Wav2Letter: an End-to-End ConvNet-based Speech Recognition System

Ronan Collobert
Facebook AI Research, Menlo Park
locronan@fb.com

Christian Puhrsch
Facebook AI Research, Menlo Park
cpuhrsch@fb.com

Gabriel Synnaeve
Facebook AI Research, New York
gab@fb.com

Abstract

This paper presents a simple end-to-end model for speech recognition, combining a convolutional network based acoustic model and a graph decoding. It is trained to output letters, with transcribed speech, without the need for force alignment of phonemes. We introduce an automatic segmentation criterion for training from sequence annotation without alignment that is on par with CTC [6] while being simpler. We show competitive results in word error rate on the Librispeech corpus [18] with MFCC features, and promising results from raw waveform.

1 Introduction

We present an end-to-end system to speech recognition, going from the speech signal (e.g. Mel-Frequency Cepstral Coefficients (MFCC), power spectrum, or raw waveform) to the transcription. The acoustic model is trained using letters (graphemes) directly, which take out the need for an intermediate (human or automatic) phonetic transcription. Indeed, the classical pipeline to build state of the art systems for speech recognition consists in first training an HMM/GMM model to force align the units on which the final acoustic model operates (most often context-dependent phone states). This approach takes its roots in HMM/GMM training [27]. The improvements brought by deep neural networks (DNNs) [14, 10] and convolutional neural networks (CNNs) [24, 25] for acoustic modeling only extend this training pipeline.

The current state of the art on Librispeech (the dataset that we used for our evaluations) uses this approach too [18, 20], with an additional step of speaker adaptation [22, 19]. Recently, [23] proposed GMM-free training, but the approach still requires to generate a force alignment. An approach that cut ties with the HMM/GMM pipeline (and with force alignment) was to train with a recurrent neural network (RNN) [7] for phoneme transcription. There are now competitive end-to-end approaches of acoustic models toppled with RNNs layers as in [8, 13, 21, 1], trained with a sequence criterion [6]. However these models are computationally expensive, and thus take a long time to train.

Compared to classical approaches that need phonetic annotation (often derived from a phonetic dictionary, rules, and generative training), we propose to train the model end-to-end, using graphemes directly. Compared to sequence criterion based approaches that train directly from speech signal to graphemes [13], we propose a simple(r) architecture (23 millions of parameters for our best model, vs. 100 millions of parameters in [1]) based on convolutional networks for the acoustic model, toppled with a graph transformer network [4], trained with a simpler sequence criterion. Our word-error-rate on clean speech is slightly better than [8], and slightly worse than [1], in particular factoring that they train on 12,000 hours while we only train on the 960h available in LibriSpeech's train set. Finally, some of our models are also trained on the raw waveform, as in [15, 16]. The rest of the paper is

structured as follows: the next section presents the convolutional networks used for acoustic modeling, along with the automatic segmentation criterion. The following section shows experimental results comparing different features, the criterion, and our current best word error rates on LibriSpeech.

2 Architecture

Our speech recognition system is a standard convolutional neural network [12] fed with various different features, trained through an alternative to the Connectionist Temporal Classification (CTC) [6], and coupled with a simple beam search decoder. In the following sub-sections, we detail each of these components.

2.1 Features

We consider three types of input features for our model: MFCCs, power-spectrum, and raw wave. MFCCs are carefully designed speech-specific features, often found in classical HMM/GMM speech systems [27] because of their dimensionality compression (13 coefficients are often enough to span speech frequencies). Power-spectrum features are found in most recent deep learning acoustic modeling features [1]. Raw wave has been somewhat explored in few recent work [15, 16]. ConvNets have the advantage to be flexible enough to be used with either of these input feature types. Our acoustic models output letter scores (one score per letter, given a dictionary $\mathcal{L}$).

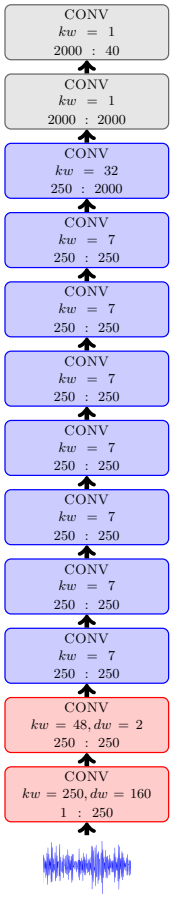


Figure 1: Our neural network architecture for raw wave. First two layers are convolutions with strides. Last two layers are convolutions with $kw = 1$, which are equivalent to fully connected layers. Power spectrum and MFCC based networks do not have the first layer.

2.2 ConvNet Acoustic Model

The acoustic models we considered in this paper are all based on standard 1D convolutional neural networks (ConvNets). ConvNets interleave convolution operations with pointwise non-linearity operations. Often ConvNets also embark pooling layers: these type of layers allow the network to "see" a larger context, without increasing the number of parameters, by locally aggregating the previous convolution operation output. Instead, our networks leverage striding convolutions. Given $(x_t)_{t=1...T_x}$ an input sequence with T_x frames of d_x dimensional vectors, a convolution with kernel width kw, stride dw and d_y frame size output computes the following:

$$y_t^i = b_i + \sum_{j=1}^{d_x} \sum_{k=1}^{kw} w_{i,j,k}\, x^j_{dw\times(t-1)+k} \qquad \forall 1 \leq i \leq d_y, \qquad (1)$$

where $b \in \mathbb{R}^{d_y}$ and $w \in \mathbb{R}^{d_y \times d_x \times kw}$ are the parameters of the convolution (to be learned).

Pointwise non-linear layers are added after convolutional layers. In our experience, we surprisingly found that using hyperbolic tangents, their piecewise linear counterpart HardTanh (as in [16]) or ReLU units lead to similar results.

There are some slight variations between the architectures, depending on the input features. MFCC-based networks need less striding, as standard MFCC filters are applied with large strides on the input raw sequence. With power spectrum-based and raw wave-based networks, we observed that the overall stride of the network was more important than where the convolution with strides were placed. We found thus preferrable to set the strided convolutions near the first input layers of the network, as it leads to the fastest architectures: with power spectrum features or raw wave, the input sequences are very long and the first convolutions are thus the most expensive ones.

The last layer of our convolutional network outputs one score per letter in the letter dictionary ($d_y = |\mathcal{L}|$). Our architecture for raw wave is shown in Figure 1 and is inspired by [16]. The architectures for both power spectrum and MFCC features do not include the first layer. The full network can be seen as a non-linear convolution, with a kernel width of size 31280 and stride equal to 320; given the sample rate of our data is 16KHz, label scores are produced using a window of 1955 ms, with steps of 20ms.

2.3 Inferring Segmentation with AutoSegCriterion

Most large labeled speech databases provide only a text transcription for each audio file. In a classification framework (and given our acoustic model produces letter predictions), one would need the segmentation of each letter in the transcription to train properly the model. Unfortunately, manually labeling the segmentation of each letter would be tedious. Several solutions have been explored in the speech community to alleviate this issue: HMM/GMM models use an iterative EM procedure: (i) during the Estimation step, the best segmentation is inferred, according to the current model, by maximizing the joint probability of the letter (or any sub-word unit) transcription and input sequence. (ii) During the Maximization step the model is optimized by minimizing a frame-level criterion, based on the (now fixed) inferred segmentation. This approach is also often used to boostrap the training of neural network-based acoustic models.

Other alternatives have been explored in the context of hybrid HMM/NN systems, such as the MMI criterion [2] which maximizes the mutual information between the acoustic sequence and word sequences or the Minimum Bayse Risk (MBR) criterion [5].

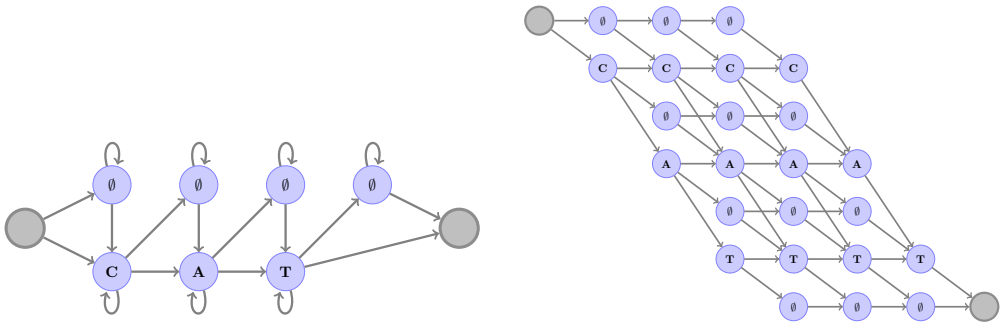
More recently, standalone neural network architectures have been trained using criterions which jointly infer the segmentation of the transcription while increase the overall score of the right transcription [6, 17]. The most popular one is certainly the Connectionist Temporal Classification (CTC) criterion, which is at the core of Baidu's Deep Speech architecture [1]. CTC assumes that the network output probability scores, normalized at the frame level. It considers all possible sequence of letters (or any sub-word units), which can lead to a to a given transcription. CTC also allow a special "blank" state to be optionally inserted between each letters. The rational behind the blank state is two-folds: (i) modeling "garbage" frames which might occur between each letter and (ii) identifying the separation between two identical consecutive letters in a transcription. Figure 2a shows an example of the sequences accepted by CTC for a given transcription. In practice, this graph is unfolded as shown in Figure 2b, over the available frames output by the acoustic model. We denote $\mathcal{G}_{ctc}(\theta, T)$ an unfolded graph over T frames for a given transcription θ, and $\pi = \pi_1, \ldots, \pi_T \in \mathcal{G}_{ctc}(\theta, T)$ a path in this graph representing a (valid) sequence of letters for this transcription. At each time step t, each node of the graph is assigned with the corresponding log-probability letter (that we denote $f_t(\cdot)$) output by the acoustic model. CTC aims at maximizing the "overall" score of paths in $\mathcal{G}_{ctc}(\theta, T)$; for that purpose, it minimizes the Forward score:

$$CTC(\theta, T) = - \operatorname*{logadd}_{\pi \in \mathcal{G}_{ctc}(\theta,T)} \sum_{t=1}^{T} f_{\pi_t}(x) \,, \tag{2}$$

where the "logadd" operation, also often called "log-sum-exp" is defined as $\text{logadd}(a, b) = \exp(\log(a) + \log(b))$. This overall score can be efficiently computed with the Forward algorithm. To put things in perspective, if one would replace the $\text{logadd}(\cdot)$ by a $\max(\cdot)$ in (2) (which can be then efficiently computed by the Viterbi algorithm, the counterpart of the Forward algorithm), one would then maximize the score of the *best* path, according to the model belief. The $\text{logadd}(\cdot)$ can be seen as a smooth version of the $\max(\cdot)$: paths with similar scores will be attributed the same weight in the overall score (and hence receive the same gradient), and paths with much larger score will have much more overall weight than paths with low scores. In practice, using the $\text{logadd}(\cdot)$ works much better than the $\max(\cdot)$. It is also worth noting that maximizing (2) does not diverge, as the acoustic model is assumed to output normalized scores (log-probabilities) $f_i(\cdot)$.

In this paper, we explore an alternative to CTC, with three differences: (i) there are no blank labels, (ii) un-normalized scores on the nodes (and possibly un-normalized transition scores on the edges) (iii) global normalization instead of per-frame normalization:

- The advantage of (i) is that it produces a much simpler graph (see Figure 3a and Figure 3b). We found that in practice there was no advantage of having a blank class to model the



(a) (b)

Figure 2: The CTC criterion graph. (a) Graph which represents all the acceptable sequences of letters (with the blank state denoted "$\emptyset$"), for the transcription "cat". (b) Shows the same graph unfolded over 5 frames. There are no transitions scores. At each time step, nodes are assigned a conditional probability output by the neural network acoustic model.

possible "garbage" frames between letters. Modeling letter repetitions (which is also an important quality of the blank label in CTC) can be easily replaced by repetition character labels (we used two extra labels for two and three repetitions). For example "caterpillar" could be written as "caterpil2ar", where "2" is a label to represent the repetition of the previous letter. Not having blank labels also simplifies the decoder.

- With (ii) one can easily plug an external language model, which would insert transition scores on the edges of the graph. This could be particularly useful in future work, if one wanted to model representations more high-level than letters. In that respect, avoiding normalized transitions is important to alleviate the problem of "label bias" [3, 11]. In this work, we limited ourselves to transition scalars, which are learned together with the acoustic model.
- The normalization evoked in (iii) is necessary when using un-normalized scores on nodes or edges; it insures incorrect transcriptions will have a low confidence.

In the following, we name our criterion "Auto Segmentation Criterion" (ASG). Considering the same notations than for CTC in (2), and an unfolded graph $\mathcal{G}_{asg}(\theta, T)$ over T frames for a given transcription θ (as in Figure 3b), as well as a fully connected graph $\mathcal{G}_{full}(\theta, T)$ over T frames (representing all possible sequence of letters, as in Figure 3c), ASG aims at minimizing:

This is similar to CTC but with transition matrix added for the sequence of chars between example for example in case of cat between (c,a,t)

this is a penalization term that contains transition matrix between all tokens (size * size) and it demotes other sequences

$$ASG(\theta, T) = -\operatorname*{logadd}_{\pi \in \mathcal{G}_{asg}(\theta,T)} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x)) + \operatorname*{logadd}_{\pi \in \mathcal{G}_{full}(\theta,T)} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x)), \quad (3)$$

where $g_{i,j}(\cdot)$ is a transition score model to jump from label i to label j. The left-hand part of 3 promotes sequences of letters leading to the right transcription, and the right-hand part demotes all sequences of letters. As for CTC, these two parts can be efficiently computed with the Forward algorithm. Derivatives with respect to $f_i(\cdot)$ and $g_{i,j}(\cdot)$ can be obtained (maths are a bit tedious) by applying the chain rule through the Forward recursion.

2.4 Beam-Search Decoder

We wrote our own one-pass decoder, which performs a simple beam-search with beam threholding, histogram pruning and language model smearing [26]. We kept the decoder as simple as possible (under 1000 lines of C code). We did not implement any sort of model adaptation before decoding, nor any word graph rescoring. Our decoder relies on KenLM [9] for the language modeling part. It also accepts un-normalized acoustic scores (transitions and emissions from the acoustic model) as input. The decoder attempts to maximize the following:

$$\mathcal{L}(\theta) = \operatorname*{logadd}_{\pi \in \mathcal{G}_{asg}(\theta,T)} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x)) + \alpha \log P_{lm}(\theta) + \beta |\theta|, \quad (4)$$

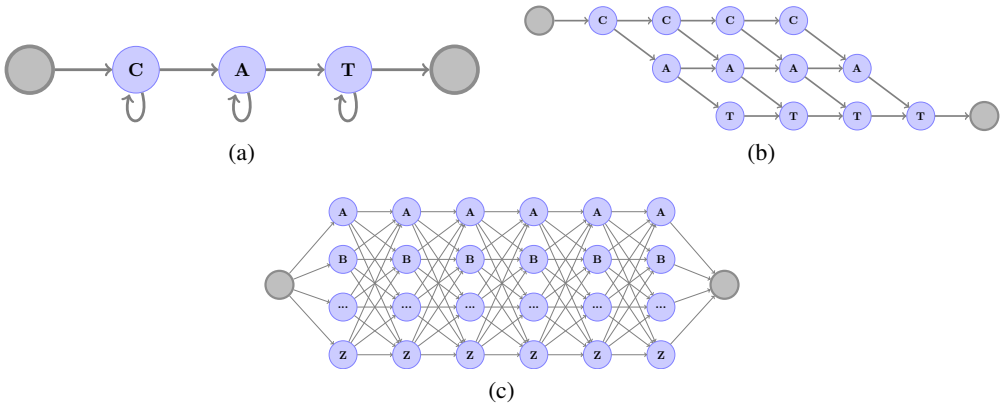


Figure 3: The ASG criterion graph. (a) Graph which represents all the acceptable sequences of letters for the transcription "cat". (b) Shows the same graph unfolded over 5 frames. (c) Shows the corresponding fully connected graph, which describe all possible sequences of letter; this graph is used for normalization purposes. Un-normalized transitions scores are possible on the edges. At each time step, nodes are assigned a conditional un-normalized score, output by the neural network acoustic model.

where $P_{lm}(\theta)$ is the probability of the language model given a transcription θ, α and β are two hyper-parameters which control the weight of the language model and the word insertion penalty respectively.

3 Experiments

We implemented everything using Torch7[1]. The ASG criterion as well as the decoder were implemented in C (and then interfaced into Torch).

We consider as benchmark LibriSpeech, a large speech database freely available for download [18]. LibriSpeech comes with its own train, validation and test sets. Except when specified, we used all the available data (about 1000h of audio files) for training and validating our models. We use the original 16 KHz sampling rate. The vocabulary $\mathcal{L}$ contains 30 graphemes: the standard English alphabet plus the apostrophe, silence, and two special "repetition" graphemes which encode the duplication (once or twice) of the previous letter (see Section 2.3).

The architecture hyper-parameters, as well the decoder ones were tuned using the validation set. In the following, we either report letter-error-rates (LERs) or word-error-rates (WERs). WERs have been obtained by using our own decoder (see Section 2.4), with the standard 4-gram language model provided with LibriSpeech[2].

MFCC features are computed with 13 coefficients, a 25 ms sliding window and 10 ms stride. We included first and second order derivatives. Power spectrum features are computed with a 25 ms window, 10 ms stride, and have 257 components. All features are normalized (mean 0, std 1) per input sequence.

3.1 Results

Table 1 reports a comparison between CTC and ASG, in terms of LER and speed. Our ASG criterion is implemented in C (CPU only), leveraging SSE instructions when possible. Our batching is done with an OpenMP parallel for. We picked the CTC criterion implementation provided by Baidu[3]. Both criteria lead to the same LER. For comparing the speed, we report performance for sequence sizes as reported initially by Baidu, but also for longer sequence sizes, which corresponds to our average use

[1] `http://www.torch.ch`.

[2] `http://www.openslr.org/11`.

[3] `https://github.com/baidu-research/warp-ctc`.

Table 1: CTC vs ASG. CTC is Baidu's implementation. ASG is implemented on CPU (core in C, threading in Lua). (a) reports performance in LER. Timings (in *ms*) for small sequences (input frames: 150, letter vocabulary size: 28, transcription size: 40) and long sequences (input frames: 700, letter vocabulary size: 28, transcription size: 200) are reported in (b) and (c) respectively. Timings include both forward and backward passes. CPU implementations use 8 threads.

(a)

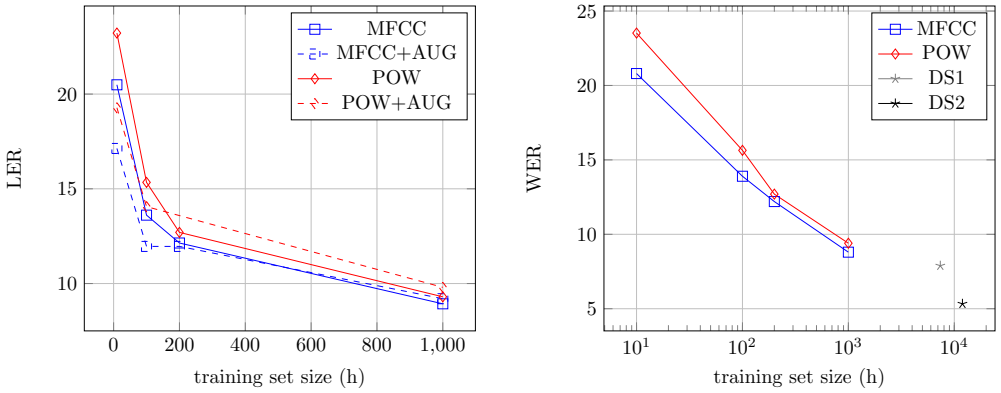
	ASG	CTC
dev-clean	10.4	10.7
test-clean	10.1	10.5

(b)

batch size	CTC		ASG
	CPU	GPU	CPU
1	1.9	5.9	2.5
4	2.0	6.0	2.8
8	2.0	6.1	2.8

(c)

batch size	CTC		ASG
	CPU	GPU	CPU
1	40.9	97.9	16.0
4	41.6	99.6	17.7
8	41.7	100.3	19.2



(a) (b)

Figure 4: Valid LER (a) and WER (b) v.s. training set size (10h, 100h, 200h, 1000h). This compares MFCC-based and power spectrum-based (POW) architectures. AUG experiments include data augmentation. In (b) we provide Baidu Deep Speech 1 and 2 numbers on LibriSpeech, as a comparison [8, 1].

case. ASG appears faster on long sequences, even though it is running on CPU only. Baidu's GPU CTC implementation seems more aimed at larger vocabularies (e.g. 5000 Chinese characters).

We also investigated the impact of the training size on the dataset, as well as the effect of a simple data augmentation procedure, where shifts were introduced in the input frames, as well as stretching. For that purpose, we tuned the size of our architectures (given a particular size of the dataset), to avoid over-fitting. Figure 4a shows the augmentation helps for small training set size. However, with enough training data, the effect of data augmentation vanishes, and both type of features appear to perform similarly. Figure 4b reports the WER with respect to the available training data size. We observe that we compare very well against Deep Speech 1 & 2 which were trained with much more data [8, 1].

Finally, we report in Table 2 the best results of our system so far, trained on 1000h of speech, for each type of features. The overall stride of architectures is 320 (see Figure 1), which produces a label every 20 ms. We found that one could squeeze out about 1% in performance by refining the precision of the output. This is efficiently achieved by shifting the input sequence, and feeding it to the network

Table 2: LER/WER of the best sets of hyper-parameters for each feature types.

	MFCC		PS		Raw	
	LER	WER	LER	WER	LER	WER
dev-clean	6.9		9.3		10.3	
test-clean	6.9	7.2	9.1	9.4	10.6	10.1

several times. Results in Table 2 were obtained by a single extra shift of 10 ms. Both power spectrum and raw features are performing slightly worse than MFCCs. One could expect, however, that with enough data (see Figure 4) the gap would vanish.

4 Conclusion

We have introduced a simple end-to-end automatic speech recognition system, which combines a standard 1D convolutional neural network, a sequence criterion which can infer the segmentation, and a simple beam-search decoder. The decoding results are competitive on the LibriSpeech corpus with MFCC features (7.2% WER), and promising with power spectrum and raw speech (9.4% WER and 10.1% WER respectively). We showed that our AutoSegCriterion can be faster than CTC [6], and as accurate (table 1). Our approach breaks free from HMM/GMM pre-training and force-alignment, as well as not being as computationally intensive as RNN-based approaches [1] (on average, one LibriSpeech sentence is processed in less than 60ms by our ConvNet, and the decoder runs at 8.6x on a single thread).

References

[1] AMODEI, D., ANUBHAI, R., BATTENBERG, E., CASE, C., CASPER, J., CATANZARO, B., CHEN, J., CHRZANOWSKI, M., COATES, A., DIAMOS, G., ET AL. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595* (2015).

[2] BAHL, L. R., BROWN, P. F., DE SOUZA, P. V., AND MERCER, R. L. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 1986 IEEE International Conference on* (1986), IEEE, pp. 49–52.

[3] BOTTOU, L. *Une approche theorique de l'apprentissage connexionniste et applications a la reconnaissance de la parole*. PhD thesis, 1991.

[4] BOTTOU, L., BENGIO, Y., AND LE CUN, Y. Global training of document processing systems using graph transformer networks. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on* (1997), IEEE, pp. 489–494.

[5] GIBSON, M., AND HAIN, T. Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition. In *Proceedings of INTERSPEECH* (2006), IEEE, pp. 2406—-2409.

[6] GRAVES, A., FERNÁNDEZ, S., GOMEZ, F., AND SCHMIDHUBER, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (2006), ACM, pp. 369–376.

[7] GRAVES, A., MOHAMED, A.-R., AND HINTON, G. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 6645–6649.

[8] HANNUN, A., CASE, C., CASPER, J., CATANZARO, B., DIAMOS, G., ELSEN, E., PRENGER, R., SATHEESH, S., SENGUPTA, S., COATES, A., ET AL. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[9] HEAFIELD, K., POUZYREVSKY, I., CLARK, J. H., AND KOEHN, P. Scalable modified kneser-ney language model estimation. In *ACL (2)* (2013), pp. 690–696.

[10] HINTON, G., DENG, L., YU, D., DAHL, G. E., MOHAMED, A.-R., JAITLY, N., SENIOR, A., VANHOUCKE, V., NGUYEN, P., SAINATH, T. N., ET AL. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE 29*, 6 (2012), 82–97.

[11] LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Eighteenth International Conference on Machine Learning, ICML* (2001).

[12] LECUN, Y., AND BENGIO, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[13] MIAO, Y., GOWAYYED, M., AND METZE, F. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. *arXiv preprint arXiv:1507.08240* (2015).

[14] MOHAMED, A.-R., DAHL, G. E., AND HINTON, G. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on 20*, 1 (2012), 14–22.

[15] PALAZ, D., COLLOBERT, R., AND DOSS, M. M. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018* (2013).

[16] PALAZ, D., COLLOBERT, R., ET AL. Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of Interspeech* (2015), no. EPFL-CONF-210029.

[17] PALAZ, D., MAGIMAI-DOSS, M., AND COLLOBERT, R. Joint phoneme segmentation inference and classification using crfs. In *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on* (2014), IEEE, pp. 587–591.

[18] PANAYOTOV, V., CHEN, G., POVEY, D., AND KHUDANPUR, S. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (2015), IEEE, pp. 5206–5210.

[19] PEDDINTI, V., CHEN, G., MANOHAR, V., KO, T., POVEY, D., AND KHUDANPUR, S. Jhu aspire system: Robust lvcsr with tdnns, i-vector adaptation, and rnn-lms. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop* (2015).

[20] PEDDINTI, V., POVEY, D., AND KHUDANPUR, S. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proceedings of INTERSPEECH* (2015).

[21] SAON, G., KUO, H.-K. J., RENNIE, S., AND PICHENY, M. The ibm 2015 english conversational telephone speech recognition system. *arXiv preprint arXiv:1505.05899* (2015).

[22] SAON, G., SOLTAU, H., NAHAMOO, D., AND PICHENY, M. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU* (2013), pp. 55–59.

[23] SENIOR, A., HEIGOLD, G., BACCHIANI, M., AND LIAO, H. Gmm-free dnn training. In *Proceedings of ICASSP* (2014), pp. 5639–5643.

[24] SERCU, T., PUHRSCH, C., KINGSBURY, B., AND LECUN, Y. Very deep multilingual convolutional neural networks for lvcsr. *arXiv preprint arXiv:1509.08967* (2015).

[25] SOLTAU, H., SAON, G., AND SAINATH, T. N. Joint training of convolutional and non-convolutional neural networks. In *ICASSP* (2014), pp. 5572–5576.

[26] STEINBISS, V., TRAN, B.-H., AND NEY, H. Improvements in beam search. In *ICSLP* (1994), vol. 94, pp. 2143–2146.

[27] WOODLAND, P. C., AND YOUNG, S. J. The htk tied-state continuous speech recogniser. In *Eurospeech* (1993).