

English Conversational Telephone Speech Recognition by Humans and Machines

George Saon, Gakuto Kurata, Tom Sercu*

*Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis
Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny*

IBM Watson, Yorktown Heights, USA

*IBM Research - Tokyo, Japan

Lynn-Li Lim, Bergul Roomi, Phil Hall

Appen, Sydney, Australia

Abstract

One of the most difficult speech recognition tasks is accurate recognition of human to human communication. Advances in deep learning over the last few years have produced major speech recognition improvements on the representative Switchboard conversational corpus. Word error rates 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? A recent paper by Microsoft suggests that we have already achieved human performance. In trying to verify this statement, we performed an independent set of human performance measurements on two conversational tasks and found that human performance 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 word error rate of our own English conversational telephone LVCSR system to the level of 5.5%/10.3% on the Switchboard/CallHome subsets of the Hub5 2000 evaluation, which - at least at the writing of this paper - 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 three models: one LSTM with multiple feature inputs, a second LSTM trained with speaker-adversarial multi-task learning and a third residual net (ResNet) with 25 convolutional layers and time-dilated convolutions. 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.

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. After receiving the transcripts, the following filtering rules were applied:

- All non-speech markers were tagged as non-lexical items which are ignored during scoring. Examples of non-speech markers are: [laughter], [breathing], [noise], {no speech}, etc.
- Other markers such as '...', '–', '(())' were eliminated prior to scoring.
- All partial words ending in '-' were marked as non-lexical items.
- All punctuation marks such as '!', '‘’', '!’ and '?’ were eliminated prior to scoring.

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.

	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

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].

Additionally, in Tables 2 and 3, we take a closer look at the most frequent substitution, deletion and insertion errors for our system output and the best human transcript after quality checking. While many of the errors look similar to those reported in [1], there is a glaring discrepancy in the frequency of top deletions for CallHome between our human transcript and theirs. This suggests 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.

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

SWB		CH	
ASR	Human	ASR	Human
11: and / in	16: (%hes) / oh	21: was / is	28: (%hes) / oh
9: was / is	12: was / is	16: him / them	22: was / is
7: it / that	7: (i-) / %hes	15: in / and	11: (%hes) / %bcack
6: (%hes) / oh	5: (%hes) / a	8: a / the	10: bentsy / benji
6: him / them	5: (%hes) / hmm	8: and / in	10: yeah / yep
6: too / to	5: (a-) / %hes	8: is / was	9: a / the
5: (%hes) / i	5: could / can	8: two / to	8: is / was
5: then / and	5: that / it	7: the / a	7: (%hes) / a
4: (%hes) / %bcack	4: %bcack / oh	7: too / to	7: the / a
4: (%hes) / am	4: and / in	6: (%hes) / a	7: well / oh

Table 2: Most frequent substitution errors for humans and ASR system on SWB and CH.

Deletions				Insertions			
SWB		CH		SWB		CH	
ASR	Human	ASR	Human	ASR	Human	ASR	Human
30: it	19: i	46: i	20: i	13: i	16: is	23: a	17: is
20: i	17: it	46: it	18: and	10: a	14: %hes	14: is	17: it
17: that	16: and	39: and	15: it	7: and	12: i	11: i	16: and
16: a	14: that	32: is	15: the	7: of	11: and	10: are	14: have
14: and	14: you	26: oh	14: is	6: you	9: it	10: you	13: a
14: oh	12: is	25: a	15: not	5: do	6: do	9: the	13: that
14: you	12: the	20: to	10: a	5: the	5: have	8: have	12: i
12: %bcack	11: a	19: that	10: in	5: yeah	5: yeah	8: that	11: %hes
12: the	10: of	19: the	10: that	4: air	5: you	7: and	10: not
11: to	9: have	18: %bcack	10: to	4: in	4: are	7: it	9: oh

Table 3: Most frequent deletion and insertion errors for humans and ASR system on SWB and CH.

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 a variety of testsets. Since the RT’02, RT’03, RT’04 and DEV’04f testsets have not been used in more than a decade, we are fairly confident that performance improvements on these testsets are indicative of real progress. Statistics about all the testsets used in the experiments are given in Table 4.

Testset	Duration	Nb. speakers	Nb. words
Hub5’00 SWB	2.1h	40	21.4K
Hub5’00 CH	1.6h	40	21.6K
RT’02	6.4h	120	64.0K
RT’03	7.2h	144	76.0K
RT’04	3.4h	72	36.7K
DEV’04f	3.2h	72	37.8K

Table 4: Testsets that are used to report experimental results.

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. LSTM acoustic models

All models presented here share the following characteristics. Their architecture consists in 4-6 bidirectional layers with 1024 cells per layer (512 per direction), one linear bot-

tleneck 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 [9] smoothed by adding the scaled gradient of the cross-entropy loss [10]. Implementation of the LSTM was done in Torch [11] 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.

The first two improvements are fairly banal and consist in increasing the number of layers from 4 (like in our previous model [5]) to 6 and in realigning the training data with a 6-layer LSTM and retraining another LSTM. The effect of these steps is shown in the first three rows of Table 5 across all testsets.

LSTM	SWB	CH	RT'02	RT'03	RT'04	DEV'04f
4-layer	8.0	14.3	12.2	11.6	11.0	10.8
6-layer	7.7	14.0	11.8	11.4	10.8	10.4
Realigned	7.7	13.8	11.7	11.2	10.8	10.2
SA-MTL	7.6	13.6	11.5	11.0	10.7	10.1
Feat. fusion	7.2	12.7	10.7	10.2	10.1	9.6

Table 5: Word error rates for LSTM AMs across all testsets (36M n-gram LM).

The second set of experiments was centered around the use of speaker-adversarial multi-task learning (SA-MTL). In [12], the authors introduce domain-adversarial neural networks which are models that are trained to not distinguish between in-domain, labeled data and out-of-domain, unlabeled data. This is achieved by training a domain classifier in parallel with the main classifier and by subtracting the gradient component from the domain classifier when estimating the parameters of the main classifier. This idea has been successfully applied in speech by [13] in the context of noise robustness where the author proposes noise-adversarial MTL to suppress the effects of noise. Here, 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.

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 \mathbf{x} denotes a minibatch, \mathcal{L}_{CE} , \mathcal{L}_{MSE} denote respectively

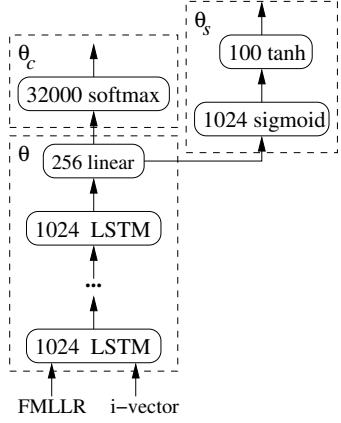


Figure 1: *LSTM with speaker-adversarial MTL architecture.*

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 5 rows 3 and 4, 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. The thought process leading to this experiment was that we wanted to add utterance-level information to our models which were only looking at a window of 21 consecutive frames. One possibility was to train an end-to-end LSTM using CTC as in [14, 15, 16, 17] and append the features from the last LSTM layer before the softmax to our existing features. This experiment worked quite well however, upon closer inspection, it turned out that the CTC model used a different set of input features: Logmel+ Δ + $\Delta\Delta$ instead of PLP followed by LDA and FMLLR. The question then naturally arose whether the gains came from CTC modeling or from the different input representations. To answer this question, 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, indicated in Table 5 rows 3 and 5, is the same as with CTC features meaning that the CTC modeling step was not needed. Finally, we note that the feature fusion LSTM compares favorably with other single acoustic models from the literature as mentioned in [17] (Table 4).

3.2. 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 [18] and used in speech recognition in [1, 19]. 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 occurring without the residual connections.

Table 6 shows four residual network model architectures

	(a)	(b)	(c)	(d)
Summary	Bottleneck 1-3333	1-3333 NoTimStride	1-2222 TimStride	1-3333 TimStride
# 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	conv5x5, 64	conv5x5, 64	conv5x5, 64	conv5x5, 64
$64 \times 32 \times T$	maxpool (2x1)	maxpool (2x1)	maxpool (2x1)	maxpool (2x1)
Stage 1 ($64 \times 32 \times T$)	<i>initStride 1x1</i> $3 \times [\text{conv } 1 \times 1, 64$ $\text{conv } 3 \times 3, 64$ $\text{conv } 1 \times 1, 256]$	<i>initStride 1x1</i> $3 \times [\text{conv } 3 \times 3, 64$ $\text{conv } 3 \times 3, 64]$	<i>initStride 1x1</i> $2 \times [\text{conv } 3 \times 3, 64$ $\text{conv } 3 \times 3, 64]$	<i>initStride 1x1</i> $3 \times [\text{conv } 3 \times 3, 64$ $\text{conv } 3 \times 3, 64]$
Stage 2 ($128 \times 16 \times T$)	<i>initStride 2x1</i> $3 \times [\text{conv } 1 \times 1, 128$ $\text{conv } 3 \times 3, 128$ $\text{conv } 1 \times 1, 512]$	<i>initStride 2x1</i> $3 \times [\text{conv } 3 \times 3, 128$ $\text{conv } 3 \times 3, 128]$	<i>initStride 2x1</i> $2 \times [\text{conv } 3 \times 3, 128$ $\text{conv } 3 \times 3, 128]$	<i>initStride 2x1</i> $3 \times [\text{conv } 3 \times 3, 128$ $\text{conv } 3 \times 3, 128]$
Stage 3 ($256 \times 8 \times T$)	<i>initStride 2x1</i> $3 \times [\text{conv } 1 \times 1, 256$ $\text{conv } 3 \times 3, 256$ $\text{conv } 1 \times 1, 1024]$	<i>initStride 2x1</i> $3 \times [\text{conv } 3 \times 3, 256$ $\text{conv } 3 \times 3, 256]$	<i>initStride 2x1</i> $2 \times [\text{conv } 3 \times 3, 256$ $\text{conv } 3 \times 3, 256]$	<i>initStride 2x1</i> $3 \times [\text{conv } 3 \times 3, 256$ $\text{conv } 3 \times 3, 256]$
Stage 4 ($512 \times 4 \times T$)	<i>initStride 2x1</i> $3 \times [\text{conv } 1 \times 1, 512$ $\text{conv } 3 \times 3, 512$ $\text{conv } 1 \times 1, 2048]$ maxpool (2x1)	<i>initStride 2x1</i> $3 \times [\text{conv } 3 \times 3, 512$ $\text{conv } 3 \times 3, 512]$ maxpool (2x1)	<i>initStride 2x2</i> $2 \times [\text{conv } 3 \times 3, 512$ $\text{conv } 3 \times 3, 512]$ maxpool (2x2)	<i>initStride 2x2</i> $3 \times [\text{conv } 3 \times 3, 512$ $\text{conv } 3 \times 3, 512]$ maxpool (2x2)
Output	$3 \times \text{FC } 2084$ $\text{FC } 1024$ $\text{FC } 32k$	$3 \times \text{FC } 2084$ $\text{FC } 1024$ $\text{FC } 32k$	$3 \times \text{FC } 2084$ $\text{FC } 1024$ $\text{FC } 32k$	$3 \times \text{FC } 2084$ $\text{FC } 1024$ $\text{FC } 32k$
(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
(ST) DEV'04f		11.3	11.1	11.2

Table 6: 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.

and their performance on the testsets with small LM. We achieved best results with basic residual blocks without bottleneck, similar to the observations from [20] 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 [22] with exponent $\gamma = 0.8$. We use full pre-activation identity shortcut connections which keep a clean information path [21] without nonlinearity after addition. For batch normalization the statistics are accumulated per feature map and per frequency bin following [24].

In order to use residual networks for acoustic modeling, we need to adapt the residual blocks (see Figure 2), while taking efficient convolution on sequences into account. In ResNets for image classification the convolutional pathway only includes padded convolutions, so does not reduce the size of the feature maps. The addition with the shortcut pathway is trivial, since both feature maps have the same size. In contrast, for convolutions on sequences we can not pad along the time directions. Padding along the time direction would modify the values on the edges based on the input sliding window location, thus making efficient convolution over a full utterance impossible (see [23]). So we do not pad the convolutions in time and as a consequence, the convolutional pathway reduces the size of the feature maps along the time direction. In this case, we need to crop on the shortcut connection to match the size of the feature maps coming out of the convolutional pathway. It is important to note that this does not impact the ability to convolve the residual net over full utterances at once: since the values at the edges are computed the same as everywhere else, they are

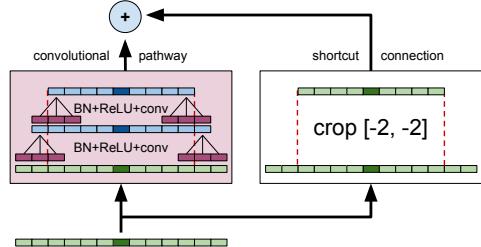


Figure 2: Residual connections on sequences. The convolutions are unpadded and reduce the size of the feature maps in the time direction (indicated with red dashed lines). To match this reduction, we simply crop the edges along the time on the shortcut connection.

independent of the position of the input window.

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 6, the “initStride” field on the first line of each stage indicates the (frequency \times 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 [24]. In Table 6, 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 [24]. 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 [24]: 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 6 (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 mini-batch. We sequence trained this model for 200M frames with the boosted MMI criterion [9].

3.3. Model combination

In Table 7 we report the performance of the best individual models described in the previous paragraphs as well as the results after frame-level score fusion across all testsets. All decodings are done with an 85K word vocabulary and a 4-gram language model with 36M n-grams. We note that LSTMs and ResNets exhibit a strong complementarity which improves the WER for all testsets.

Model	SWB	CH	RT'02	RT'03	RT'04	DEV'04f
LSTM1 (SA-MTL)	7.6	13.6	11.5	11.0	10.7	10.1
LSTM2 (Feat. fusion)	7.2	12.7	10.7	10.2	10.1	9.6
ResNet	7.6	14.5	12.2	12.2	11.5	11.1
ResNet+LSTM2	6.8	12.2	10.2	10.0	9.7	9.4
ResNet+LSTM1+LSTM2	6.7	12.1	10.1	10.0	9.7	9.2

Table 7: Word error rates for LSTMs and ResNet and frame-level score fusion results across all testsets (36M n-gram LM).

3.4. 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 3. 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 [25]. The Char-LSTM added an additional LSTM layer to estimate word-embeddings from character sequences as illustrated in Figure 4 [26]. 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. We first clustered the vocabulary using Brown clustering [27]. When training Word-LSTM-MTL and Char-LSTM-MTL, weighted summation of cross-entropy of predicting next word given its history and next class given its history was used as objective function.

Inspired by the complementarity of convolutional and non-convolutional acoustic models, we experimented with a convolution-based LM in the form of dilated causal convolution as used in WAVENET [28]. 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 5 has a simple configuration for illustration purposes).

For these five LMs, the training data and training procedures are common and described below:

- We used the same vocabulary of 85K words from [5].
- We first train the LM with a corpus of 560M words consisting of publicly available text data from LDC, including Switchboard, Fisher, Gigaword, and Broadcast News and Conversations. Then, starting from the trained model, we further train the LM with only the transcripts of the 1975 hours audio data used to train the acoustic model, consisting of 24M words.

	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

Table 8: *WER on SWB and CH with various LM configurations.*

- We controlled the leaning rate by ADAM [29] and introduced a self-stabilization term to coordinate the layer-wise learning rates [30].
- For all models, we tuned the hyper-parameters 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. Note that LM probabilities were linearly interpolated and the interpolation weights of all LMs were estimated using the heldout data.

Table 8 shows WER on SWB and CH with various LM configurations. The LSTM-based LMs show significant improvements over the strong n-gram + model-M results. The Word-DCC also has a marginal improvement over the n-gram + model-M. The effect of multi-task learning was confirmed especially on CH. Among the five LSTM-based and convolution-based LMs, word-LSTM-MTL achieved the best WER of 5.6% and 10.3% on SWB and CH respectively. By combining five LMs on top of n-gram + model-M, we achieved 5.5% and 10.3% WER for SWB and CH respectively. Lastly, we summarize the improvements due to the various language model rescoring steps across all testsets in Table 9. We noticed that the testset references have inconsistent transcription conventions with regards to spellings which are not followed by periods for SWB and CH (e.g. T V) and followed by periods for the other testsets (such as T. V.). The last line of Table 9 shows the WERs when periods are removed from both the references and system outputs by adding the filtering rules A. => A ... Z. => Z to the GLM file.

More details about the language modeling are given in a companion paper [31].

	SWB	CH	RT'02	RT'03	RT'04	DEV'04f
n-gram	6.7	12.1	10.1	10.0	9.7	9.2
+ model-M	6.1	11.2	9.4	9.4	9.0	8.8
+ LSTM + DCC	5.5	10.3	8.3	8.3	8.0	8.0
‘.’ removal	5.5	10.3	8.3	8.0	7.7	7.1

Table 9: Word error rates for the different LM rescoring steps across all testsets. Last line shows WERs after ‘.’ removal from the references and system outputs.

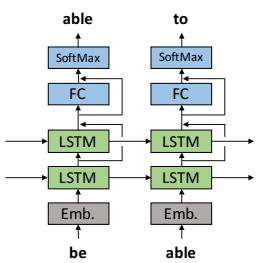


Figure 3: Word-LSTM

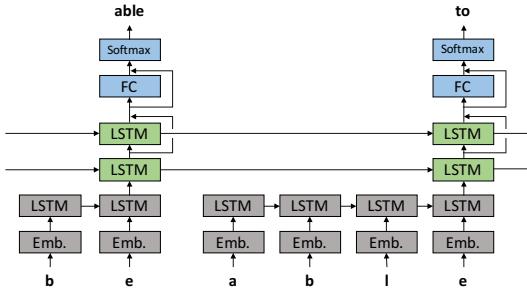


Figure 4: Char-LSTM

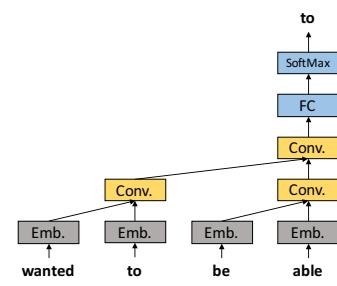


Figure 5: Word-DCC

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, two things were instrumental in reaching this level of performance. The first one is a steady improvement in bidirectional LSTM modeling, chief among them being a simple feature fusion experiment. The second one is the replacement of VGG nets with residual nets which are a more effective architecture on the ImageNet classification task. When combined together, these recurrent and convolutional nets show good complementarity and enhanced accuracy on a variety of testsets. 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.

The second and perhaps more important point made in this paper is that, unlike what was claimed in [1], we do not believe that human parity has been reached on this task. The reasons why we came to the opposite conclusion are twofold. 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 ASR performance is the average WER across all testsets which is around 8% for our system. The second and more direct argument is that the human WER of expert transcribers that were asked to do a high-quality job is simply 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] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswarah, “Boosted MMI for model and feature-space discriminative training,” in *Proc. ICASSP*, 2008, pp. 4057–4060.
- [10] 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.
- [11] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.
- [12] 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.
- [13] Y. Shinohara, “Adversarial multi-task learning of deep neural networks for robust speech recognition,” *Interspeech 2016*, pp. 2369–2372, 2016.
- [14] Y. Miao, M. Gowayyed, and F. Metze, “Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding,” *arXiv preprint arXiv:1507.08240*, 2015.
- [15] H. Sak, A. Senior, K. Rao, O. Irsay, A. Graves, F. Beaufays, and J. Schalkwyk, “Learning acoustic frame labeling for speech recognition with recurrent neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4280–4284.

- [16] H. Soltau, H. Liao, and H. Sak, “Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition,” *arXiv preprint arXiv:1610.09975*, 2016.
- [17] 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.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR arXiv:1512.03385*, 2015.
- [19] Y. Zhang, W. Chan, and N. Jaitly, “Very deep convolutional networks for end-to-end speech recognition,” *Proc. ICASSP*, 2017.
- [20] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.
- [21] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” *arXiv preprint arXiv:1603.05027*, 2016.
- [22] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, “Very deep multilingual convolutional neural networks for lvcsr,” *Proc. ICASSP*, 2016.
- [23] T. Sercu and V. Goel, “Advances in very deep convolutional neural networks for lvcsr,” *arXiv*, 2016.
- [24] ———, “Dense prediction on sequences with time-dilated convolutions for speech recognition,” *NIPS End-to-end Learning for Speech and Audio Processing Workshop*, 2016.
- [25] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” *arXiv preprint arXiv:1409.2329*, 2014.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] D. Kingma and J. Ba, “ADAM: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [30] P. Ghahremani and J. Droppo, “Self-stabilized deep neural network,” in *Proc. ICASSP*, 2016, pp. 6645–6649.
- [31] 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.