INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

English Conversational Telephone Speech Recognition by Humans and Machines

George Saon[1], *Gakuto Kurata*[2], *Tom Sercu*[1], *Kartik Audhkhasi*[1], *Samuel Thomas*[1], *Dimitrios Dimitriadis*[1], *Xiaodong Cui*[1], *Bhuvana Ramabhadran*[1], *Michael Picheny*[1], *Lynn-Li Lim*[3], *Bergul Roomi*[3], *Phil Hall*[3]

[1]IBM Watson, Yorktown Heights, USA
[2]IBM Research - Tokyo, Japan
[3]Appen, Sydney, Australia

Abstract

Word error rates on the Switchboard conversational corpus that just a few years ago were 14% have dropped to 8.0%, then 6.6% and most recently 5.8%, and are now believed to be within striking range of human performance. This then raises two issues: what is human performance, and how far down can we still drive speech recognition error rates? In trying to assess human performance, we performed an independent set of measurements on the Switchboard and CallHome subsets of the Hub5 2000 evaluation and found that human accuracy may be considerably better than what was earlier reported, giving the community a significantly harder goal to achieve. We also report on our own efforts in this area, presenting a set of acoustic and language modeling techniques that lowered the WER of our system to 5.5%/10.3% on these subsets, which is a new performance milestone (albeit not at what we measure to be human performance). On the acoustic side, we use a score fusion of one LSTM with multiple feature inputs, a second LSTM trained with speaker-adversarial multi-task learning and a third convolutional residual net (ResNet). On the language modeling side, we use word and character LSTMs and convolutional WaveNet-style language models.

Index Terms: LSTM, ResNet, dilated convolutions, conversational speech recognition

1. Introduction

With the performance of ASR systems inching ever closer to that of humans, it is important to benchmark human performance accurately. In [1], the authors claim a human word error rate (WER) of 5.9%/11.3% on the Switchboard/CallHome subsets (SWB/CH) of the NIST Hub5 2000 evaluation testset. When compared with [2] which quotes a WER of 4%, the 5.9% estimate seemed rather high (albeit measured on different data). This intriguing discrepancy prompted us to launch our own human transcription effort in order to confirm (or disconfirm) the estimates from [1]. The findings from this effort were doubly surprising. First, we were expecting the SWB measurement to be closer to the Lippmann estimate of 4% but could only get down to 5.1% for the best transcriber after quality checks. Second, the same transcriber achieved a surprisingly low 6.8% WER for CallHome (we were expecting a much higher number based on the 11.3% estimate).

For comparison, our latest ASR system achieves 5.5%/10.3% WER on SWB/CH. This means that "human parity" is attainable on this particular Switchboard subset (although not achieved yet) but is a distant dream for the CallHome task. What makes the Switchboard and CallHome testsets so different one might ask? The biggest problem with the SWB testset is that 36 out of 40 test speakers appear in the training data, some in as many as 8 different conversations [3], and our acoustic models are very good at memorizing speech patterns seen during training. The second problem is that the SWB and CH tasks differ in the style of conversational speech: SWB consists of conversations between strangers while CH consists of calls between friends and family members. Speaking style between strangers tends to be more formal whereas the CallHome style is more casual making CH a harder task. The training data collected by LDC under the Switchboard and Fisher protocols is almost entirely Switchboard-like meaning that testing on CallHome is a mismatched scenario for ASR systems. Since ASR systems are generally not robust to mismatched training and testing conditions, it comes as no surprise that the degradation in performance from SWB to CH for ASR systems is larger than that of expert transcribers.

On the system side, we have simplified and improved our acoustic models considerably and experimented with more sophisticated language models such as LSTM and WaveNet LMs. Most of the AM improvement comes from LSTMs that operate on multiple features or use a different training criterion such as speaker-adversarial multi-task learning. Additionally, replacing the VGG convolutional nets [4] that we had in our last year's system [5] with ResNets [6] turned out to be beneficial for performance. On the LM side, adding LSTM word and character-based LMs resulted in substantial accuracy gains.

The rest of the paper is organized as follows: in section 2 we talk about the human transcription experiments; in section 3 we describe a series of system improvements pertaining to both acoustic and language modeling and in section 4 we summarize our findings.

2. Human transcription experiments

These experiments were carried out by IBM's preferred speech transcription provider, Appen. The transcription protocol that was agreed upon was to have three independent transcribers provide transcripts which were quality checked by a fourth senior transcriber. All four transcribers are native US English speakers and were selected based on the quality of their work on past transcription projects. The transcribers were familiarized with the LDC transcription guidelines which cover hyphenations, spelled abbreviations, contractions, partial words, non-speech sounds, etc. They were provided with separate audio files for each utterance, where the utterances are defined by the reference segmentation (1831 files for SWB, 2635 files for CH). The exact same files are processed by our ASR system. The transcription time was estimated at 12-14 times real-time (xRT) for the first pass for Transcribers 1-3 and an additional 1.7-2xRT for the second quality checking pass (by Transcriber 4). Both passes involved listening to the audio multiple times: around 3-4 times for the first pass and 1-2 times for the second. In order to use NIST's scoring tool `sclite`, we had to convert the transcripts into CTM files which have time-marked word boundary information. This was done by splitting the duration of the utterance uniformly across the number of words.

In Table 1 we show the error rates of the three transcribers before and after quality checking by the fourth transcriber as well as the human WER reported in [1]. Unsurprisingly, there is some variation among transcriber performance and the quality checking pass reduces the error rate across all transcribers. We surmise that the very different estimates for the human error rate for CallHome (6.8% versus 11.3%) can be attributed to a much lower deletion rate for our best human transcript.

Table 1: *Word error rates on SWB and CH for human transcribers before and after quality checking contrasted with the human WER reported in [1].*

	WER SWB	WER CH
Transcriber 1 raw	6.1	8.7
Transcriber 1 QC	5.6	7.8
Transcriber 2 raw	5.3	6.9
Transcriber 2 QC	**5.1**	**6.8**
Transcriber 3 raw	5.7	8.0
Transcriber 3 QC	5.2	7.6
Human WER from [1]	5.9	11.3

3. System improvements

In this section we discuss the training data and testsets that were used as well as improvements in acoustic and language modeling. The training set for our acoustic models consists of 262 hours of Switchboard 1 audio with transcripts provided by Mississippi State University, 1698 hours from the Fisher data collection and 15 hours of CallHome audio. In order to allay fears that we may be overfitting to the Hub5 2000 testsets by extensively testing on them, we have decided to report results on three more testsets: RT'02 (6.4h, 120 speakers, 64K words), RT'03 (7.2h, 144 speakers, 76K words) and RT'04 (3.4h, 72 speakers, 36.7K words).

In [7], we have shown that convolutional and non-convolutional AMs have comparable performance and good complementarity. Hence, the strategy for our previous systems [8, 5] was to use a combination of recurrent and convolutional nets. For example, in last year's system we used a score fusion of three models which share the same decision tree: unfolded RNNs with maxout activations, LSTMs and VGG nets. This year, in order to simplify and enhance the overall architecture, we eliminated the maxout RNN, we improved the LSTMs and we replaced the VGG nets with residual nets (ResNets).

3.1. ResNet acoustic models

On the convolutional network acoustic modeling side, we trained residual networks with pre-activation identity shortcut connections. Residual Networks were introduced for image recognition in [9] and used in speech recognition in [1, 10]. The novelty of residual networks is to introduce shortcut connections between so-called "blocks" of convolutional layers, which was argued to improve the flow of information and gradients, and allows training even deeper CNNs without the optimization problem occuring without the residual connections.

Table 2: *ResNet architectures and results. Decoding with small LM (4M n-grams). In the bottom rows (results on test-sets). XE-300 indicates the network was cross-entropy trained on the 300h SWB corpus only, XE and ST for training on the 2000h SWB+Fisher corpus. Column (d) has best performance, compared against 3 different ablation variants: (a) with bottleneck blocks and without pooling, (b) without pooling, and (c) less depth. The size of the output of the 3×3 convolutions is indicated for each stage.*

	(a)	(b)	(c)	(d)
Summary	Bottleneck 1-3333	1-3333 NoTimestride	1-2222 Timestride	1-3333 Timestride
# param	64.3 M	67.1 M	60.8 M	67.1 M
Input	$3 \times 64 \times 31$	$3 \times 64 \times 55$	$3 \times 64 \times 56$	$3 \times 64 \times 76$
Stage 0 64x32xT	conv5x5, 64 maxpool (2x1)	conv5x5, 64 maxpool (2x1)	conv5x5, 64 maxpool (2x1)	conv5x5, 64 maxpool (2x1)
Stage 1 (64x32xT)	*initStride 1x1* 3x [conv 1x1, 64 conv 3x3, 64 conv 1x1, 256]	*initStride 1x1* 3x [conv 3x3, 64 conv 3x3, 64]	*initStride 1x1* 2x [conv 3x3, 64 conv 3x3, 64]	*initStride 1x1* 3x [conv 3x3, 64 conv 3x3, 64]
Stage 2 (128x16xT)	*initStride 2x1* 3x [conv 1x1, 128 conv 3x3, 128 conv 1x1, 512]	*initStride 2x1* 3x [conv 3x3, 128 conv 3x3, 128]	*initStride 2x1* 2x [conv 3x3, 128 conv 3x3, 128]	*initStride 2x1* 3x [conv 3x3, 128 conv 3x3, 128]
Stage 3 (256x8xT)	*initStride 2x1* 3x [conv 1x1, 256 conv 3x3, 256 conv 1x1, 1024]	*initStride 2x1* 3x [conv 3x3, 256 conv 3x3, 256]	*initStride 2x1* 2x [conv 3x3, 256 conv 3x3, 256]	*initStride 2x1* 3x [conv 3x3, 256 conv 3x3, 256]
Stage 4 (512x4xT)	*initStride 2x1* 3x [conv 1x1, 512 conv 3x3, 512 conv 1x1, 2048] maxpool (2x1)	*initStride 2x1* 3x [conv 3x3, 512 conv 3x3, 512] maxpool (2x1)	*initStride* **2x2** 2x [conv 3x3, 512 conv 3x3, 512] maxpool (**2x2**)	*initStride* **2x2** 3x [conv 3x3, 512 conv 3x3, 512] maxpool (**2x2**)
Output	3x FC 2084 FC 1024 FC 32k	3x FC 2084 FC 1024 FC 32k	3x FC 2084 FC 1024 FC 32k	3x FC 2084 FC 1024 FC 32k
(XE-300) SWB	11.8	11.2	11.3	11.4
(XE) SWB		9.7	9.5	9.2
(ST) SWB		8.6	8.7	8.3
(ST) CH		15.5	15.0	14.9
(ST) RT'02		13.4	13.3	13.1
(ST) RT'03		13.1	12.7	12.7
(ST) RT'04		12.1	12.0	11.9

Table 2 shows four residual network model architectures and their performance on the testsets with a small n-gram LM. We achieved best results with basic residual blocks without bottleneck, similar to the observations from [11] on CIFAR and SVHN experiments. However, bottleneck residual blocks could possibly be optimal with a larger computational budget. The input to our network are VTLN-warped logmel features with 64 mel bins. We perform data-balancing according to [13] with exponent $\gamma = 0.8$. We use full pre-activation identity shortcut connections which keep a clean information path [12] without nonlinearity after addition. For batch normalization the statistics are accumulated per feature map and per frequency bin following [15].

Let us now consider how to use strided pooling and strided convolutions, and the relation to time-dilated convolutions. First off, in the frequency direction, similar as for images, convolutions are padded so they do not reduce the size. Rather, the size is reduced by a factor of 2 through convolutions with stride 2. In Table 2, the "initStride" field on the first line of each stage indicates the (frequency x time) stride for the first block of that stage, where the number of feature maps is increased. This stride applies to both the first 3×3 convolution of the block, and the 1×1 convolution in the projection shortcut. The output feature map size is indicated in the left column for each stage.

Secondly, along the time direction, strided convolutions and strided pooling is optional, but was found to improve performance [15]. In Table 2, Stage 4, column (c) and (d), bold indicates striding in time. Note that, when adding time-strided conv and pool to an architecture, we need to increase the input context window to compensate for the additional size reduction. For residual networks, similar as for VGG-style networks, we indeed observe that time-strided time-pooling improves performance, see column (b) vs (d).

When transitioning from cross-entropy (XE) to sequence training (ST), we want to modify our network to do dense prediction efficiently [15]. This means the intermediate states of the convolutional layers and output of the ResNet should maintain inputs full time-resolution, i.e. it should produce an output CD state distribution for each input frame. We can achieve this by using time-dilated convolutions according to the same recipe as in [15]: for each layer which originally strides in time with factor 2, set time-stride to 1 and dilate with factor 2 all consecutive convolutions, maxpooling and fully connected layers. This includes the projection shortcut in the first block of each stage, though dilation for these 1×1 convolutions is irrelevant. After these modifications, the residual net can be used for dense prediction on sequences.

The ResNet which we will use in further sections is in Table 2 (d). It has 12 residual blocks, 30 weight layers and 67.1 M parameters. We trained this model using Nesterov accelerated gradient with learningrate 0.03 and momentum 0.99. Implementation of the CNN was also done in Torch with cuDNN v5.0 backend. Cross-entropy training took about 80 days for 1.5 billion samples, on 2 Nvidia K80 GPU's (4 devices) with batch size 64 per GPU and full synchronization between every minibatch. We sequence trained this model for 200M frames with the boosted MMI criterion [16].

3.2. LSTM acoustic models

All models presented here share the following characteristics. Their architecture consists in 6 bidirectional layers with 1024 cells per layer (512 per direction), one linear bottleneck layer with 256 units and an output layer with 32K units corresponding to as many context-dependent HMM states (shown on the left side of Figure 1). Training is done on non-overlapping subsequences of 21 frames where each frame consists of 40-dimensional FMLLR features to which we append 100-dimensional i-vectors. We group subsequences from different utterances into minibatches of size 128 for processing speed and reliable gradient estimates. The training consists of 14 passes of cross-entropy followed by 1 pass of SGD sequence training using the boosted MMI criterion [16] smoothed by adding the scaled gradient of the cross-entropy loss [17]. Implementation of the LSTM was done in Torch [18] with cuDNN v5.0 backend. Cross-entropy training for each model took about 2 weeks for 700M samples/epoch, on a single Nvidia K80 GPU device. One set of experiments was centered around the use of speaker-adversarial multi-task learning (SA-MTL). Inspired by the ideas of domain-adversarial neural networks [19] and noise-adversarial MTL [20], we experiment with training a speaker classifier in addition to the main CD-HMM state classifier in order to suppress the effects of speaker variability on ASR performance. Since i-vectors are a good low-dimensional representation of a speaker, we decided to train the speaker classifier to predict the i-vector inputs using an MSE loss function. The speaker classifier has one sigmoid layer and one hyperbolic tangent layer as shown in Figure 1.

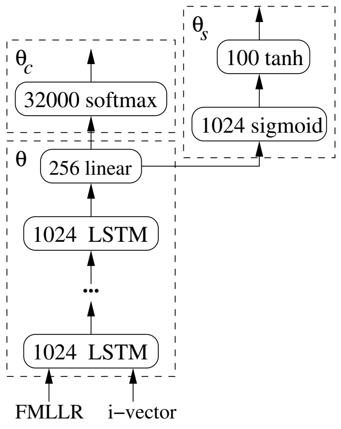

Figure 1: *LSTM with speaker-adversarial MTL architecture.*

If we denote by θ, θ_c, θ_s the parameters of the common LSTM, the main classifier (weights of linear layer before softmax) and the speaker classifier, the SGD update is done according to:

$$
\begin{aligned}
\hat{\theta}_c &= \theta_c - \epsilon \frac{\partial \mathcal{L}_{CE}(\mathbf{x})}{\partial \theta_c} \\
\hat{\theta}_s &= \theta_s - \epsilon \frac{\partial \mathcal{L}_{MSE}(\mathbf{x})}{\partial \theta_s} \\
\hat{\theta} &= \theta - \epsilon \left(\frac{\partial \mathcal{L}_{CE}(\mathbf{x})}{\partial \theta} - \lambda \frac{\partial \mathcal{L}_{MSE}(\mathbf{x})}{\partial \theta} \right)
\end{aligned}
$$

where **x** denotes a minibatch, $\mathcal{L}_{CE}$, $\mathcal{L}_{MSE}$ denote respectively the cross-entropy loss of the main classifier and the mean-squared error loss of the i-vector classifier, λ is a scaling parameter (typically set to 0.1), and ϵ is the learning rate. After the model is trained, the i-vector classifier branch is discarded at test time. As can be seen from Table 3 rows 1 and 2, we observe some small gains across all testsets which are also due in part to reestimating the VTLN warp factors and FMLLR transforms using an LSTM decoding output (old factors and transforms were based off a GMM decoding).

Last but not least, the largest improvement in LSTM modeling was achieved through feature fusion. We built an LSTM trained on fused FMLLR+i-vector+Logmel+Δ+$\Delta\Delta$ features the standard way (without speaker-adversarial MTL). The WER improvement from adding the Logmel features is indicated in Table 3 rows 1 and 3. Finally, we note that the feature fusion LSTM compares favorably with other single acoustic models from the literature as mentioned in [21] (Table 4).

Table 3: *Word error rates for LSTMs, ResNet and frame-level score fusion results across all testsets (36M n-gram LM).*

Model	SWB	CH	RT'02	RT'03	RT'04
LSTM (baseline)	7.7	14.0	11.8	11.4	10.8
LSTM1 (SA-MTL)	7.6	13.6	11.5	11.0	10.7
LSTM2 (Feat. fusion)	7.2	12.7	10.7	10.2	10.1
ResNet	7.6	14.5	12.2	12.2	11.5
ResNet+LSTM2	6.8	12.2	10.2	10.0	9.7
ResNet+LSTM1+LSTM2	6.7	12.1	10.1	10.0	9.7

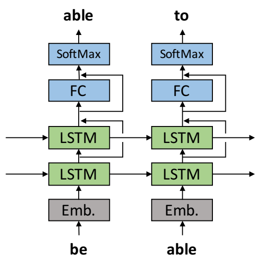

Figure 2: *Word-LSTM*

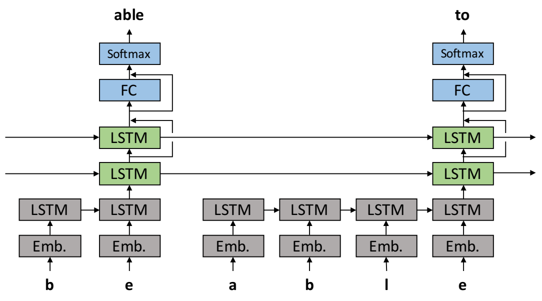

Figure 3: *Char-LSTM*

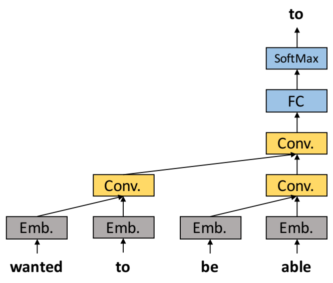

Figure 4: *Word-DCC*

Table 4: *WER on SWB and CH with various LM configurations.*

	WER [%]	
	SWB	CH
n-gram	6.7	12.1
n-gram + model-M	6.1	11.2
n-gram + model-M + Word-LSTM	5.6	10.4
n-gram + model-M + Char-LSTM	5.7	10.6
n-gram + model-M + Word-LSTM-MTL	5.6	10.3
n-gram + model-M + Char-LSTM-MTL	5.6	10.4
n-gram + model-M + Word-DCC	5.8	10.8
n-gram + model-M + 4 LSTMs + DCC	**5.5**	**10.3**

3.3. Language modeling improvements

In addition to n-gram and model-M used in our previous system [5], we introduced LSTM-based as well as convolution-based LMs in this paper.

We experimented with four LSTM LMs, namely *Word-LSTM*, *Char-LSTM*, *Word-LSTM-MTL*, and *Char-LSTM-MTL*. The Word-LSTM had one word-embeddings layer, two LSTM layers, one fully-connected layer, and one softmax layer, as shown in Figure 2. The upper LSTM layer and the fully-connected layer were wrapped by residual connections [6]. Dropout was only applied to the vertical dimension and not applied to the time dimension [22]. The Char-LSTM added an additional LSTM layer to estimate word-embeddings from character sequences as illustrated in Figure 3 [23]. Both Word-LSTM and Char-LSTM used the cross-entropy loss of predicting the next word given its history as objective function, similar to conventional LMs. In addition, we introduced multi-task learning (MTL) in Word-LSTM-MTL and Char-LSTM-MTL by adding a word class prediction loss where the classes are determined using Brown clustering [24] of the vocabulary words.

We also experimented with a convolution-based LM in the form of dilated causal convolution as used in WaveNet [25]. The resulting model is called *Word-DCC* and consists of word-embeddings layer, causal convolution layers with dilation, convolution layers, fully-connected layers, softmax layer, and residual connections. The actual number of layers and dilation/window sizes were determined using heldout data (Figure 4 has a simple configuration for illustration purposes).

All LMs are trained with the same vocabulary of 85K words from [5] and a corpus of 560M words consisting of publicly available text data from LDC, including Switchboard, Fisher, Gigaword, and Broadcast News and Conversations. The models are further finetuned with only the transcripts of the 1975 hours audio data (24M words). The learning rate is controled by ADAM [26] and we introduced a self-stabilization term to coordinate the layer-wise learning rates [27]. The hyper-parameters were tuned based on the perplexity of the heldout data which is a subset of the acoustic transcripts. The approximate number of parameters for each model was 90M to 130M.

We first generated word lattices with the n-gram LM and our best acoustic model consisting of ResNet and two LSTMs. Then we rescored the word lattices with the model-M and generated n-best lists from the rescored lattice. Finally, we applied the four LSTM-based LMs and the convolution-based LM. Table 4 shows the WERs on SWB and CH with various LM configurations whereas Table 5 shows the improvements due to LM rescoring for all testsets. More details about the language modeling are given in a companion paper [28].

Table 5: *Word error rates for the different LM rescoring steps across all testsets.*

	SWB	CH	RT'02	RT'03	RT'04
n-gram	6.7	12.1	10.1	10.0	9.7
+ model-M	6.1	11.2	9.4	9.4	9.0
+ LSTM + DCC	5.5	10.3	8.3	8.3	8.0

4. Conclusion

We have presented a set of acoustic and language modeling improvements to our English Switchboard system that resulted in a new record word error rate on this task. On the acoustic side, we improved our bidirectional LSTMs through feature fusion and speaker-adversarial training and we replaced VGG nets with ResNets. On the language modeling side, we exploited the same complementarity between recurrent and convolutional architectures by adding word and character-based LSTM LMs and a convolutional WaveNet LM.

We do not believe that human parity has been reached on this task. First, the Hub5'00 SWB testset has a large overlap between training and test speakers which results in ASR systems having deceptively good performance. A more realistic level of performance is the average WER across all testsets which is around 8% for our system. Second, the human WER of expert transcribers that were asked to do a high-quality job is lower than what was previously reported. On an optimistic note, this means that the future of research in conversational speech recognition looks bright for at least a few more years.

5. References

[1] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “Achieving human parity in conversational speech recognition,” *arXiv preprint arXiv:1610.05256*, 2016.

[2] R. P. Lippmann, “Speech recognition by machines and humans,” *Speech communication*, vol. 22, no. 1, pp. 1–15, 1997.

[3] J. Fiscus, W. M. Fisher, A. F. Martin, M. A. Przybocki, and D. S. Pallett, “2000 nist evaluation of conversational speech recognition over the telephone: English and mandarin performance results,” in *Proc. Speech Transcription Workshop*. Citeseer, 2000, pp. 1–5.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR arXiv:1409.1556*, 2014.

[5] G. Saon, T. Sercu, S. Rennie, and H.-K. Kuo, “The IBM 2016 English conversational speech recognition system,” in *Seventeenth Annual Conference of the International Speech Communication Association*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016, pp. 770–778.

[7] H. Soltau, G. Saon, and T. N. Sainath, “Joint training of convolutional and non-convolutional neural networks,” *to Proc. ICASSP*, 2014.

[8] G. Saon, H.-K. Kuo, S. Rennie, and M. Picheny, “The IBM 2015 English conversational speech recognition system,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR arXiv:1512.03385*, 2015.

[10] Y. Zhang, W. Chan, and N. Jaitly, “Very deep convolutional networks for end-to-end speech recognition,” *Proc. ICASSP*, 2017.

[11] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” *arXiv preprint arXiv:1603.05027*, 2016.

[13] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, “Very deep multilingual convolutional neural networks for lvcsr,” *Proc. ICASSP*, 2016.

[14] T. Sercu and V. Goel, “Advances in very deep convolutional neural networks for lvcsr,” *arXiv*, 2016.

[15] ——, “Dense prediction on sequences with time-dilated convolutions for speech recognition,” *NIPS End-to-end Learning for Speech and Audio Processing Workshop*, 2016.

[16] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, “Boosted MMI for model and feature-space discriminative training,” in *Proc. of ICASSP*, 2008, pp. 4057–4060.

[17] H. Su, G. Li, D. Yu, and F. Seide, “Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription,” *Proc. ICASSP*, 2013.

[18] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[19] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[20] Y. Shinohara, “Adversarial multi-task learning of deep neural networks for robust speech recognition,” *Interspeech 2016*, pp. 2369–2372, 2016.

[21] L. Hairong, Z. Zhu, X. Li, and S. Satheesh, “Gram-ctc: Automatic unit selection and target decomposition for sequence labelling,” *CoRR arXiv:1703.00096*, 2017.

[22] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” *arXiv preprint arXiv:1409.2329*, 2014.

[23] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, “Finding function in form: Compositional character models for open vocabulary word representation,” *arXiv preprint arXiv:1508.02096*, 2015.

[24] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, “Class-based n-gram models of natural language,” *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[25] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.

[26] D. Kingma and J. Ba, “ADAM: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[27] P. Ghahremani and J. Droppo, “Self-stabilized deep neural network,” in *Proc. ICASSP*, 2016, pp. 6645–6649.

[28] G. Kurata, A. Sethy, B. Ramabhadran, and G. Saon, “Empirical exploration of LSTM and CNN language models for speech recognition,” *Submitted to Interspeech 2017*, 2017.