being fed into the ASR system.

To strike a balance between recognition latency and accuracy, LC-BLSTM was first introduced in [5]. Unlike BLSTM that cannot produce any hypothesis until the whole audio input is processed, LC-BLSTM only utilizes a limited number of right context (RC) frames to make predictions, which controls the latency. Similar to BLSTM, each LC-BLSTM layer also has two LSTMs, one left-to-right LSTM and one right-to-left LSTM. The difference is that the the input sequence is first divided into overlapping chunks of chunk size (CS) frames. The amount of overlap between chunks is equal to RC frames. When forwarding left-to-right LSTM, hidden states and cell states are carried over between chunks, so that we have unlimited left context as in BLSTM. We forward right-to-left LSTM on each chunk and only keep $CS - RC$ frames of output activations, so that each input frame has at least RC frames of right context to produce its activation. This mechanism enables LC-BLSTM to generate better acoustic embeddings than LSTM, without delaying the generation until the encoder has seen the whole audio.

4. Modeling Units

4.1. Chenone

Chenone was first introduced in [6] as an alternative to traditional senone [27] unit. Chenone not only eliminated the need for a phonetic lexicon, usually generated by linguists, but also showed WER improvement relative to phonetic modeling units in some cases. In order to use chenone as modeling unit in CTC training, we shift the labels after force alignment by one (all labels +1), and use label $\#0$ as the blank label in CTC. Essentially the neural network is now modeling distribution over all chenones plus one blank symbol. We then squeeze same adjacent labels in the alignment sequence into one label, let the encoder trained with CTC criterion to learn the implicit alignments.

After a CTC model is trained, the next step is to construct a decoding graph. We first build an $\mathrm{H} \circ \mathrm{C} \circ \mathrm{L} \circ \mathrm{G}$[1] graph following the standard procedure for CD-HMM (here, we always assume every HMM only has 1 state); then we transform it to a new graph which can consume an extra blank symbol. For each FST state s in the decoding graph (shown in Figure 1), we split that state into two FST states, s and s'; we moved the outgoing edges ($e2$ and $e4$) to start from s'; A self loop edge with blank label as input and ϵ as output symbol with weight one (in the semi-ring's sense) is added; s and s' are then connected by a $\epsilon : \epsilon$ edge. Self-loop edges ($e3$ in this example) and incoming edge ($e1$) in this case are not changed. The transition model is on-the-fly converted to a new mapping function that shifts the output units by one. Once we change the transition model and decoding graph, the standard Kaldi decoder can be used to decode chenone-CTC model.

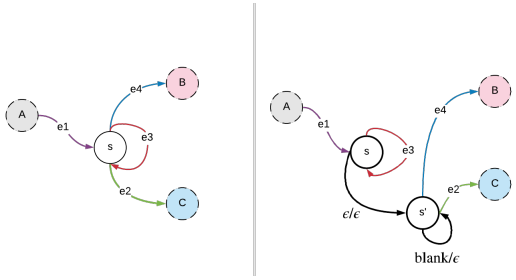

Figure 1: *Convert a standard CD-HMM FST to be CTC compatible. A standard FST was shown on the left and a CTC compatible FST after conversion on the right*

4.2. Wordpiece

Subword unit representations such as byte pair encoding (BPE) [37] and wordpiece model [38] have been proposed with improved performance in many natural language processing (NLP) tasks. This approach chooses to divide words into a limited set of subword units, e.g. the word "hello" may be encoded as "_he ll o", where the underscore indicates word start. In this work, we train wordpiece model using unigram language model word segmentation algorithm [39], and use the generated wordpiece vocab plus blank symbol as modeling units.

Since wordpiece is context independent, we only need to build an $\mathrm{H} \circ \mathrm{L} \circ \mathrm{G}$ graph for decoding, where: H transduces $n+1$ symbols to n wordpieces, i.e. absorbing the blank symbol; L maps the sequence of wordpieces to the sequences of words,

[1]Note that L here is simply a mapping between word and its graphemes. Examples could be found in [6]

e.g. "_he ll o" to "hello", and is done by the trained wordpiece model; G is the standard n-gram word level LM. Once these FSTs are constructed, a standard procedure is used to compose H, L and G together.

Compared to alignment-based training which first need to train a bootstrap model and build decision tree; then use the bootstrap model to perform force alignment and finally CE training, we only need to train a wordpiece model using text-only data and followed by one-stage CTC training. The whole training pipeline is greatly simplified.

5. Experiments

To evaluate the performance of different modeling units and training criterion, we first conduct experiments on the LibriSpeech corpus [40], followed by experiments on two larger and more challenging internal datasets of German and Turkish social media videos.

5.1. Data

The LibriSpeech corpus contains about 960 hours of read speech data for training, and 4 development and test sets ({*dev, test*}-{*clean,other*}), where *other* sets are more acoustic challenging. We use the official 4-gram language model (LM) with 200K vocabulary for all first-pass decoding and n-best generation for neural LM rescoring.

The German and Turkish datasets are our in-house video datasets, which are sampled from public social media videos. The datasets are completely de-identified before transcription; both transcribers and researchers do not have access to any user-identifiable information (UII). The training, validation and test set sizes are shown in Table 1. All hyper-parameter tuning is done on validation set. The video datasets contain a diverse array of speakers, accents, video categories, and acoustic conditions, and are more challenging than the LibriSpeech dataset explored in this work.

We use the same bootstrap model obtained from decision tree building stage to segment the training split of both LibriSpeech and video dataset to up to 10 seconds [2]. We also segment the dev and test split of video dataset to up to 10s, while no segmentation is performed on LibriSpeech dev and test sets in order to be comparable with other published results. The benefit of segmenting training data was shown in [10].

Table 1: *Dataset sizes for internal Video ASR tasks. Number of utterances in parentheses.*

Language	German	Turkish
train	3K hrs (~135K)	3.1K hrs (~137K)
valid	14.5 hrs (~600)	14.4 hrs (~600)
test	24.2 hrs (~1K)	24.4 hrs (~1K)

5.2. Experiment Setup

We follow [6, 10] to use context- and position-dependent graphemes (i.e., *chenones*) for CE baselines and CTC experiments. We bootstrap our HMM-GMM system using the standard Kaldi [41] LibriSpeech recipe. We use 1-state HMM topology with fixed self-loop and forward transition probability (both 0.5). 80-dimensional log Mel-filter bank features are extracted with a 10ms frame shift and 25ms FFT windows. Speed perturbation and *SpecAugment* [42] (LD policy without time warping) are applied to all experiments unless specially noted.

Since the focus of this work is not on network architecture searches, we use our previously found best setup across all following experiments. For acoustic transformers, we use a 24-layer transformer encoder architecture with embedding dimension 512 and 8 attention heads; the FFN dimension is 2048. Iterative loss was applied to intermediate output embeddings of layer 6, 12 and 18. We use three VGG blocks [43] to encode acoustic features before feeding into transformer layers: each VGG block contains 2 consecutive convolution layers with a 3-by-3 kernel followed by a ReLu non-linearity and a pooling layer; 64 channels are used in the convolution layer of the first VGG block and increase to 128 for the second block and 256 for the third block. Max-pooling is performed at a 2-by-2 grid, with optional stride choice from 1 to 3 in each block. This model has about 81M parameters and we note the model as *vggTrf*. For LC-BLSTMs, we follow [6] and use 5 layers with 800 hidden units per layer per direction. Optional subsampling by a factor of 2 or 3 can be applied after the first hidden layer. This model has about 83M parameters, similar to the transformer model. Dropout is applied in all experiments: 0.1 for transformers and 0.2 for LC-BLSTM.

All neural network training is performed using an in-house developed *PySpeech* framework that is built on top of the open-sourced PyTorch-based *fairseq* [44] toolkit. Adam optimizer [45] with (0.9, 0.999) betas and $1e^{-8}$ epsilon is used in all experiments. We apply the *tri-stage* [42] learning rate (LR) scheduler. The hold stage LR is $1e^{-3}$. For experiments on transformer models, we follow a schedule of (48K, 100K, 200K) steps with a batch contains up to 20,000 frames. For LC-BLSTM models we use schedule of (16K, 32K, 64K) steps with a batch contains up to 50,000 frames. All models are trained on 32 Nvidia V100 GPUs for 200 epochs in total. Training is usually finished between 2 to 4 days.

Test set WERs are obtained using the best model based on evaluated WER on the development set. The best checkpoints for both LibriSpeech *test-clean* and *test-other* are selected using the *dev-other* development set. For video, the best checkpoint was selected using the valid set.

5.3. Effect of Stride

In the first set of experiments, we investigate what is the best stride for different modeling units. For *vggTrf* model, striding is achieved by setting stride of max-pooling layers, e.g. a total stride of 8 can be achieved by setting stride equals 2 for all three VGG blocks. For LC-BLSTM, we apply subsampling factor of 2 for activation of the three consecutive hidden layers right after the first layer. We follow the setup in [6] and use stride 2 for the chenone-CE baseline. We use 2K wordpiece size in this study. Results on LibriSpeech *test-other* dataset shown in Table 2.

The results show that for chenone-CTC, the best stride is 4 and it out performs CE baseline. While for WP-CTC system, we find that we can use stride as high as 8 without losing much accuracy. Because wordpiece model is trained based on word frequency, some common whole words, e.g. 'welcome' and 'information', made into the vocab. The acoustic model can use larger stride since the corresponding input frames will span longer. On the other hand, although context dependent, chenones are essentially still characters and have a shorter corresponding time span in input sequence. So we observed that WER degrades quickly when larger stride is used.

[2] This is achieved by force aligning the whole audio against the reference using the LC-BLSTM acoustic model.

Table 2: *Effect of stride for wordpiece and chenone*

Unit	Criterion	Stride	*LC-BLSTM*	*vggTrf*
chenone	CE	2	8.81	5.59
WP	CTC	2	9.75	5.96
		2*2	**8.92**	**5.74**
		3*2	9.16	5.84
		2*2*2	8.94	5.90
		3*2*2	9.34	6.51
chenone	CTC	2	9.43	5.28
		2*2	**8.51**	**5.16**
		3*2	9.93	6.35
		2*2*2	17.94	17.27

5.4. Effect of Wordpiece Dictionary Size

In the second set of experiments, we explore the effect of different wordpiece sizes. Here, all models use a total stride of 8. We report results on LibriSpeech *test-other* dataset in Table 3.

Table 3: *WER using different wordpiece sizes*

Wordpiece size	*LC-BLSTM*	*vggTrf*
1K	**8.71**	**5.86**
2K	8.94	5.90
5K	9.13	6.05
10K	9.23	6.35
16K	9.44	6.51

The results show that on LibriSpeech, smaller wordpiece vocab size tends to work better. It's worth exploring vocab sizes below 1K in future study. We also explored subword regularization during training following [39]. We tried the setup of $(l = 64, \alpha = 0.1/0.5)$ but results turn out slightly worse. During decoding we also tried building decoding graph with multiple pronunciations in the lexicon with corresponding pronunciation probability for the lexicon entries. The results turn out on-par. We also explored sMBR and mWER training after CTC stage. It provides very marginal gain so we didn't include the results in this study.

To achieve best WERs, we trained a 36-layer stride-4 *vggTrf* model with chenone-CTC(about 124M parameters), and changed time masking of *SpecAugment* LD policy to $(T = 30, mT = 10)$. Its performance and those of some other published LibriSpeech systems can be found in Table 4. The new system outperforms our previous best hybrid system [10] by 11% and 14% respectively on *test-clean* and *test-other*. We also trained a 42-layer transformer LM following the setup in [46] with the LibriSpeech transcriptions and 800M-word text-only data. The transformer LM achieved perplexity 52.35 on the dev set (a combination of *dev-clean* and *dev-other*). We then perform n-best rescoring on up to 100-best hypotheses generated by first pass decoding. The oracle error rate of the n-best hypotheses are 1.0% and 2.2% on *test-clean* and *test-other* respectively. Our final WERs (2.10%/4.20%) are the best results among all hybrid systems on this widely used benchmark.

5.5. Experiments on Video dataset

Finally, we tested *vggTrf* with CTC criterion on the more challenging and larger scale internal *VideoASR* tasks, as described in Section 5.1. Stride 2 is used for CE, 4 for chenone-CTC and 8 for WP-CTC training. Results are shown in Table 5. We find that both CTC systems outperform the baseline CE system.

Table 4: *Comparison of our chenone-CTC with previous best results on LibriSpeech. "4g" means the official 4-gram LM was used; "NNLM" means a neural LM was used.*

Arch.	System	LM	test-clean	test-other
LAS	Karita et al. [11]	NNLM	2.6	5.7
	Park et al. [47]	NNLM	2.2	5.2
	Synnaeve et al. [13]	NNLM	2.33	5.17
Transducer	Zhang et al. [26]	No LM	2.4	5.6
		NNLM	**2.0**	4.6
Hybrid	RWTH [8]	4g	3.8	8.8
		+NNLM	2.3	5.0
	Han et al. [48]	4g	2.9	8.3
		+NNLM	2.2	5.8
	Wang et al. [10]	4g	2.60	5.59
		+NNLM	2.26	4.85
	Ours	4g	**2.31**	**4.79**
		+NNLM	2.10	**4.20**

It's also consistent that chenone-CTC outperforms WP-CTC in terms of WER. On the other hand, WP-CTC system is significantly better in terms of real-time factor (RTF) due to the larger stride used. Blank frame skipping during decoding also contributes to the speedup. We skip frames if blank label posterior is greater than 99%. Empirically, we found that over 20% of the frames in chenone-CTC and over 50% of frames in WP-CTC were skipped.

Table 5: *Experiment results on internal VideoASR tasks*

Unit	Criterion	German		Turkish	
		WER	RTF	WER	RTF
chenone	CE	15.54	0.26	21.92	0.25
WP	CTC	14.32	**0.07**	19.04	**0.06**
chenone		**13.74**	0.10	**18.17**	0.10

6. Discussions and Conclusions

In this work, we pushed the performance boundary of hybrid ASR using transformer-based acoustic models with context-dependent CTC training. Modeling choices are discussed and compared in detail, and as Table 4 shows, our system yields state-of-the-art results on the LibriSpeech benchmark.

For real world production system however, we must take into account not only recognition accuracy, but also engineering complexity, inference efficiency, flexibility etc. We have shown that hybrid systems can be built with fewer steps leveraging wordpiece and CTC training. Yet, the system's accuracy is very competitive with a much faster runtime inference speed. Results on a more challenging internal dataset show similar results, confirming that such a wordpiece-CTC system indeed has great potential.

7. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal processing magazine*, vol. 29, 2012.

[2] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015.

[3] S. Zhang, H. Jiang, S. Wei, and L. Dai, "Feedforward sequential memory neural networks without recurrent feedback," *arXiv:1510.02693*, 2015.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014.

[5] Y. Zhang, G. Chen, D. Yu *et al.*, "Highway long short-term memory RNNs for distant speech recognition," in *Proc. ICASSP*, 2016.

[6] D. Le, X. Zhang, W. Zheng, C. Fügen *et al.*, "From Senones to Chenones: Tied Context-Dependent Graphemes for Hybrid Speech Recognition," *Proc. ASRU*, 2019.

[7] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions," *Proc. Interspeech*, 2019.

[8] C. Lüscher, E. Beck, K. Irie *et al.*, "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention-w/o Data Augmentation," *Proc. Interspeech*, 2019.

[9] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *Proc. NueralIPS*, 2017.

[10] Y. Wang, A. Mohamed, D. Le, C. Liu *et al.*, "Transformer-based Acoustic Modeling for Hybrid Speech Recognition," in *Proc. ICASSP*, 2019.

[11] S. Karita, N. Chen, T. Hayashi *et al.*, "A Comparative Study on Transformer vs RNN in Speech Applications," in *Proc. ASRU*, 2019.

[12] M. Sperber, J. Niehues, G. Neubig *et al.*, "Self-Attentional Acoustic Models," in *Proc. Interspeech*, 2018.

[13] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko *et al.*, "End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures," *arXiv:1911.08460*, 2019.

[14] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large Vocabulary Continuous Speech Recognition With Context-Dependent DBN-HMMS," in *Proc. ICASSP*, 2011.

[15] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic Modeling using Deep Belief Networks," *IEEE TASLP*, 2011.

[16] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani *et al.*, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016.

[17] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. ICML*, 2006.

[18] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai *et al.*, "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," in *Proc. ICML*, 2016.

[19] Andrew, H. Sak, F. de Chaumont Quitry, T. Sainath *et al.*, "Acoustic modelling with CD-CTC-SMBR LSTM RNNS," in *Proc. ASRU*, 2015.

[20] H. Sak, A. Senior, K. Rao, O. İrsoy *et al.*, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Proc. ICASSP*, 2015.

[21] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *Proc. Interspeech*, 2015.

[22] G. Pundak and T. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

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

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

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

[26] Q. Zhang, H. Lu, H. Sak, A. Tripathi *et al.*, "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss," in *Proc. ICASSP*, 2020.

[27] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE TASLP*, 2011.

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

[29] "hybrid ctc-attention based end-to-end speech recognition using subword units."

[30] 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.

[31] A. Das, J. Li, G. Ye, R. Zhao *et al.*, "Advancing Acoustic-to-Word CTC Model with Attention and Mixed-Units," *IEEE TASLP*, 2018.

[32] H. Soltau, H. Liao, and H. Sak, "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition," in *Proc. Interspeech*, 2017.

[33] M. Mohri, F. Pereira, and M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech & Language*, 2002.

[34] O. Abdel-Hamid, A. Mohamed, H. Jiang *et al.*, "Convolutional neural networks for speech recognition," *IEEE TASLP*, 2014.

[35] A. Tjandra, C. Liu, F. Zhang *et al.*, "Deja-vu: Double Feature Presentation and Iterated loss in Deep Transformer Networks," in *Proc. ICASSP*, 2020.

[36] J. Lei Ba, J. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[37] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proc. ACL*, 2016.

[38] Y. Wu, M. Schuster, Z. Chen, Q. V. Le *et al.*, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *arXiv:1609.08144*, 2016.

[39] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *Proc. ACL*, 2018.

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

[41] D. Povey, A. Ghoshal, G. Boulianne *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. ASRU*, 2011.

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

[43] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. ICLR*, 2015.

[44] O. Myle, E. Sergey, B. Alexei, F. Angela *et al.*, "fairseq: A Fast, Extensible Toolkit for Sequence Modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[46] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Proc. Interspeech*, 2019.

[47] D. S. Park, Y. Zhang, C.-C. Chiu, Y. Chen *et al.*, "SpecAugment on Large Scale Datasets," in *Proc. ICASSP*, 2020.

[48] K. J. Han, R. Prieto, K. Wu, and T. Ma, "State-of-the-Art Speech Recognition Using Multi-Stream Self-Attention With Dilated 1D Convolutions," in *Proc. ASRU*, 2019.