ACHIEVING HUMAN PARITY IN CONVERSATIONAL SPEECH RECOGNITION

W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu and G. Zweig

Microsoft Research
Technical Report MSR-TR-2016-71

ABSTRACT

Conversational speech recognition has served as a flagship speech recognition task since the release of the DARPA Switchboard corpus in the 1990s. In this paper, we measure the human error rate on the widely used NIST 2000 test set, and find that our latest automated system has reached human parity. The error rate of professional transcriptionists is 5.9% for the Switchboard portion of the data, in which newly acquainted pairs of people discuss an assigned topic, and 11.3% for the CallHome portion where friends and family members have open-ended conversations. In both cases, our automated system establishes a new state-of-the-art, and edges past the human benchmark. This marks the first time that human parity has been reported for conversational speech. The key to our system's performance is the systematic use of convolutional and LSTM neural networks, combined with a novel spatial smoothing method and lattice-free MMI acoustic training.

Index Terms— Conversational speech recognition, convolutional neural networks, recurrent neural networks, VGG, ResNet, LACE, BLSTM, spatial smoothing.

1. INTRODUCTION

Recent years have seen human performance levels reached or surpassed in tasks ranging from the games of chess and Go [1, 2] to simple speech recognition tasks like carefully read newspaper speech [3] and rigidly constrained small-vocabulary tasks in noise [4, 5]. In the area of speech recognition, much of the pioneering early work was driven by a series of carefully designed tasks with DARPA-funded datasets publicly released by the LDC and NIST [6]: first simple ones like the "resource management" task [7] with a small vocabulary and carefully controlled grammar; then read speech recognition in the Wall Street Journal task [8]; then Broadcast News [9]; each progressively more difficult for automatic systems. One of last big initiatives in this area was in conversational telephone speech (CTS), which is especially difficult due to the spontaneous (neither read nor planned) nature of the speech, its informality, and the self-corrections, hesitations and other disfluencies that are pervasive. The Switchboard [10] and later Fisher [11] data collections of the 1990s and early 2000s provide what is to date the largest and best studied of the conversational corpora. The history of work in this area includes key contributions by institutions such as IBM [12], BBN [13], SRI [14], AT&T [15], LIMSI [16], Cambridge University [17], Microsoft [18] and numerous others.

In the past, human performance on this task has been widely cited as being 4% [19]. However, the error rate estimate in [19] is attributed to a "personal communication," and the actual source of this number is ephemeral. To better understand human performance, we have used professional transcribers to transcribe the actual test sets that we are working with, specifically the CallHome and Switchboard portions of the NIST eval 2000 test set. We find that the human error rates on these two parts are different almost by a factor of two, so a single number is inappropriate to cite. The error rate on Switchboard is about 5.9%, and for CallHome 11.3%. We improve on our recently reported conversational speech recognition system [20] by about 0.4%, and now exceed human performance by a small margin.

Our progress is a result of the careful engineering and optimization of convolutional and recurrent neural networks. While the basic structures have been well known for a long period [21, 22, 23, 24, 25, 26, 27], it is only recently that they have emerged as the best models for speech recognition. Surprisingly, this is the case for both acoustic modeling [28, 29, 30, 31, 32, 33] and language modeling [34, 35, 36]. In comparison to the standard feed-forward MLPs or DNNs that first demonstrated breakthrough performance on conversational speech recognition [18], these acoustic models have the ability to model a large amount of acoustic context with temporal invariance, and in the case of convolutional models, with frequency invariance as well. In language modeling, recurrent models appear to improve over classical N-gram models through the use of an unbounded word history, as well as the generalization ability of continuous word representations [37]. In the meantime, ensemble learning has become commonly used in several neural models [38, 39, 35], to improve robustness by reducing bias and variance.

This paper is an expanded version of [20], with the following additional features:

1. A comprehensive assessment of human performance on

the NIST eval 2000 test set

2. The description of a novel spatial regularization method which significantly boosts our LSTM acoustic model performance

3. The use of LSTM rather than RNN-LMs, and the use of a letter-trigram input representation

4. Expanded coverage of the Microsoft Cognitive Toolkit (CNTK) used to build our models

5. An analysis of human versus machine errors, which indicates substantial equivalence, with the exception of the word classes of backchannel acknowledgements (e.g. "uh-huh") and hesitations (e.g. "um").

The remainder of this paper describes our system in detail. Section 2 describes our measurement of human performance. Section 3 describes the CNN and LSTM models. Section 4 describes our implementation of i-vector adaptation. Section 5 presents out lattice-free MMI training process. Language model rescoring is a significant part of our system, and described in Section 6. We describe the CNTK toolkit that forms the basis of our neural network models in Section 7. Experimental results are presented in Section 8, followed by an error analysis in section 9, a review of relevant prior work in 10 and concluding remarks.

2. HUMAN PERFORMANCE

To measure human performance, we leveraged an existing pipeline in which Microsoft data is transcribed on a weekly basis. This pipeline uses a large commercial vendor to perform two-pass transcription. In the first pass, a transcriber works from scratch to transcribe the data. In the second pass, a second listener monitors the data to do error correction. Dozens of hours of test data are processed in each batch.

One week, we added the NIST 2000 CTS evaluation data to the work-list, without further comment. The intention was to measure the error rate of professional transcribers going about their normal everyday business. Aside from the standard two-pass checking in place, we did not do a complex multi-party transcription and adjudication process. The transcribers were given the same audio segments as were provided to the speech recognition system, which results in short sentences or sentence fragments from a single channel. This makes the task easier since the speakers are more clearly separated, and more difficult since the two sides of the conversation are not interleaved. Thus, it is the same condition as reported for our automated systems. The resulting numbers are 5.9% for the Switchboard portion, and 11.3% for the CallHome portion of the NIST 2000 test set, using the NIST scoring protocol. These numbers should be taken as an indication of the "error rate" of a trained professional working in industry-standard speech transcript production.

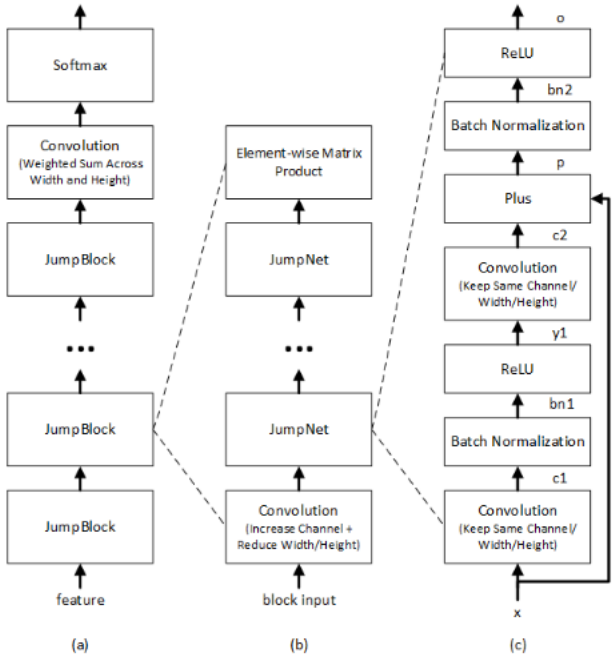

Fig. 1. LACE network architecture

Past work [40] reports inter-transcriber error rates for data taken from the later RT03 test set (which contains Switchboard and Fisher, but no CallHome data). Error rates of 4.1 to 4.5% are reported for extremely careful multiple transcriptions, and 9.6% for "quick transcriptions." While this is a different test set, the numbers are in line with our findings.

We note that the bulk of the Fisher training data, and the bulk of the data overall, was transcribed with the "quick transcription" guidelines. Thus, the current state of the art is actually far exceeding the noise level in its own training data.

Perhaps the most important point is the extreme variability between the two test subsets. The more informal CallHome data has almost double the human error rate of the Switchboard data. Interestingly, the same informality, multiple speakers per channel, and recording conditions that make CallHome hard for computers make it difficult for people as well. Notably, the performance of our artificial system aligns almost exactly with the performance of people on both sets.

3. CONVOLUTIONAL AND LSTM NEURAL NETWORKS

3.1. CNNs

We use three CNN variants. The first is the VGG architecture of [41]. Compared to the networks used previously in image recognition, this network uses small (3x3) filters, is deeper, and applies up to five convolutional layers before pooling. The second network is modeled on the ResNet architecture [42], which adds highway connections [43], i.e. a linear trans-

Table 1. Comparison of CNN architectures

VGG Net (85M Parameters)	Residual-Net (38M Parameters)	LACE (65M Parameters)
14 weight layers	49 weight layers	22 weight layers
40x41 input	40x41 input	40x61 input
3 – conv 3x3, 96	3 – [conv 1x1, 64 conv 3x3, 64 conv 1x1, 256]	5 – conv 3x3, 128
Max pool	4 – [conv 1x1, 128 conv 3x3, 128 conv 1x1, 512]	5 – conv 3x3, 256
4 – conv 3x3, 192	6 – [conv 1x1, 256 conv 3x3, 256 conv 1x1, 1024]	5 – conv 3x3, 512
Max pool	3 – [conv 1x1, 512 conv 3x3, 512 conv 1x1, 2048]	5 – conv 3x3, 1024
4 – conv 3x3, 384	Average pool	1 – conv 3x4, 1
Max pool	Softmax (9000)	Softmax (9000)
2 – FC – 4096		
Softmax (9000)		

form of each layer's input to the layer's output [43, 44]. The only difference is that we apply Batch Normalization before computing ReLU activations.

The last CNN variant is the LACE (layer-wise context expansion with attention) model [45]. LACE is a TDNN [23] variant in which each higher layer is a weighted sum of nonlinear transformations of a window of lower layer frames. In other words, each higher layer exploits broader context than lower layers. Lower layers focus on extracting simple local patterns while higher layers extract complex patterns that cover broader contexts. Since not all frames in a window carry the same importance, an attention mask is applied. The LACE model differs from the earlier TDNN models e.g. [23, 46] in the use of a learned attention mask and ResNet like linear pass-through. As illustrated in detail in Figure 1, the model is composed of four blocks, each with the same architecture. Each block starts with a convolution layer with stride 2 which sub-samples the input and increases the number of channels. This layer is followed by four RELU-convolution layers with jump links similar to those used in ResNet. Table 1 compares the layer structure and parameters of the three CNN architectures.

We also trained a fused model by combining a ResNet model and a VGG model at the senone posterior level. Both base models are independently trained, and then the score fusion weight is optimized on development data. The fused system is our best single system.

Table 2. Accuracy improvements from spatial smoothing. The model is a six layer BLSTM, using i-vectors and 40 dimensional filterbank features, and a dimension of 512 in each direction of each layer.

Senones	Eval 2000 SWB WER (%)	
	Baseline	Spatial Smoothing
9000	9.9	9.3
27000	10.6	9.2

3.2. LSTMs

While our best performing models are convolutional, the use of long short-term memory networks (LSTMs) is a close second. We use a bidirectional architecture [47] without frame-skipping [29]. The core model structure is the LSTM defined in [28]. We found that using networks with more than six layers did not improve the word error rate on the development set, and chose 512 hidden units, per direction, per layer, as that provided a reasonable trade-off between training time and final model accuracy.

3.3. Spatial Smoothing

Inspired by the human auditory cortex, where neighboring neurons tend to simultaneously activate, we employ a spatial smoothing technique to improve the accuracy of our LSTM models.

The smoothing is implemented as a regularization term on the activations between layers of the acoustic model. First, each vector of activations is re-interpreted as a 2-dimensional image. For example, if there are 512 neurons, they are interpreted as the pixels of a 16 by 32 image. Second, this image is high-pass filtered. The filter is implemented as a circular convolution with a 3 by 3 kernel. The center tap of the kernel has a value of 1, and the remaining eight having a value of $-1/8$. Third, the energy of this high-pass filtered image is computed and added to the training objective function. We have found empirically that a suitable scale for this energy is 0.1 with respect to the existing cross entropy objective function.

The overall effect of this process is to make the training algorithm prefer models that have correlated neurons, and to improve the word error rate of the acoustic model. Table 2 shows the benefit in error rate for some of our early systems.

4. SPEAKER ADAPTIVE MODELING

Speaker adaptive modeling in our system is based on conditioning the network on an i-vector [48] characterization of each speaker [49, 50]. A 100-dimensional i-vector is generated for each conversation side. For the LSTM system, the conversation-side i-vector v_s is appended to each frame of input. For convolutional networks, this approach is inappropriate because we do not expect to see spatially contiguous

Table 3. Performance improvements from i-vector and LFMMI training on the eval 2000 set.

Configuration	WER (%)					
	ReLU-DNN		BLSTM		LACE	
	CH	SWB	CH	SWB	CH	SWB
Baseline	21.9	13.4	17.3	10.3	16.9	10.4
i-vector	20.1	11.5	17.6	9.9	16.4	9.3
i-vector+LFMMI	17.9	10.2	16.3	8.9	15.2	8.5

patterns in the input. Instead, for the CNNs, we add a learnable weight matrix W^l to each layer, and add $W^l v_s$ to the activation of the layer before the nonlinearity. Thus, in the CNN, the i-vector essentially serves as an speaker-dependent bias to each layer. Note that the i-vectors are estimated using MFCC features; by using them subsequently in systems based on log-filterbank features, we may benefit from a form of feature combination.

Performance improvements from i-vectors are shown in Table 3. The full experimental setup is described in Section 8.

5. LATTICE-FREE SEQUENCE TRAINING

After standard cross-entropy training, we optimize the model parameters using the maximum mutual information (MMI) objective function. Denoting a word sequence by w and its corresponding acoustic realization by a, the training criterion is

$$\sum_{w,a \in \text{data}} \log \frac{P(w)P(a|w)}{\sum'_w P(w')P(a|w')} \quad .$$

As noted in [51, 52], the necessary gradient for use in back-propagation is a simple function of the posterior probability of a particular acoustic model state at a given time, as computed by summing over all possible word sequences in an unconstrained manner. As first done in [12], and more recently in [53], this can be accomplished with a straightforward alpha-beta computation over the finite state acceptor representing the decoding search space. In [12], the search space is taken to be an acceptor representing the composition $HCLG$ for a unigram language model L on words. In [53], a language model on phonemes is used instead.

In our implementation, we use a mixed-history acoustic unit language model. In this model, the probability of transitioning into a new context-dependent phonetic state (senone) is conditioned on both the senone and phone history. We found this model to perform better than either purely word-based or phone-based models. Based on a set of initial experiments, we developed the following procedure:

1. Perform a forced alignment of the training data to select lexical variants and determine senone sequences.
2. Compress consecutive framewise occurrences of a single senone into a single occurrence.
3. Compute a variable-length N-gram language model from this data, where the history state consists of the previous phone and previous senones within the current phone.

To illustrate this, consider the sample senone sequence {*s_s2.1288, s_s3.1061, s_s4.1096*}, {*eh_s2.527, eh_s3.128, eh_s4.66*}, {*t_s2.729, t_s3.572, t_s4.748*}. We would compute, e.g.,
$P(t\_s2.729|s, eh\_s2.527, eh\_s3.128, eh\_s4.66)$ and then
$P(t\_s3.572|eh, t\_s2.729)$.

We construct the denominator graph from this language model, and HMM transition probabilities as determined by transition-counting in the senone sequences found in the training data. Our approach not only largely reduces the complexity of building up the language model but also provides very reliable training performance.

We have found it convenient to do the full computation, without pruning, in a series of matrix-vector operations on the GPU. The underlying acceptor is represented with a sparse matrix, and we maintain a dense likelihood vector for each time frame. The alpha and beta recursions are implemented with CUSPARSE level-2 routines: sparse-matrix, dense vector multiplies. Run time is about 100 times faster than real time. As in [53], we use cross-entropy regularization. In all the lattice-free MMI (LFMMI) experiments mentioned below we use a trigram language model. Most of the gain is usually obtained after processing 24 to 48 hours of data.

6. LM RESCORING AND SYSTEM COMBINATION

An initial decoding is done with a WFST decoder, using the architecture described in [54]. We use an N-gram language model trained and pruned with the SRILM toolkit [55]. The first-pass LM has approximately 15.9 million bigrams, trigrams, and 4grams, and a vocabulary of 30500 words. and gives a perplexity of 54 on RT-03 speech transcripts. The initial decoding produces a lattice with the pronunciation variants marked, from which 500-best lists are generated for rescoring purposes. Subsequent N-best rescoring uses an unpruned LM comprising 145 million N-grams. All N-gram LMs were estimated by a maximum entropy criterion as described in [56].

The N-best hypotheses are then rescored using a combination of the large N-gram LM and several neural net LMs. We have experimented with both RNN LMs and LSTM LMs, and describe the details in the following two sections.

6.1. RNN-LM setup

Our RNN-LMs are trained and evaluated using the CUED-RNNLM toolkit [57]. Our RNN-LM configuration has sev-

eral distinctive features, as described below.

1. We trained both standard, forward-predicting RNN-LMs and backward RNN-LMs that predict words in reverse temporal order. The log probabilities from both models are added.

2. As is customary, the RNN-LM probability estimates are interpolated at the word-level with corresponding N-gram LM probabilities (separately for the forward and backward models). In addition, we trained a second RNN-LM for each direction, obtained by starting with different random initial weights. The two RNN-LMs and the N-gram LM for each direction are interpolated with weights of (0.375, 0.375, 0.25).

3. In order to make use of LM training data that is not fully matched to the target conversational speech domain, we start RNN-LM training with the union of in-domain (here, CTS) and out-of-domain (e.g., Web) data. Upon convergence, the network undergoes a second training phase using the in-domain data only. Both training phases use in-domain validation data to regulate the learning rate schedule and termination. Because the size of the out-of-domain data is a multiple of the in-domain data, a standard training on a simple union of the data would not yield a well-matched model, and have poor perplexity in the target domain.

4. We found best results with an RNN-LM configuration that had a second, non-recurrent hidden layer. This produced lower perplexity and word error than the standard, single-hidden-layer RNN-LM architecture [34].[1] The overall network architecture thus had two hidden layers with 1000 units each, using ReLU nonlinearities. Training used noise-contrastive estimation (NCE) [58].

5. The RNN-LM output vocabulary consists of all words occurring more than once in the in-domain training set. While the RNN-LM estimates a probability for unknown words, we take a different approach in rescoring: The number of out-of-set words is recorded for each hypothesis and a penalty for them is empirically estimated, along with the weights for the LM score proper and the pronunciation probabilities, using the SRILM *nbest-optimize* tool.

6.2. LSTM-LM setup

After obtaining good results with RNN-LMs we also explored the LSTM recurrent network architecture for language modeling, inspired by recent work showing gains over RNN-LMs for conversational speech recognition [36]. In addition to applying the lessons learned from our RNN-LM experiments, we explored additional alternatives, as described below.

[1] However, adding more hidden layers produced no further gains.

Table 4. LSTM perplexities (PPL) as a function of hidden layers, trained on in-domain data only, computed on 2001 CTS eval transcripts.

Language model	PPL
letter trigram input with one layer (baseline)	63.2
+ two hidden layers	61.8
+ three hidden layers	59.1
+ four hidden layers	59.6
+ five hidden layers	60.2
+ six hidden layers	63.7

1. There are two types of input vectors our LSTM LMs take, word based one-hot vector input and letter trigram vector [59] input. Including both forward and backward models, we trained four different LSTM LMs in total.

2. For the word based input, we leveraged the approach from [60] to tie the input embedding and output embedding together.

3. Here we also used a two-phase training schedule to train the LSTM LMs. First we train the model on the combination of in-domain and out-domain data for four data passes without any learning rate adjustment. We then start from the resulting model and train on in-domain data until convergence.

4. Overall the letter trigram based models perform a little better than the word based language model. We tried applying dropout on both types of language models but didn't see an improvement.

5. Convergence was improved through a variation of self-stabilization [61], in which each output vector x of nonlinearities are scaled by $\frac{1}{4}\ln(1+e^{4\beta})$, where a β is a scalar that is learned for each output. This has a similar effect as the scale of the well-known batch normalization technique, but can be used in recurrent loops.

6. Table 4 shows the impact of number of layers on the final perplexities. Based on this, we proceeded with three hidden layers, with 1000 hidden units each. The perplexities of each LSTM-LM we used in the final combination (before interpolating with the N-gram model) can be found in Table 5.

For the final system, we interpolated two LSTM-LMs with an N-gram LM for the forward-direction LM, and similarly for the backward-direction LM. All LSTMs use three hidden layers and are trained on in-domain and web data.

Table 5. Perplexities (PPL) of the four LSTM LMs used in the final combination. PPL is computed on 2001 CTS eval transcripts. All the LSTM LMs are with three hidden layers.

Language model	PPL
RNN: 2 layers + word input (baseline)	59.8
LSTM: word input in forward direction	54.4
LSTM: word input in backward direction	53.4
LSTM: letter trigram input in forward direction	52.1
LSTM: letter trigram input in backward direction	52.0

Table 6. Performance of various versions of neural-net-based LM rescoring. Perplexities (PPL) are computed on 2001 CTS eval transcripts; word error rates (WER) on the NIST 2000 Switchboard test set.

Language model	PPL	WER
4-gram LM (baseline)	75.5	8.6
+ RNN-LM, CTS data only	63.2	7.6
+ Web data training	61.9	7.4
+ 2nd hidden layer	59.8	7.4
+ backward RNN-LM	-	7.1
+ 2-RNN-LM interpolation	57.4	6.9
+ LSTM-LM, CTS + Web data	51.4	6.9
+ 2-LSTM-LM interpolation	50.5	6.8
+ backward LSTM-LM	-	6.6

Unlike for the RNN-LMs, the two models being interpolated differ not just in their random initialization, but also in the input encoding (one uses a triletter encoding, the other a one-hot word encoding). The forward and backward LM log probability scores are combined additively.

6.3. Training data

The 4-gram language model for decoding was trained on the available CTS transcripts from the DARPA EARS program: Switchboard (3M words), BBN Switchboard-2 transcripts (850k), Fisher (21M), English CallHome (200k), and the University of Washington conversational Web corpus (191M). A separate N-gram model was trained from each source and interpolated with weights optimized on RT-03 transcripts. For the unpruned large rescoring 4-gram, an additional LM component was added, trained on 133M word of LDC Broadcast News texts. The N-gram LM configuration is modeled after that described in [50], except that maxent smoothing was used.

The RNN and LSTM LMs were trained on Switchboard and Fisher transcripts as in-domain data (20M words for gradient computation, 3M for validation). To this we added 62M words of UW Web data as out-of-domain data, for use in the two-phase training procedure described above.

6.4. RNN-LM and LSTM-LM performance

Table 6 gives perplexity and word error performance for various recurrent neural net LM setups, from simple to more complex. The acoustic model used was the ResNet CNN.

As can be seen, each of the measures described earlier adds incremental gains, which, however, add up to a substantial improvement overall. The total gain relative to a purely N-gram based system is a 20% relative error reduction with RNN-LMs, and 23% with LSTM-LMs. As shown later (see Table 7) the gains with different acoustic models are similar.

6.5. System Combination

The LM rescoring is carried out separately for each acoustic model. The rescored N-best lists from each subsystem are then aligned into a single confusion network [62] using the SRILM *nbest-rover* tool. However, the number of potential candidate systems is too large to allow an all-out combination, both for practical reasons and due to overfitting issues. Instead, we perform a greedy search, starting with the single best system, and successively adding additional systems, to find a small set of systems that are maximally complementary. The RT-02 Switchboard set was used for this search procedure. The relative weighting (for confusion-network mediated voting) of the different systems is optimized using an EM algorithm, using the same data, and smoothed hierarchically by interpolating each set of system weights with the preceding one in the search.

7. MICROSOFT COGNITIVE TOOLKIT (CNTK)

All neural networks in the final system were trained with the Microsoft Cognitive Toolkit, or CNTK [63, 64], on a Linux-based multi-GPU server farm. CNTK allows for flexible model definition, while at the same time scaling very efficiently across multiple GPUs *and* multiple servers. The resulting fast experimental turnaround using the full 2000h corpus was critical for our work.

7.1. Flexible, Terse Model Definition

In CNTK, a neural network (and the training criteria) are specified by its formula, using a custom functional language (BrainScript), or Python. A graph-based execution engine, which provides automatic differentiation, then trains the model's parameters through SGD. Leveraging a stock library of common layer types, networks can be specified very tersely. Samples can be found in [63].

7.2. Multi-Server Training using 1-bit SGD

Training the acoustic models in this paper on a single GPU would take many weeks or even months. CNTK made training times feasible by parallelizing the SGD training with our

Table 7. Word error rates (%) on the NIST 2000 CTS test set with different acoustic models. Unless otherwise noted, models are trained on the full 2000 hours of data and have 9k senones. Our automated system makes about a dozen fewer errors than people on the SWB set, not visible below due to rounding.

Model	N-gram LM		RNN-LM		LSTM-LM	
	CH	SWB	CH	SWB	CH	SWB
300h ResNet	19.2	10.0	17.7	8.2	17.0	7.7
ResNet GMM alignment	15.3	8.8	13.7	7.3	12.8	6.9
ResNet	14.8	8.6	13.2	6.9	12.5	6.6
VGG + ResNet	14.5	8.4	13.0	6.9	12.2	6.4
VGG	15.7	9.1	14.1	7.6	13.2	7.1
LACE	14.8	8.3	13.5	7.1	12.7	6.7
BLSTM	16.6	8.9	15.1	7.4	14.4	7.0
BLSTM 27k senones	16.2	8.7	14.6	7.5	13.6	7.0
BLSTM 27k, spatial smoothing	14.9	8.3	13.7	7.0	13.0	6.7
Final ASR System	13.3	7.4	12.0	6.2	**11.1**	**5.9**
Human Performance	-	-	-	-	11.3	5.9

Table 8. Comparative error rates from the literature.

Model	N-gram LM		NN LM	
	CH	SWB	CH	SWB
Saon et al. [50] LSTM	15.1	9.0	-	-
Povey et al. [53] LSTM	15.3	8.5	-	-
Saon et al. [50] Combination	13.7	7.6	12.2	6.6

1-bit SGD parallelization technique [65]. This data-parallel method distributes minibatches over multiple worker nodes, and then aggregates the sub-gradients. While the necessary communication time would otherwise be prohibitive, the 1-bit SGD method eliminates the bottleneck by two techniques: *1-bit quantization of gradients* and *automatic minibatch-size scaling*.

In [65], we showed that gradient values can be quantized to just a single bit, if one carries over the quantization error from one minibatch to the next. Each time a sub-gradient is quantized, the quantization error is computed and remembered, and then added to the next minibatch's sub-gradient. This reduces the required bandwidth 32-fold with minimal loss in accuracy. Secondly, automatic minibatch-size scaling progressively decreases the frequency of model updates. At regular intervals (e.g. every 72h of training data), the trainer tries larger minibatch sizes on a small subset of data and picks the largest that maintains training loss.

These two techniques allow for excellent multi-GPU/server scalability, and reduced the acoustic-model training times on 2000h from months to between 1 and 3 weeks, making this work feasible.

8. EXPERIMENTAL SETUP AND RESULTS

8.1. Speech corpora

We train with the commonly used English CTS (Switchboard and Fisher) corpora. Evaluation is carried out on the NIST 2000 CTS test set, which comprises both Switchboard (SWB) and CallHome (CH) subsets. The Switchboard-1 portion of the NIST 2002 CTS test set was used for tuning and development. The acoustic training data is comprised by LDC corpora 97S62, 2004S13, 2005S13, 2004S11 and 2004S09; see [12] for a full description.

8.2. Acoustic Model Details

Forty-dimensional log-filterbank features were extracted every 10 milliseconds, using a 25-millisecond analysis window. The CNN models used window sizes as indicated in Figure 1, and the LSTMs processed one frame of input at a time. The bulk of our models use three state left-to-right triphone models with 9000 tied states. Additionally, we have trained several models with 27k tied states. The phonetic inventory includes special models for noise, vocalized-noise, laughter and silence. We use a 30k-vocabulary derived from the most common words in the Switchboard and Fisher corpora. The decoder uses a statically compiled unigram graph, and dynamically applies the language model score. The unigram graph has about 300k states and 500k arcs.

Table 3 shows the result of i-vector adaptation and LFMMI training on several of our early systems. We achieve a 5–8% relative improvement from i-vectors, including on CNN systems. The last row of Table 3 shows the effect of LFMMI training on the different models. We see a consistent 7–10% further relative reduction in error rate for all models. Considering the great increase in procedural simplicity

Table 9. Most common substitutions for ASR system and humans. The number of times each error occurs is followed by the word in the reference, and what appears in the hypothesis instead.

CH		SWB	
ASR	**Human**	**ASR**	**Human**
45: (%hesitation) / %bcack	12: a / the	29: (%hesitation) / %bcack	12: (%hesitation) / hmm
12: was / is	10: (%hesitation) / a	9: (%hesitation) / oh	10: (%hesitation) / oh
9: (%hesitation) / a	10: was / is	9: was / is	9: was / is
8: (%hesitation) / oh	7: (%hesitation) / hmm	8: and / in	8: (%hesitation) / a
8: a / the	7: bentsy / bensi	6: (%hesitation) / i	5: in / and
7: and / in	7: is / was	6: in / and	4: (%hesitation) / %bcack
7: it / that	6: could / can	5: (%hesitation) / a	4: and / in
6: in / and	6: well / oh	5: (%hesitation) / yeah	4: is / was
5: a / to	5: (%hesitation) / %bcack	5: a / the	4: that / it
5: aw / oh	5: (%hesitation) / oh	5: jeez / jeeze	4: the / a

Table 10. Most common deletions for ASR system and humans.

CH		SWB	
ASR	**Human**	**ASR**	**Human**
44: i	73: i	31: it	34: i
33: it	59: and	26: i	30: and
29: a	48: it	19: a	29: it
29: and	47: is	17: that	22: a
25: is	45: the	15: you	22: that
19: he	41: %bcack	13: and	22: you
18: are	37: a	12: have	17: the
17: oh	33: you	12: oh	17: to
17: that	31: oh	11: are	15: oh
17: the	30: that	11: is	15: yeah

Table 11. Most common insertions for ASR system and humans.

CH		SWB	
ASR	**Human**	**ASR**	**Human**
15: a	10: i	19: i	12: i
15: is	9: and	9: and	11: and
11: i	8: a	7: of	9: you
11: the	8: that	6: do	8: is
11: you	8: the	6: is	6: they
9: it	7: have	5: but	5: do
7: oh	5: you	5: yeah	5: have
6: and	4: are	4: air	5: it
6: in	4: is	4: in	5: yeah
6: know	4: they	4: you	4: a

of LFMMI over the previous practice of writing lattices and post-processing them, we consider LFMMI to be a significant advance in technology.

8.3. Comparative System Performance

The performance of our component models is shown in Table 7, along with the final system combination result, and human benchmarks. Key benchmarks from the literature are shown in Table 8. Out of the 15 models built, only models selected in the final system combination are shown.

9. ERROR ANALYSIS

In this section, we compare the errors made by our artificial recognizer with those made by human transcribers. We find that the artificial errors are substantially the same as human ones, with one large exception: confusions between *backchannel* words and *hesitations*. The distinction is that backchannel words like "uh-huh" are an acknowledgment of the speaker, also signaling that the speaker should keep talking, while hesitations like "uh" are used to indicate that the current speaker has more to say and wants to keep his or her turn.[2] As turn-management devices, these two classes of words therefore have exactly opposite functions.

Table 9 shows the ten most common substitutions for both humans and the artificial system. Tables 10 and 11 do the same for deletions and insertions. Focusing on the substitutions, we see that by far the most common error in the ASR system is the confusion of a hesitation in the reference for a backchannel in the hypothesis. People do not seem to have this problem. We speculate that this is due to the nature of the Fisher training corpus, where the "quick transcription" guidelines were predominately used [40]. We find that there is inconsistent treatment of backchannel and hesitation in the resulting data; the relatively poor performance of the automatic system here might simply be due to confusions in the training data annotations. For perspective, there are over twenty-one

[2]The NIST scoring protocol treats hesitation words as optional, and we therefore delete them from our recognizer output prior to scoring. This explains why we see many substitutions of backchannels for hesitations, but not vice-versa.

Table 12. Overall substitution, deletion and insertion rates.

	CH		SWB	
	ASR	**Human**	**ASR**	**Human**
sub	6.5	4.1	3.3	2.6
del	3.3	6.5	1.8	2.7
ins	1.4	0.7	0.7	0.7
all	11.1	11.3	5.9	5.9

thousand words in each test set. Thus the errors due to hesitation/backchannel substitutions account for an error rate of only about 0.2% absolute.

The most frequent substitution for people on the Switchboard corpus was mistaking a hesitation in the reference for the word "hmm." The scoring guidelines treat "hmm" as a word distinct from backchannels and hesitations, so this is not a scoring mistake. Examination of the contexts in which the error is made show that it is most often intended to acknowledge the other speaker, i.e. as a backchannel. For both people and our automated system, the insertion and deletion patterns are similar: short function words are by far the most frequent errors. In particular, the single most common error made by the transcriptionists was to omit the word "I." While we believe further improvement in function and content words is possible, the significance of the remaining backchannel/hesitation confusions is unclear.

Table 12 shows the overall error rates broken down by substitutions, insertions and deletions. We see that the human transcribers have a somewhat lower substitution rate, and a higher deletion rate. The relatively higher deletion rate might reflect a human bias to avoid outputting uncertain information, or the productivity demands on a professional transcriber. In all cases, the number of insertions is relatively small.

10. RELATION TO PRIOR WORK

Compared to earlier applications of CNNs to speech recognition [66, 67], our networks are much deeper, and use linear bypass connections across convolutional layers. They are similar in spirit to those studied more recently by [31, 30, 50, 32, 33]. We improve on these architectures with the LACE model [45], which iteratively expands the effective window size, layer-by-layer, and adds an attention mask to differentially weight distant context.

Our spatial regularization technique is similar in spirit to stimulated deep neural networks [68]. Whereas stimulated networks use a supervision signal to encourage locality of activations in the model, our technique is automatic.

Our use of lattice-free MMI is distinctive, and extends previous work [12, 53] by proposing the use of a mixed triphone/phoneme history in the language model.

On the language modeling side, we achieve a performance boost by combining multiple LSTM-LMs in both forward and backward directions, and by using a two-phase training regimen to get best results from out-of-domain data. For our best CNN system, LSTM-LM rescoring yields a relative word error reduction of 23%, and a 20% relative gain for the combined recognition system, considerably larger than previously reported for conversational speech recognition [36].

11. CONCLUSIONS

We have measured the human error rate on NIST's 2000 conversational telephone speech recognition task. We find that there is a great deal of variability between the Switchboard and CallHome subsets, with 5.9% and 11.3% error rates respectively. For the first time, we report automatic recognition performance on par with human performance on this task. Our system's performance can be attributed to the systematic use of LSTMs for both acoustic and language modeling, as well as CNNs in the acoustic model, and extensive combination of complementary models.

Acknowledgments

We thank Arul Menezes for access to the Microsoft transcription pipeline; Chris Basoglu, Amit Agarwal and Marko Radmilac for their invaluable assistance with CNTK; Jinyu Li and Partha Parthasarathy for many helpful conversations. We also thank X. Chen from Cambridge University for valuable assistance with the CUED-RNNLM toolkit, and ICSI for compute and data resources.

12. REFERENCES

[1] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep Blue", *Artificial intelligence*, vol. 134, pp. 57–83, 2002.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, pp. 484–489, 2016.

[3] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al., "Deep Speech 2: End-to-end speech recognition in English and Mandarin", arXiv preprint arXiv:1512.02595, 2015.

[4] T. T. Kristjansson, J. R. Hershey, P. A. Olsen, S. J. Rennie, and R. A. Gopinath, "Super-human multi-talker speech recognition: the IBM 2006 Speech Separation Challenge system", *in Proc. Interspeech*, vol. 12, p. 155, 2006.

[5] C. Weng, D. Yu, M. L. Seltzer, and J. Droppo, "Single-channel mixed speech recognition using deep neural networks", *in Proc. IEEE ICASSP*, pp. 5632–5636. IEEE, 2014.

[6] D. S. Pallett, "A look at NIST's benchmark ASR tests: past, present, and future", *in IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 483–488. IEEE, 2003.

[7] P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett, "The DARPA 1000-word resource management database for continuous speech recognition", *in Proc. IEEE ICASSP*, pp. 651–654. IEEE, 1988.

[8] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus", *in Proceedings of the workshop on Speech and Natural Language*, pp. 357–362. Association for Computational Linguistics, 1992.

[9] D. Graff, Z. Wu, R. MacIntyre, and M. Liberman, "The 1996 broadcast news speech and language-model corpus", *in Proceedings of the DARPA Workshop on Spoken Language technology*, pp. 11–14, 1997.

[10] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development", *in Proc. IEEE ICASSP*, vol. 1, pp. 517–520. IEEE, 1992.

[11] C. Cieri, D. Miller, and K. Walker, "The Fisher corpus: a resource for the next generations of speech-to-text", *in LREC*, vol. 4, pp. 69–71, 2004.

[12] S. F. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1596–1608, 2006.

[13] S. Matsoukas, J.-L. Gauvain, G. Adda, T. Colthurst, C.-L. Kao, O. Kimball, L. Lamel, F. Lefevre, J. Z. Ma, J. Makhoul, et al., "Advances in transcription of broadcast news and conversational telephone speech within the combined ears bbn/limsi system", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1541–1556, 2006.

[14] A. Stolcke, B. Chen, H. Franco, V. R. R. Gadde, M. Graciarena, M.-Y. Hwang, K. Kirchhoff, A. Mandal, N. Morgan, X. Lei, et al., "Recent innovations in speech-to-text transcription at SRI-ICSI-UW", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1729–1744, 2006.

[15] A. Ljolje, "The AT&T 2001 LVCSR system", NIST LVCSR Workshop, 2001.

[16] J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, L. Chen, and F. Lefevre, "Conversational telephone speech recognition", *in Proc. IEEE ICASSP*, vol. 1, pp. I–212. IEEE, 2003.

[17] G. Evermann, H. Y. Chan, M. J. F. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, and P. C. Woodland, "Development of the 2003 cu-htk conversational telephone speech transcription system", *in Proc. IEEE ICASSP*, vol. 1, pp. I–249. IEEE, 2004.

[18] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks", *in Proc. Interspeech*, pp. 437–440, 2011.

[19] R. P. Lippmann, "Speech recognition by machines and humans", *Speech Communication*, vol. 22, pp. 1–15, 1997.

[20] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The Microsoft 2016 conversational speech recognition system", submitted to ICASSP, 2017, preprint at https://arxiv.org/abs/1609.03528.

[21] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks", *Physical Review Letters*, vol. 59, pp. 2229, 1987.

[22] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, vol. 1, pp. 270–280, 1989.

[23] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks", *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, pp. 328–339, 1989.

[24] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series", *The handbook of brain theory and neural networks*, vol. 3361, pp. 1995, 1995.

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural computation*, vol. 1, pp. 541–551, 1989.

[26] T. Robinson and F. Fallside, "A recurrent error propagation network speech recognition system", *Computer Speech & Language*, vol. 5, pp. 259–274, 1991.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

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

[29] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition", arXiv preprint arXiv:1507.06947, 2015.

[30] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny, "The IBM 2015 English conversational telephone speech recognition system", arXiv preprint arXiv:1505.05899, 2015.

[31] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for lvcsr", *in Proc. IEEE ICASSP*, pp. 4955–4959. IEEE, 2016.

[32] M. Bi, Y. Qian, and K. Yu, "Very deep convolutional neural networks for lvcsr", *in Proc. Interspeech*, 2015.

[33] Y. Qian, M. Bi, T. Tan, and K. Yu, "Very deep convolutional neural networks for noise robust speech recognition", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. PP, pp. 1–1, 2016.

[34] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model", *in Proc. Interspeech*, vol. 2, p. 3, 2010.

[35] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model", *in IEEE Speech and Language Technology Workshop*, pp. 234–239, 2012.

[36] I. Medennikov, A. Prudnikov, and A. Zatvornitskiy, "Improving English conversational telephone speech recognition", *in Proc. Interspeech*, pp. 2–6, 2016.

[37] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations", *in HLT-NAACL*, vol. 13, pp. 746–751, 2013.

[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", *in Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[39] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al., "Deep speech: Scaling up end-to-end speech recognition", arXiv preprint arXiv:1412.5567, 2014.

[40] M. L. Glenn, S. Strassel, H. Lee, K. Maeda, R. Zakhary, and X. Li, "Transcription methods for consistency, volume and efficiency", *in LREC*, 2010.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556, 2014.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", arXiv preprint arXiv:1512.03385, 2015.

[43] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks", *CoRR*, vol. abs/1505.00387, 2015.

[44] P. Ghahremani, J. Droppo, and M. L. Seltzer, "Linearly augmented deep neural network", *in Proc. IEEE ICASSP*, pp. 5085–5089. IEEE, 2016.

[45] D. Yu, W. Xiong, J. Droppo, A. Stolcke, G. Ye, J. Li, and G. Zweig, "Deep convolutional neural networks with layer-wise context expansion and attention", *in Proc. Interspeech*, 2016.

[46] A. Waibel, H. Sawai, and K. Shikano, "Consonant recognition by modular construction of large phonemic time-delay neural networks", *in Proc. IEEE ICASSP*, pp. 112–115. IEEE, 1989.

[47] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures", *Neural Networks*, vol. 18, pp. 602–610, 2005.

[48] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, pp. 788–798, 2011.

[49] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors", *in IEEE Speech Recognition and Understanding Workshop*, pp. 55–59, 2013.

[50] G. Saon, T. Sercu, S. J. Rennie, and H. J. Kuo, "The IBM 2016 English conversational telephone speech recognition system", *in Proc. Interspeech*, pp. 7–11, Sep. 2016.

[51] G. Wang and K. Sim, "Sequential classification criteria for NNs in automatic speech recognition", *in Proc. Interspeech*, 2011.

[52] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks", *in Proc. Interspeech*, pp. 2345–2349, 2013.

[53] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI", *in Proc. Interspeech*, pp. 2751–2755, 2016.

[54] C. Mendis, J. Droppo, S. Maleki, M. Musuvathi, T. Mytkowicz, and G. Zweig, "Parallelizing WFST speech decoders", *in Proc. IEEE ICASSP*, pp. 5325–5329. IEEE, 2016.

[55] A. Stolcke, "SRILM—an extensible language modeling toolkit", *in Proc. Interspeech*, vol. 2002, p. 2002, 2002.

[56] T. Alumäe and M. Kurimo, "Efficient estimation of maximum entropy language models with N-gram features: An SRILM extension", *in Proc. Interspeech*, pp. 1820–1823, 2012.

[57] X. Chen, X. Liu, Y. Qian, M. J. F. Gales, and P. C. Woodland, "CUED-RNNLM: An open-source toolkit for efficient training and evaluation of recurrent neural network language models", *in Proc. IEEE ICASSP*, pp. 6000–6004. IEEE, 2016.

[58] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models", *AISTATS*, vol. 1, pp. 6, 2010.

[59] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data", *in Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 2333–2338. ACM, 2013.

[60] O. Press and L. Wolf, "Using the output embedding to improve language models", arXiv preprint arXiv:1608.05859, 2016.

[61] P. Ghahremani and J. Droppo, "Self-stabilized deep neural network", *in Proc. IEEE ICASSP*. IEEE, 2016.

[62] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauché, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng, "The SRI March 2000 Hub-5 conversational speech transcription system", *in Proceedings NIST Speech Transcription Workshop*, College Park, MD, May 2000.

[63] Microsoft Research, "The Microsoft Cognition Toolkit (CNTK)", https://cntk.ai.

[64] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, et al., "An introduction to computational networks and the computational network toolkit", Technical report, Technical report, Tech. Rep. MSR, Microsoft Research, 2014, 2014. research. microsoft. com/apps/pubs, 2014.

[65] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs", *in Proc. Interspeech*, pp. 1058–1062, 2014.

[66] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr", *in Proc. IEEE ICASSP*, pp. 8614–8618. IEEE, 2013.

[67] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition", *in Proc. IEEE ICASSP*, pp. 4277–4280. IEEE, 2012.

[68] C. Wu, P. Karanasou, M. J. Gales, and K. C. Sim, "Stimulated deep neural network for speech recognition", *in Proc. Interspeech*, pp. 400–404, 2016.