FAST AND SLOW ACOUSTIC MODEL

Kshitiz Kumar, Emilian Stoimenov, Hosam Khalil, Jian Wu

Microsoft Corporation, Redmond, WA

{kshitiz.kumar, emilians, hosamk, jianwu}@microsoft.com

ABSTRACT

In this work we layout a Fast & Slow (F&S) acoustic model (AM) in an encoder-decoder architecture for streaming automatic speech recognition (ASR). The Slow model represents our baseline ASR model; it's significantly larger than Fast model and provides stronger accuracy. The Fast model is generally developed for related speech applications. It has weaker ASR accuracy but is faster to evaluate and consequently leads to better user-perceived latency. We propose a joint F&S model that encodes output state information from Fast model, feeds that to Slow model to improve overall model accuracy from F&S AM. We demonstrate scenarios where individual Fast and Slow models are already available to build the joint F&S model. We apply our work on a large vocabulary ASR task. Compared to Slow AM, our Fast AM is 3-4x smaller and 11.5% relatively weaker in ASR accuracy. The proposed F&S AM achieves 4.7% relative gain over the Slow AM. We also report a progression of techniques and improve the relative gain to 8.1% by encoding additional Fast AM outputs. Our proposed framework has generic attributes - we demonstrate a specific extension by encoding two Slow models to achieve 12.2% relative gain.

Index Terms— ASR, LSTM, LC-BLSTM, Encoder-Decoder, Model Combination

1. INTRODUCTION

Deep learning techniques have demonstrated record low ASR word error rate (WER). Although we have reached human-parity in some evaluations, we are far from parity in other tasks, and there is a growing demand to improve ASR across speech applications. The practical constraints like user-perceived latency, accuracy, and computational cost guide our work towards novel ASR solutions. The deep long-short term memory (LSTM) models in [1, 2, 3], and CNN-based CLDNN in [4] have demonstrated improvements over DNN models in [5, 6, 7, 8]. We have also noted a progression of acoustic models in GRU [9], LSTM [10], BLSTM [11, 12], LC-BLSTM [13, 14], and RC-LSTM [15] with associated trade-offs in aforementioned constraints.

The advances in adoption of ASR technologies have led to numerous related applications. Some examples include keyword search (KWS), keyword verification (KWV), speaker ID (SID), language ID (LID), sound event detection, noise embeddings or ASR feature embedding networks for natural language processing tasks. There we typically build a dedicated model that executes concurrently with ASR model. Furthermore, ASR and above application-specific models may share similar technologies, *e.g.*, both models may be built on Uni- or Bidirectional-LSTM networks. However ASR models are generally much larger in order to handle the complexity in ASR tasks. An application-specific model can be a KWV model that detects and validates keywords of interest in streaming audio.

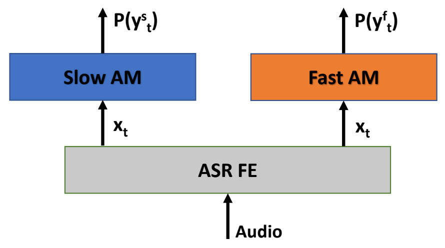

Fig. 1. *Fast AM and Slow AM for hybrid DNN-HMM ASR system. Slow AM constitutes the core ASR model. Fast AM applies to speech applications like keyword search or faster initial ASR responses.*

The KWV model is fundamentally similar to ASR AM but it's 3-4x smaller. Another example can be a dedicated model for faster intermediate ASR hypotheses for minimal user-perceived latency in streaming scenarios.

In above applications, we refer to the core ASR model as "Slow" AM and the application-specific model as "Fast" AM. We also represent individual Fast and Slow AMs in Fig. 1, where they serve respective applications. The models may also share the feature extraction (FE) module that produces features x_t at time t. ASR decoding will consume the posteriors $P(y_t^s)$, and the secondary application will consume $P(y_t^f)$. A large scale ASR system naturally develops an ecosystem of scenarios where Slow and Fast AMs are evaluated concurrently. We focus this work in developing a new direction. We encode the available information from a Fast AM to benefit the Slow AM in an F&S modeling framework. Furthermore, to the best our knowledge, we are developing a new scope and expect it to enable many novel and impactful techniques.

The rest of this work is organized in following. We present background and motivation for this work in Sec. 2. We develop our work on F&S AM in Sec. 3. We present our experiments and results in Sec. 4. We discuss insightful connections and impact of our work to related areas in Sec. 5. Finally Sec. 6 concludes this study.

2. BACKGROUND AND MOTIVATION

In this section we build on our introduction and present a background for developing F&S AM. We discuss Fast AM attributes and applications in Sec. 2.1, and follow up with a discussion on Slow AM in Sec. 2.2, and finally motivate a new scope in F&S AM.

2.1. Fast AM Attributes

There are many speech applications where a faster, although less accurate, model is desirable. Some examples include a device acoustic model that's gated by size, computation and battery requirements.

Another example is a dedicated keyword search model that's evaluated in parallel to ASR model, and needs to be fast enough. These models also serve scenarios with the strongest user-perceived latency requirements, *e. g.*, ASR-based captioning system for video streaming scenarios, where ASR response should have minimal lag between the streamed content and corresponding ASR generated captions. Live speech translation services are other examples where we can trade-off some ASR accuracy for faster system response. These applications motivate a Fast ASR AM for minimal user-perceived latency or fast ASR response in streaming ASR applications. Besides ASR, a Fast AM is also applicable to applications like LID, SID, or feature embedding networks. Typically Fast and standard ASR AMs share fundamental techniques but Fast AM is smaller and simpler.

2.2. Slow AM Attributes

We mentioned that Fast AM is important in streaming scenarios like live speech captioning and other speech applications. However, there are also applications like dictation, offline call center or voicemail transcriptions, and other applications that can work with a bounded and reasonable latency. There are also applications with demanding ASR accuracy, where we necessarily need a large scale ASR model. A Slow AM has deeper and wider network architecture to efficiently learn and generalize to larger scale unified ASR tasks. Often Fast AM is scenario-specific to deliver promising results for the corresponding scenario, whereas, Slow AM is applied to a large range of acoustic factors. Consequently the model needs to be big enough to meet desirable accuracy and robustness requirements. Note that Slow AM can also be structurally different than Fast AM.

2.3. Motivation for Fast and Slow AM

We described applications for individual Fast and Slow models, where we concurrently evaluate Fast and Slow models. That provides a rich ground to design new model architectures to combine respective information from Fast and Slow models. The design can benefit the final ASR accuracy at minimal additional computation or latency costs. We also noted that Fast AM is 3-4x smaller than Slow AM; therefore under similar computation resources, the Fast AM outputs are ahead of the Slow AM by 3-4 states. We also know that future information has significant impact for ASR, it has been demonstrated in novel AM architectures like Bidirectional LSTMs and other models in [16], [17]. Our proposed F&S AM architecture encodes Fast AM outputs. The design also ingests 3-4 future output states from the Fast AM before joining it with the Slow AM path. Above doesn't incur hard latency as Fast AM outputs are relatively ahead. Next we develop and discuss the core F&S AM technology.

3. FAST & SLOW (F&S) AM ARCHITECTURE

Following the motivation in Sec. 2, we describe the F&S AM for streaming ASR in this section. We illustrate the F&S AM in Fig. 2. At a higher level, the F&S AM encodes information from Fast AM and Slow AM for a joint encoder layer. It consists of following key components:

(A) Slow Encoder - The Slow encoder is equivalent to the corresponding encoder in Slow AM. For a typical AM with 6 hidden layers, the Slow encoder can consist of the initial 4 layers. As noted in Fig. 2, the ASR features x_t constitute inputs to the slow encoder.

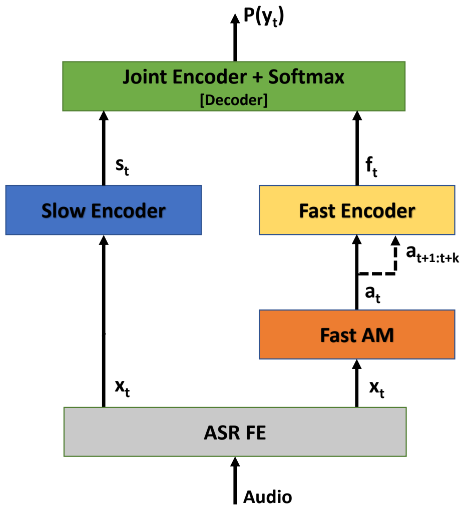

Fig. 2. *Streaming F&S AM in an encoder-decoder architecture. There, Fast and Joint Encoders constitute key innovations in comparison to Fig. 1. The Fast AM outputs are ahead of Slow AM. Therefore outputs from a_t to a_{t+k} can be encoded in Fast Encoder.*

We represent that in (1):

$$s_t = \mathcal{E}^s(x_t) \quad (1)$$
$$a_t = \mathcal{F}(x_t) \quad (2)$$
$$f_t = \mathcal{E}^f(a_t, \cdots, a_{t+k}) \quad (3)$$
$$y_t = \mathcal{D}(s_t, f_t) \quad (4)$$

The Slow encoder produces encoded sequence s_t at time t, and $\mathcal{E}^s$ represents a generic Slow encoder. $\mathcal{E}^s$ can be UniLSTM, Bidirectional LSTM or other desirable networks. The Slow encoder is structurally rich to handle the complexity of ASR tasks.

(B) Fast AM - We have motivated Fast AM in Sec. 2.1. In comparison to Slow AM, Fast AM is structurally simpler, and typically 3-4x smaller. (2) represents a Fast AM, it produces a streaming sequence of outputs a_t. Note that a_t represents a generic output from Fast AM, it can specifically represent Softmax, Logits or hidden layer states.

(C) Fast Encoder - The Fast encoder takes the output of Fast AM, a_t, and produces corresponding encoded sequence in f_t. $\mathcal{E}^f$ represents a Fast encoder. The Slow encoder and Fast AM outputs can be in different vector spaces, so the Fast encoder is tasked to map or align those outputs before their combination. We noted that Fast AM outputs are ahead of the Slow AM by 3-4 states. We leverage that observation by jointly encoding the current Fast AM output a_t as well as k future states in $a_{t+1} \cdots a_{t+k}$. These Fast AM output states bring key innovation to the F&S AM.

(D) Joint Encoder and Softmax - The Joint encoder combines information from the Slow and Fast encoders. The combination can be explicitly done by an attention model but here we follow concatenation operation for simplicity and minimal new parameters. The Joint network encodes the combined information for classification at Softmax layer. This stage consists of 3 key operations in information combination, joint encoding and classification. We also refer to this network as decoder network that produces acoustic state posteriors $P(y_t)$ for ASR decoding. Also note that $y^s_t = \mathcal{D}(s_t)$, *i.e.* in the

Table 1. *Training and Test speech data in hours.*

Training	Test ("All")
7200	200

absence of Fast AM and Fast encoder, this network in combination with Slow encoder defaults to Slow AM.

3.1. Potential Fast AM Outputs for Subsequent Encoding

We consider potential Fast AM output choices, a_t, that can be used for subsequent encoding in Fast encoder.

(A) Softmax or Logits outputs - The Softmax output represents Fast AM posteriors for hybrid DNN-HMM model. Typically that's a 9000-dim vector in our systems, and carries rich state information. However, a practical application also requires a dedicated embedding network to embed the $9k$ outputs to a smaller dimension for effective learning. One advantage of working with Softmax layer is that it can make the Fast AM an independent component in the F&S AM, *i.e.* the Fast AM can be replaced with a newer version of Fast AM without requiring to retrain rest of the network. We can also use Logits or pre-Softmax outputs, though that deeply ties the Fast AM to F&S AM and also requires the aforesaid additional parameters in an embedding network for dimensionality reduction.

(B) Hidden Layers - Hidden layers provide an alternative approach to embed Fast AM information. The outputs are typically 512 for Slow AM and even smaller 256 for Fast AM. This doesn't require the dimensionality reduction network needed for Softmax layer, and leads to an effective training. That's a big saving in computation as well as memory resources. Similar to Logits, working with hidden layers ties Fast AM to F&S AM. Designing an F&S AM with replaceable Fast AM is desirable but not the objective of this work.

3.2. Streaming Implementation of F&S AM

Our work focuses on streaming ASR applications with strong user-perceived latency requirements. The baseline Slow AM provides the best ASR accuracy under those requirements. Therefore, it's crucial for F&S AM implementation to not exceed above latency targets. We have noted that Fast and Slow AMs are evaluated concurrently in respective computation threads. Thus Fast encoder is the only key additional work in F&S AM, and can be done in the computational thread of Fast AM. Furthermore, by design the Fast AM output states are ahead of the Slow AM by 3-4 outputs states. Consequently, the Fast encoder can consume the current as well as k future Fast AM output states, without incurring latency at the Joint encoder stage. Note that borrowing few future outputs from Fast AM is a key development in this work and is facilitated due to the parallel computation of the Fast AM and Slow AM threads. Other ASR technologies like TDNN [18] or Bidrectional LSTMs borrow future information but at significant hit to user-perceived latency. Note that an alternative design can also include a client-server architecture where the Fast AM runs on ASR-enabled devices with feedback from server. There, Fast AM outputs can be directly transmitted to server to improve the server ASR.

4. EXPERIMENTS AND RESULTS

We demonstrate our work on a large vocabulary unified ASR task. We conduct experiments on over 7200 hrs of unified English training data, and 200 hrs of test data. The data is anonymized with personal identifiable information removed, and collected from sources across Microsoft applications in Cortana, Desktop, Skype, and Xbox. The data spans natural conversion, dictation, command-and-control, and includes close-talk as well as far-field audio. Our baseline ASR, *i.e.* Slow AM, is a standard 6-layers Unidirectional LSTM model trained with cross-entropy (CE) [5] criterion. The corresponding LSTM cells consist of 1024 memory units, that's projected to 512 units at hidden layer stages. Our target consists of 9k acoustic states. The Fast AM consists of 4 Unidirectional LSTM layers, with 512 memory units projected to 256 units, and the $9k$ output targets are identical to that in Slow AM. Above design makes the Fast AM 4x smaller than Slow AM. We train models on 80-dim log-Mel filter-bank features. The feature processing time window is 25-ms with 10-ms window shift. We use a 5-gram language model with over 1M vocabulary.

Table 2. *WER [%] for Fast, Slow, and F&S AMs. We also report WERR [%] over Slow AM.*

Tasks	Fraction of All Data	Fast AM	Slow AM	F&S AM	Fast WERR	F&S WERR
All	100	16.5	14.8	14.1	-11.5	4.7
A	15	14.6	14.3	13.1	-2.2	8.3
B	25	16.6	15.9	14.8	-4.8	6.6
C	50	17.3	16.0	15.1	-7.6	5.7
D	75	17.1	15.6	14.9	-9.6	5.0

We report our 1^{st} set of results in Table 2. The individual Fast and Slow models are built on identical technology and trained on data in Table 1. The corresponding word error rate (WER) is 16.5% for Fast AM and 14.8% for Slow AM. Clearly, the -11.5% WER relative (WERR) for Fast AM over Slow AM is indicative of Fast AM being 4x smaller. We train F&S AM following the technique in Sec. 3. We borrow the already trained Fast AM, and train rest of the network in Fast, Slow and Joint encoder networks. The Fast and Joint encoders consist of respectively 1 and 2 UniLSTM layers, and $k = 3$ for Fast encoder input. In comparison to individual models in Fig. 1, the new parameters in F&S AM are primarily from the Fast encoder. We considered few choices for Fast AM outputs a_t in Sec. 3.1, and develop our work on the top hidden layer output for minimal new parameters and training efficiency. The Fast AM is shared between F&S AM and other speech applications. In our work we freeze all the Fast AM parameters except the top hidden layer. Overall we report 4.7% WERR from F&S AM over Slow AM in Table 2.

We find above results very significant because Fast AM by itself has a large regression over Slow AM. The F&S architecture demonstrates an effective approach to encode information from a weaker AM for strong gains on top of a better AM. For additional context, these improvements exceed what we noted for ROVER [19] model combination in [20], where we necessarily require models with comparable accuracy for any improvements. We believe that Fast encoder distills relevant information and encodes that to enrich the F&S AM. In Table 2 we also report WER for increasingly larger subset of "All" test data. The subset "A" represents 15% of All data. There Fast AM is within 2.2% relative of Slow AM, and F&S AM shows much larger 8.3% WERR over Slow AM. We also validate an expected trend between the WERR for Fast AM and F&S AM, and summarize that a stronger Fast AM leads to stronger overall gain

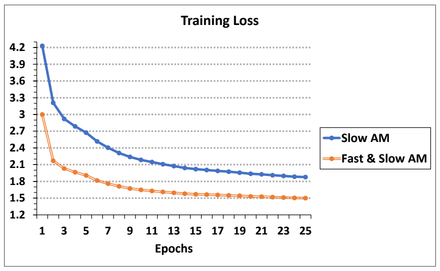

Fig. 3. *CE training loss for Slow and F&S AM.*

Table 3. *WER [%] for F&S AM, and WERR [%] over Slow AM.*

Tasks	Fast AM	Slow AM	F&S AM	Fast WERR	F&S WERR
Video (en-US)	20.5	18.7	17.1	-9.7	8.6
Dialects	14.4	13.5	12.3	-6.0	9.0

from F&S AM. The subset "D" shows 5% WERR for F&S AM even though the Fast AM is much weaker with -9.6% WERR. We also study the CE training loss across initial training epochs in Fig. 3, and note faster as well as stronger convergence with F&S AM. That illustrates significant merit in the convergence characteristics of the F&S AM and serves a useful reference for future developments.

In our F&S architecture we combined Fast encoder output with corresponding Slow encoder output. We also considered an alternative where we joined the Fast encoder output to the input of Slow encoder, *i.e.* to the features x_t for Slow encoder. That resulted in a WER of 14.3%. Above validates that, (a) the proposed F&S architecture offers WER gains over joining encoded Fast AM outputs to the ASR features x_t for Slow encoder, (b) our architecture offers latency advantages by computing the Slow and Fast encoders in parallel threads, while joining information at a later stage. We also report generalization aspects of F&S AM in Table 3. The evaluation consists of insufficiently represented tasks in our training, *e.g.*, Video and English dialects. We report a strong 8.6-9.0% WERR on those tasks and infer good generalization attributes in F&S AM.

4.1. Extensions in F&S modeling technique

Most of our work so far was in the context of seeking higher ASR accuracy goals within user-perceived latency limits. Next, we consider 2 extensions to further test the core F&S modeling technique.

(A) Additional Fast AM outputs for Fast Encoder - Our designed F&S AM encodes 3 future output states from Fast AM, *i.e.* $k = 3$ in (3). We consider an application where 3 additional Fast AM outputs may meet the latency requirements, and achieve 8.1% WERR from F&S AM with $k = 6$ in Table 4. That indicates a strong merit and scope in the k parameter selection in F&S AM, where we can select larger k to significantly boost ASR accuracy at minimal delay in ASR response. In Fig. 2 we also verified that encoding few future Fast AM outputs, that are already ahead of Slow AM path, is a crucial design, and its absence would diminish gains.

(B) Encoding Slow AM Outputs - We also studied an extension by replacing Fast AM with Slow AM, *i.e.* encoding Slow AM outputs (WER=14.8%) with $k = 6$ in our proposed architecture. This experiment pushes the work towards the best achievable ASR accuracy by encoding larger and more accurate outputs, and produces 12.2% WERR in Table 5. The study shows that the F&S AM architecture offers many avenues for further developments. Note that above approach also increases the model size to 2x. Therefore, we trained UniLSTM models with 8 or 10 LSTM layers for a relevant baseline. Those deeper models provided only $1-2\%$ WERR over our baseline Slow AM with 6 layers. These results are consistant with a prior observation in [16]. Overall we infer significant merit in F&S architecture. It mitigates vanishing gradient issues found in standard deep learning models, and significantly improves ASR accuracy for streaming ASR solutions.

Table 4. *WER [%] with encoding additional Fast AM outputs.*

Tasks	Slow AM	F&S AM (k=3)	F&S AM (k=6)
All	14.8	14.1	13.6
WERR [%]	-	4.7	8.1

Table 5. *WER [%] by replacing Fast AM with Slow AM in Fig. 2*

Tasks	Slow AM	Slow & Slow AM	WERR [%]
All	14.8	13.0	12.2

5. CONNECTIONS AND DISCUSSION

The goal of our research is to actively discover and build connections with related ASR technologies. We expect the F&S AM technique to have generic attributes and applications beyond the presented scope in hybrid acoustic models. The F&S AM technique resembles and draws motivations from the encoder-decoder in E2E and RNN-T systems [21, 22]. Subsequently in Fig. 2 we can also consider an approach like RNN-T by encoding y_{t-1} for Fast encoder. A future work can replace Fast AM with certain outputs from RNN-T or E2E models. We can also develop this work in purely RNN-T settings in F&S RNN-T AM.

We also noted connections with ROVER combination in Sec. 4. Furthermore, a prior work [20] conducted AM combination by training a separate prediction model. There the individual models being combined retain their unique identities. In contrast, our work trains the final model in complete context of Fast AM. That ensures greater knowledge distillation power in our work. Our work also connects with boosting approaches where a stronger model develops on a weaker model. We demonstrate among the first applications of boosting techniques for large scale deep learning model, where we build a superior model by encoding a weaker AM.

6. CONCLUSION

An ASR ecosystem creates applications where the core ASR and an application-specific Fast AM are evaluated concurrently. This work introduces a new framework in Fast & Slow model that encodes Fast AM outputs and improves the overall ASR accuracy. We also noted connections with E2E and RNN-T model architectures, and that our work is among the first to borrow some architectural components from E2E models for hybrid models. We developed a number of techniques and reported 4.7% WERR from F&S AM over Slow AM on 7200 hrs training task. We improved that to 8.1% with encoding a few more Fast AM outputs. We also developed extensions for 12.2% WERR.

7. REFERENCES

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Proc. Interspeech*, 2014, pp. 338–342.

[3] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, “Fast and accurate recurrent neural network acoustic models for speech recognition,” *arXiv preprint arXiv:1507.06947*, 2015.

[4] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[5] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] Michael L Seltzer, Dong Yu, and Yongqiang Wang, “An investigation of deep neural networks for noise robust speech recognition,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7398–7402.

[7] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael L. Seltzer, Geoffrey Zweig, Xiaodong He, Jason D. Williams, Yifan Gong, and Alex Acero, “Recent advances in deep learning for speech research at microsoft,” *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8604–8608, 2013.

[8] G.E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing,*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[10] Haşim Sak, Andrew Senior, and Françoise Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Fifteenth annual conference of the international speech communication association*, 2014.

[11] Mike Schuster and Kuldip K Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[12] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber, “Bidirectional lstm networks for improved phoneme classification and recognition,” in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 799–804.

[13] Kai Chen and Qiang Huo, “Training deep bidirectional lstm acoustic model for lvcsr by a context-sensitive-chunk bptt approach,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 7, pp. 1185–1193, 2016.

[14] Shaofei Xue and Zhijie Yan, “Improving latency-controlled blstm acoustic models for online speech recognition,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5340–5344.

[15] K. Kumar, C. Liu, Y. Gong, and J. Wu, “1-D row-convolution LSTM: Fast streaming ASR at accuracy parity with LC-BLSTM,” in *Interspeech*, 2020.

[16] Jinyu Li, Liang Lu, Changliang Liu, and Yifan Gong, “Improving layer trajectory lstm with future context frames,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6550–6554.

[17] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International conference on machine learning*, 2016, pp. 173–182.

[18] Vijayaditya Peddinti, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, “Low latency acoustic modeling using temporal convolution and lstms,” *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 373–377, 2017.

[19] Jonathan G Fiscus, “A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover),” in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, 1997, pp. 347–354.

[20] K. Kumar and Y. Gong, “Static and dynamic state predictions for acoustic model combination,” 2019, pp. 2782–2786.

[21] Hasim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays, “Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping,” in *Proc. of Interspeech*, 2017.

[22] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brake, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” *CoRR*, vol. abs/1508.04395, 2015.