Letter-Based Speech Recognition with Gated ConvNets

Vitaliy Liptchinsky VITALIY888@FB.COM
Facebook AI Research
Menlo Park, USA

Gabriel Synnaeve GAB@FB.COM
Facebook AI Research
New York, USA

Ronan Collobert LOCRONAN@FB.COM
Facebook AI Research
Menlo Park, USA

Abstract

In this paper we introduce a new speech recognition system, leveraging a simple letter-based ConvNet acoustic model. The acoustic model requires only audio transcription for training – no alignment annotations, nor any forced alignment step is needed. At inference, our decoder takes only a word list and a language model, and is fed with letter scores from the acoustic model – no phonetic word lexicon is needed. Key ingredients for the acoustic model are Gated Linear Units and high dropout. We show near state-of-the-art results in word error rate on the LibriSpeech corpus (Panayotov et al., 2015) using log-mel filterbanks, both on the CLEAN and OTHER configurations.

1. Introduction

Top speech recognition systems are either complicated pipelines or using more data that is publicly available. We set out to show that it is possible to train a nearly state of the art speech recognition system for read speech, with a public dataset (LibriSpeech), on a GPU-equipped workstation. Thus, we present an end-to-end system for speech recognition, going log-mel filterbanks to the transcription in words. The acoustic model is trained using letters (graphemes) directly, which take out the need for an intermediate (human or automatic) phonetic transcription.

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 or senone states). This approach takes its roots in HMM/GMM training (Woodland and Young, 1993). The improvements brought by deep neural networks (DNNs) (Mohamed et al., 2012; Hinton et al., 2012) and convolutional neural networks (CNNs) (Sercu et al., 2016; Soltau et al., 2014) for acoustic modeling only extend this training pipeline. The current state of the art on LibriSpeech belongs to this approach too (Panayotov et al., 2015; Peddinti et al., 2015b), with an additional step of speaker adaptation (Saon et al., 2013; Peddinti et al., 2015a). Recently, Senior et al. (2014) proposed GMM-free training, but the approach still requires to generate a forced alignment.

An approach that cut ties with the HMM/GMM pipeline (and with forced alignment) was to train with a recurrent neural network (RNN) (Graves et al., 2013) for phoneme transcription. This approach has been then extended to character-based systems and

improved with attention mechanisms (Bahdanau et al., 2016; Chan et al., 2016), but best systems are still behind state-of-the-art phone-based (or senone-based) systems. Competitive end-to-end approaches are acoustic models toppled with RNNs layers as in (Hannun et al., 2014; Miao et al., 2015; Saon et al., 2015; Amodei et al., 2016) trained with a sequence criterion Tang et al. (2017) (the most popular ones being CTC (Graves et al., 2006) and MMI (Bahl et al., 1986)). On conversational speech (that is not the topic of this paper), the state of the art is still held by complex ConvNets+RNNs acoustic models (which are also trained or refined with a sequence criterion), coupled to domain-adapted language models (Xiong et al., 2017; Saon et al., 2017).

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, we propose an RNN-free architecture based on convolutional networks for the acoustic model, toppled with a simple sequence-level variant of CTC.

We reach the clean speech performance of (Peddinti et al., 2015b), but without performing speaker adaptation. Our word-error-rate on clean speech is better than (Amodei et al., 2016), while being worse on noisy speech, but they train on 11,900 hours while we only train on the 960h available in LibriSpeech's train set. 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 and decoding approaches. The last section shows experimental results on LibriSpeech.

2. Architecture

Our acoustic model (see an overview in Figure 1) is a Convolutional Neural Network (ConvNet) (LeCun and Bengio, 1995), with Gated Linear Units (GLUs) (Dauphin et al., 2017). The model is fed with 40 log-mel filterbank energies, and is trained with a variant of the Connectionist Temporal Classification (CTC) criterion (Graves et al., 2006), which does not have blank labels but embarks a simple duration model through letter transitions scores (Collobert et al., 2016). During training, we use dropout on the neural network outputs. At inference, the acoustic model is coupled with a decoder which performs a beam search, constrained with a count-based language model. We detail each of these components in the following.

2.1 Log-Mel Filterbanks

Our system relies on standard log-mel filterbanks, which are obtained by averaging spectrogram values with mel-scale filters. Log-mel filterbanks are the step preceding the cosine transform required to compute Mel-Frequency Cepstrum Coefficients (MFCCs), often found in classical HMM/GMM speech systems (Woodland and Young, 1993) because of their dimensionality compression (13 coefficients are often enough to span speech frequencies). Compared to spectrogram coefficients, log-mel filterbanks have the advantage to be more robust to small time-warping deformations.

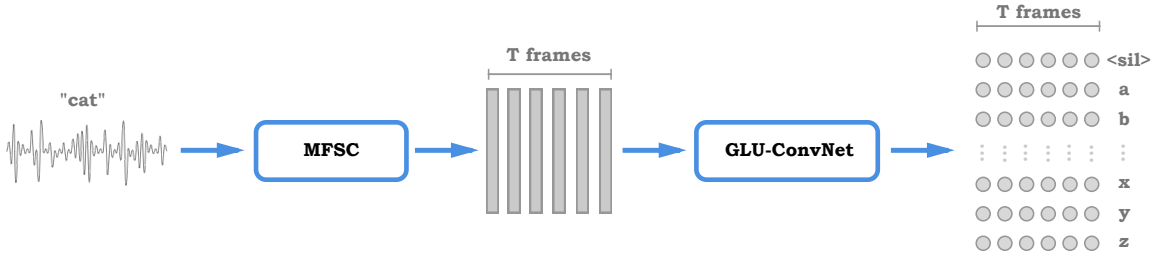


Figure 1: Overview of our acoustic model, which computes log-mel filterbanks which are fed to a Gated ConvNet. The ConvNet outputs one score for each letter in the dictionary, and for each input feature frame. At inference time, these scores are fed to a decoder (see Section 2.4) to form the most likely sequence of words. At training time, the scores are fed to the ASG criterion (see Figure 2) which promotes sequences of letters leading to the transcrition sequence (here "c a t").

2.2 Gated ConvNets for Acoustic Modeling

Our acoustic model is fed with the log-mel filterbank frames, and outputs letter scores for each input frame. At each time step, there is one score per letter in a given dictionary $\mathcal{L}$. Words are separated by a special letter `<sil>`.

The acoustic model architecture is based on a 1D Gated Convolutional Neural Network (Gated ConvNet). 1D ConvNets were introduced early in the speech community, and are also referred as Time-Delay Neural Networks (TDNNs) (Waibel, 1989). Gated ConvNets (Dauphin et al., 2017) stack 1D convolutions with Gated Linear Units. More formally, given an input sequence $\mathbf{X} \in \mathbb{R}^{T \times d^i}$ with T frames of d-dimensional vectors, the i^{th} layer of our network performs the following computation:

$$h^i(\mathbf{X}) = (\mathbf{X} * \mathbf{W}^i + \mathbf{b}^i) \otimes \sigma(\mathbf{X} * \mathbf{V}^i + \mathbf{c}^i)\,, \tag{1}$$

where $*$ is the convolution operator, $\mathbf{W}^i$, $\mathbf{V}^i \in \mathbb{R}^{d^{i+1} \times d^i \times k^i}$ and $\mathbf{b}^i$, $\mathbf{c}^i \in \mathbb{R}^{d^{i+1}}$ are the learned parameters (with convolution kernel size k^i), $\sigma(\cdot)$ is the sigmoid function and $\otimes$ is the element-wise product between matrices.

Gated ConvNets have been shown to reduce the vanishing gradient problem, as they provide a linear path for the gradients while retaining non-linear capabilities, leading to state-of-the-art performance both for natural language modeling and machine translation tasks (Dauphin et al., 2017; Gehring et al., 2017).

2.2.1 Feature Normalization and Zero-Padding

Each input feature sequence is normalized with mean 0 and variance 1. Given an input sequence $\mathbf{X} \in \mathbb{R}^{T \times d}$, a convolution with kernel size k will output $T - k + 1$ frames, due to border effects. To compensate those border effects, we pad the log-mel filterbanks $\mathbf{X}^0$ with zeroed frames. To take in account the whole network, the padding size is $\sum_i (k^i - 1)$, divided in two equal parts at the beginning and the end of the sequence.

2.3 Acoustic Model Training

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. Manually labeling the segmentation of each letter would be tedious. Several solutions have been explored in the speech community to alleviate this issue:

1. HMM/GMM models use an iterative EM procedure: during the Estimation step, the best segmentation is inferred according to the current model, during the Maximization step the model is optimized using the current inferred segmentation. This approach is also often used to boostrap the training of neural network-based acoustic models.

2. The MMI criterion (Bahl et al., 1986) maximizes the mutual information between the acoustic sequence and word sequences or the Minimum Bayes Risk (MBR) criterion (Gibson and Hain, 2006). Recent state-of-the-art systems leverage the MMI criterion (Povey et al., 2016).

3. Standalone neural network architectures have also been trained using the Connectionist Temporal Classification (CTC), which jointly infers the segmentation of the transcription while increasing the overall score of the right transcription (Graves et al., 2006). In (Amodei et al., 2016) it has been shown that letter-based acoustic models trained with CTC could compete with existing phone-based (or senone-based) systems, assuming enough training data is provided.

In this paper, we chose a variant of the Connectionist Temporal Classification. CTC considers all possible sequence sub-word units (e.g. letters), which can lead to the correct transcription. It also allows a special "blank" state to be optionally inserted between each sub-word unit. The rationale behind the blank state is two-fold: (i) modeling "garbage" frames which might occur between each letter and (ii) identifying the separation between two identical consecutive sub-word units in a transcription. Figure 2a shows the CTC graph describing all the possible sequences of letters leading to the word "cat", over 6 frames. We denote $\mathcal{G}_{ctc}(\theta, T)$ the CTC acceptance 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. CTC assumes that the network output probability scores, normalized at the frame level. At each time step t, each node of the graph is assigned with its corresponding log-probability letter i (that we denote $f_i^t(\mathbf{X})$) output by the acoustic model (given an acoustic sequence $\mathbf{X}$). CTC minimizes the Forward score over the graph $\mathcal{G}_{ctc}(\theta, T)$:

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

where the "logadd" operation (also called "log-sum-exp") is defined as $\text{logadd}(a, b) = \log(\exp(a) + \exp(b))$. This overall score can be efficiently computed with the Forward algorithm.

2.3.1 The ASG Criterion

Blank labels introduce code complexity when decoding letters into words. Indeed, with blank labels "ø", a word gets many entries in the sub-word unit transcription dictionary (e.g. the

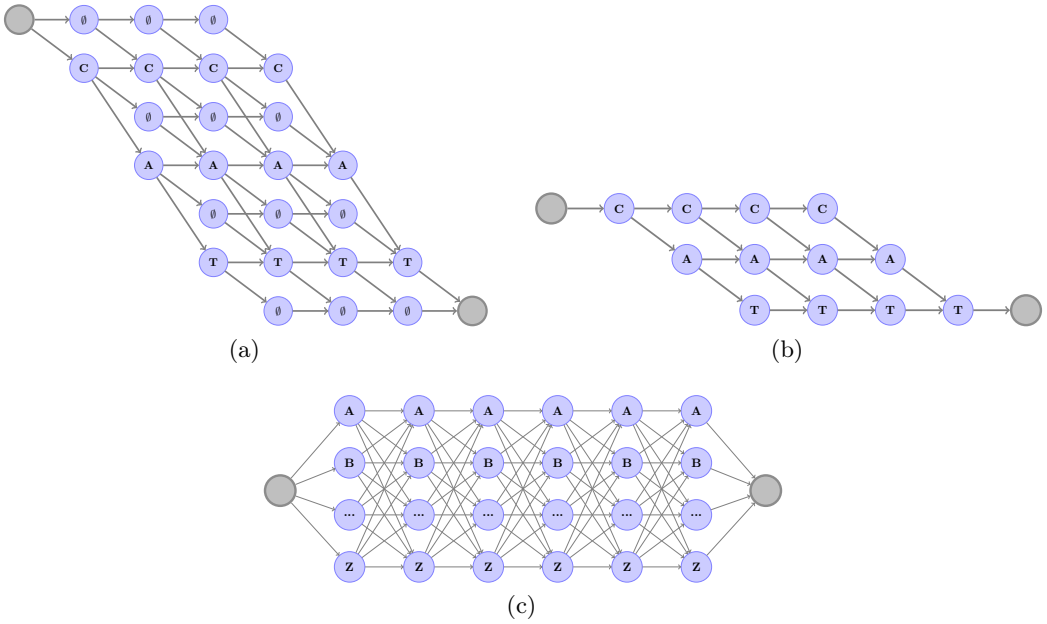


Figure 2: (a) The CTC graph which represents all the acceptable sequences of letters for the transcription "cat" over 6 frames. (b) The same graph used by ASG, where blank labels have been discarded. (c) The fully connected graph describing all possible sequences of letter; this graph is used for normalization purposes in ASG. Un-normalized transitions scores are possible on edges of these graphs. At each time step, nodes are assigned a conditional un-normalized score, output by the Gated ConvNet acoustic model.

word "cat" can be represented as "c a t", "c ø a t", "c ø a t", "c ø a ø t", etc... – instead of only "c a t"). We replace the blank label by special letters modeling repetitions of preceding letters. For example "caterpillar" can be written as "caterpil1ar", where "1" is a label to represent one repetition of the previous letter.

Removing blank labels from the CTC acceptance graph $\mathcal{G}_{ctc}(\theta, T)$ (shown in Figure 2a) leads to a simpler graph that we denote $\mathcal{G}_{asg}(\theta, T)$ (shown in Figure 2b). Unfortunately, in practice we observed that most models do not train with this simplification of CTC. Adding *unnormalized* transition scores $g_{i,j}(\cdot)$ on each edge of the graph, when moving from label i to label j fixes the issue. We observed in practice that *normalized* transitions led to similar issue as when not having transitions. Considering unnormalized transition scores implies implementing a sequence-level normalization, to prevent the model from diverging (represented by the graph $\mathcal{G}_{asg}(\theta, T)$, as shown in Figure 2c). This leads to the following criterion, dubbed ASG for "Auto SeGmentation":

$$ASG(\theta, T) = - \operatorname*{logadd}_{\pi \in \mathcal{G}_{asg}(\theta,T)} \sum_{t=1}^{T} (f^t_{\pi_t}(\mathbf{X}) + g_{\pi_{t-1},\pi_t}(\mathbf{X})) + \operatorname*{logadd}_{\pi \in \mathcal{G}_{full}(\theta,T)} \sum_{t=1}^{T} (f^t_{\pi_t}(\mathbf{X}) + g_{\pi_{t-1},\pi_t}(\mathbf{X})) \,. \tag{3}$$

The left-hand part in Equation (3) promotes the score of letter sequences leading to the right transcription (as in Equation (2) for CTC), and the right-hand part denotes the score of all sequences of letters (as does the frame-level normalization – that is the softmax on

the acoustic model – for CTC). As for CTC, these two parts can be efficiently computed with the Forward algorithm. When removing transitions in Equation (3), the sequence-level normalization becomes equivalent to the frame-level normalization and the ASG criterion is mathematically equivalent to CTC with no blank labels. As for MMI, ASG is a discriminative criterion; however ASG operates on unnormalized scores, considering a discriminative *model* of the form "$S(\pi|\mathbf{X})$" rather than a generative model "$P(\mathbf{X}|\pi)$" in MMI. See (Tang et al., 2017) for a more detailed comparison.

2.3.2 Other Training Considerations

We apply dropout at the output to all layers of the acoustic model. Dropout retains each output with a probability p, by applying a multiplication with a Bernoulli random variable taking value $1/p$ with probability p and 0 otherwise (Srivastava et al., 2014).

Following the original implementation of Gated ConvNets (Dauphin et al., 2017), we found that using both weight normalization (Salimans and Kingma, 2016) and gradient clipping (Pascanu et al., 2013) were speeding up training convergence. The clipping we implemented performs:

$$\widetilde{\nabla C} = \max(||\nabla C||, \epsilon) \frac{\nabla C}{||\nabla C||}\,, \tag{4}$$

where C is either the CTC or ASG criterion, and ϵ is some hyper-parameter which controls the maximum amplitude of the gradients.

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 (Steinbiss et al., 1994). 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 (Heafield et al., 2013) for the language modeling part. It also accepts unnormalized acoustic scores (transitions and emissions from the acoustic model) as input. The decoder attempts to maximize the following:

$$\mathcal{L}(\theta) = \underset{\pi \in \mathcal{G}_{asg}(\theta,T)}{\text{logadd}} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x)) + \alpha \log P_{lm}(\theta) + \beta|\theta| + \gamma|\{i|\pi_i = \texttt{<sil>}\}|\,, \tag{5}$$

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

The beam of the decoder tracks paths with highest scores according to Equation (5), by bookkeeping pairs of (language model, lexicon) states, as it goes through time. The language model state corresponds to the $(n-1)$-gram history of the n-gram language model, while the lexicon state is the sub-word unit position in the current word hypothesis. To maintain diversity in the beam, paths with identical (language model, lexicon) states are merged. Note that traditional decoders combine the scores of the merged paths with a $\max(\cdot)$ operation (as in a Viterbi beam-search) – which would correspond to a $\max(\cdot)$ operation in Equation (5) instead of $\text{logadd}(\cdot)$. We consider instead the $\text{logadd}(\cdot)$ operation, as it takes into account

Table 1: Architectures details. "#conv." is the number of convolutional layers. Dropout amplitude, "#hu" (number of output hidden units) and "kw" (convolution kernel width) are provided for the first and last layer (all are linearly increased with layer depth). The size of the final layer is also provided.

Architecture	#conv.	dropout first layer	dropout last layer	#hu first layer	#hu last layer	kw first layer	kw last layer	#hu full connect
Low Dropout	17	0.25	0.25	200	750	13	27	1500
High Dropout	19	0.20	0.60	200	1000	13	29	2000

the contribution of all the paths leading to the same transcription, in the same way we do during training (see Equation (3)). In Section 3.1, we show that this leads to better accuracy in practice.

3. Experiments

We benchmarked our system on LibriSpeech, a large speech database freely available for download (Panayotov et al., 2015). We kept the original 16 kHz sampling rate. We considered the two available setups in LibriSpeech: CLEAN data and OTHER. We picked all the available data (about 960h of audio files) for training, and the available development sets (both for CLEAN, and OTHER) for tuning all the hyper-parameters of our system. Test sets were used only for the final evaluations.

The letter vocabulary $\mathcal{L}$ contains 30 graphemes: the standard English alphabet plus the apostrophe, silence (<SIL>), and two special "repetition" graphemes which encode the duplication (once or twice) of the previous letter (see Section 2.3.1). Decoding is achieved with our own decoder (see Section 2.4), with the standard 4-gram language model provided with LibriSpeech[1], which contains $200,000$ words. In the following, we either report letter-error-rates (LERs) or word-error-rates (WERs).

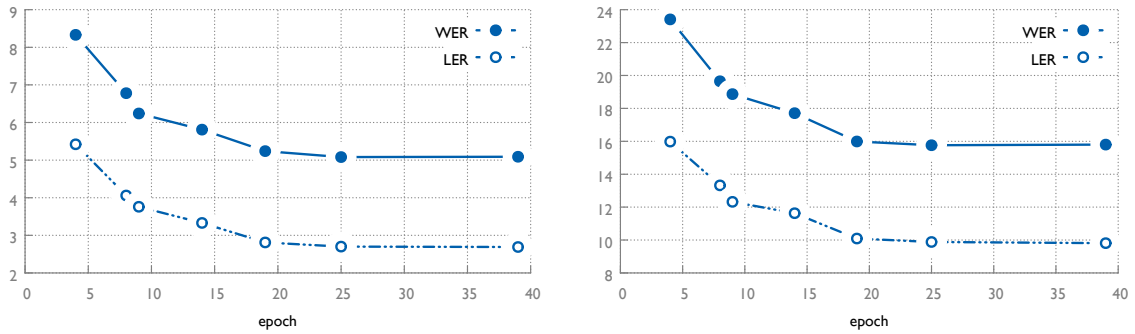
Training was achieved with mini-batch of 4 utterances. Clipping parameter (see Equation (4) was set to $\epsilon = 0.2$. We used a momentum of 0.9. Log-mel filterbanks are computed with 40 coefficients, a 25 ms sliding window and 10 ms stride.

We implemented everything using TORCH7[2]. The ASG criterion as well as the decoder were implemented in C (and then interfaced into TORCH).

3.1 Architecture

We tuned our acoustic model architectures by grid search, validating on the dev sets. We consider here two architectures, with low and high amount of dropout (see the parameter p in Section 2.3.2). Table 1 reports the details of our architectures. The amount of dropout, number of hidden units, as well as the convolution kernel width are increased linearly with the

1. `http://www.openslr.org/11`.
2. `http://www.torch.ch`. Our implementation is available at `https://github.com/facebookresearch/wav2letter`.



(a) (b)

Figure 3: LibriSpeech Letter Error Rate (LER) and Word Error Rate (WER) for the first training epochs of our Low Dropout architecture. (a) is on `dev-clean`, (b) on `dev-other`.

Table 2: Comparison in LER and WER of variants of our model on LibriSpeech. When not specified, decoding is performed with the logadd$(\cdot)$ operation to aggregate similar hypotheses (see Section 2.4).

	dev-clean		dev-other	
model	LER	WER	LER	WER
Low Dropout ($p = 0.2$)	2.7	4.8	9.8	15.2
High Dropout ($p = 0.2 \rightarrow 0.6$)	2.3	4.6	9.0	13.8
High Dropout + max$(\cdot)$ decoding	–	4.7	–	14.0

depth of the neural network. Note that as we use Gated Linear Units (see Section 2.2), each layer is duplicated as stated in Equation (1). Convolutions are followed by a fully connected layer, before the final layer which outputs 30 scores (one for each letter in the dictionary). This leads to about 130M and 208M of trainable parameters for the Low Dropout and High Dropout architectures, respectively.

Figure 3 shows the LER and WER on the LibriSpeech development sets, for the first 40 training epochs of our Low Dropout architecture. LER and WER appear surpringly well correlated, both on the "clean" and "other" version of the dataset.

In Table 2, we report WERs on the LibriSpeech development sets, both for our Low Dropout and High Dropout architectures. Increasing dropout regularize the acoustic model in a way which impacts significantly generalization, the effect being stronger on noisy speech. We also report the WER for the decoder ran with the max$(\cdot)$ operation (instead of logadd$(\cdot)$ for other results) used to aggregate paths in the beam with identical (language model, lexicon) states. It appears advantageous (as there is no code complexity increase in the decoder – one only needs to replace max$(\cdot)$ by logadd$(\cdot)$ in the code) to use the logadd$(\cdot)$ aggregation.

Table 3: Comparison of different ASR systems. We report the type of acoustic model used for various systems, as well as the type of sub-word unit. HMM stands for Hidden Markov Model, CNN for ConvNet; when not specified, CNNs are 1D. pNorm is a particular non-linearity (Zhang et al., 2014). We also report extra information (besides word transcriptions) which might be used by each system, including speaker adaptation, or any other domain-specific data.

Paper	Acoustic Model	Sub-word	Spkr Adapt.	Extra Resources
Panayotov et al. (2015)	HMM+DNN+pNorm	phone	fMLLR	phone lexicon
Amodei et al. (2016)	2D-CNN+RNN	*letter*	*none*	11.9Kh train set, Common Crawl LM
Peddinti et al. (2015b)	HMM+CNN	phone	iVectors	phone lexicon
Povey et al. (2016)	HMM+CNN	phone	iVectors	phone lexicon, phone LM, data augm.
Ko et al. (2015)	HMM+CNN+pNorm	phone	iVectors	phone lexicon, data augm.
this paper	GLU-CNN	*letter*	*none*	*none*

3.2 Comparison with other systems

In Table 3, we compare our system with several of the best systems on LibriSpeech reported in the literature. We highlighted the acoustic model architectures, as well as the type of underlying sub-word unit. Note that phone-based acoustic models output in general senones; senones are carefully selected through a complicated procedure involving a phonetic-context-based decision tree built from another GMM/HMM system. Phone-based systems also require an additional word lexicon which translates words into a sequence of phones. Most systems also perform speaker adaptation; iVectors compute a speaker embedding capturing both speaker and environment information (Xue et al., 2014), while fMMLR is a two-pass decoder technique which computes a speaker transform in the first pass (Gales and Woodland, 1996).

Deep Speech 2 (Amodei et al., 2016) is the system which is the most related to ours. In contrast to other systems which combine a Hidden Markov Model (HMM) with a ConvNet, Deep Speech 2 is a standalone neural network. In contrast to our system, Deep Speech 2 embarks a more complicated acoustic model composed of a ConvNet and a Recurrent Neural Network (RNN), while our system is a simple ConvNet. Both *Deep Speech 2* and our system rely on letters for acoustic modeling, alleviating the need of a phone-based word lexicon. Deep Speech 2 relies on a lot of speech data (combined with a very large 5-gram language model) to make the letter-based approach competitive, while we limited ourselves to the available data in the LibriSpeech benchmark.

In Table 4, we report a comparison in WER performance for all systems introduced in Table 3. Our system is very competitive with existing approaches. Deep Speech 2 – which is also a letter-based system – is outperformed on clean data, even though our system has been trained with an order of magnitude less data. We report also the WER with no decoder, that is taking the raw output of the neural network, with no alterations. The Gated ConvNet appears very good at modeling true words.

Table 4: Comparison in WER of our model with other systems on LibriSpeech.

	test-clean	test-other
(Panayotov et al., 2015)	5.5	14.0
(Amodei et al., 2016)	5.3	13.3
(Peddinti et al., 2015b)	4.8	-
(Povey et al., 2016)	4.3	-
(Ko et al., 2015)	-	12.5
this paper	4.8	14.5
this paper (*no decoder*)	6.7	20.8

Using a single GPU (and no utterance batching), our High Dropout Gated ConvNet goes over the clean (5.4h) and other (5.1h) test sets in 4min26s and 4min43s, respectively. The decoder runs over the clean and other sets in 3min56s and 30min5s, using only one CPU thread – which (considering the decoder alone) corresponds to a .01 and 0.1 Real Time Factor (RTF), respectively.

4. Conclusion

We have introduced a simple end-to-end automatic speech recognition system, which combines a large (208M parameters) but efficient ConvNet acoustic model, an easy sequence criterion which can infer the segmentation, and a simple beam-search decoder. The decoding results are competitive on the LibriSpeech corpus (4.8% WER dev-clean). Our approach breaks free from HMM/GMM pre-training and forced alignment, as well as not being as computationally intensive as RNN-based approaches (Amodei et al., 2016). We based all our work on a publicly available (free) dataset, all of which should make it easier to reproduce. Further work should include leveraging speaker identity, training from the raw waveform, data augmentation, training with more data, and better language models.

References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning (ICML)*, pages 173–182, 2016.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949, 2016.

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In

International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 49–52, 1986.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, 2016.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv:1609.03193*, 2016.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional nets. In *International Conference on Machine Learning (ICML)*, 2017.

Mark J. F. Gales and Phil C. Woodland. Mean and variance adaptation within the MLLR framework. *Computer Speech and Language*, 10(4):249–264, 1996.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017.

Matthew Gibson and Thomas Hain. Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition. In *Interspeech*, pages 2406—-2409, 2006.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 369–376. ACM, 2006.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *arXiv:1412.5567*, 2014.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified kneser-ney language model estimation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 690–696, 2013.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 29(6):82–97, 2012.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Interspeech*, 2015.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, pages 255–257, 1995.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015.

Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *Transactions on Audio, Speech, and Language Processing*, 20(1): 14–22, 2012.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an ASR corpus based on public domain audio books. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2013.

Vijayaditya Peddinti, Guoguo Chen, Vimal Manohar, Tom Ko, Daniel Povey, and Sanjeev Khudanpur. JHU ASpIRE system: Robust LVCSR with TDNNs, iVector adaptation, and RNN-LMs. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015a.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*, 2015b.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *Interspeech*, pages 2751–2755, 2016.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 901–909. 2016.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using I-Vectors. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 55–59, 2013.

George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny. The IBM 2015 english conversational telephone speech recognition system. *arXiv:1505.05899*, 2015.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv:1703.02136*, 2017.

Andrew Senior, Georg Heigold, Michiel Bacchiani, and Hank Liao. GMM-free DNN training. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5639–5643, 2014.

Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for LVCSR. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4955–4959, 2016.

Hagen Soltau, George Saon, and Tara N. Sainath. Joint training of convolutional and non-convolutional neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5572–5576, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(Jun):1929–1958, 2014.

Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. In *International Conference on Spoken Language Processing (ICSLP)*, volume 94, pages 2143–2146, 1994.

Hao Tang, Liang Lu, Lingpeng Kong, Kevin Gimpel, Karen Livescu, Chris Dyer, Noah A. Smith, and Steve Renals. End-to-end neural segmental models for speech recognition. 08 2017.

Alex Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1(1):39–46, 1989.

Philip C. Woodland and Steve J. Young. The HTK tied-state continuous speech recogniser. In *Eurospeech*, 1993.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. The Microsoft 2016 conversational speech recognition system. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5255–5259, 2017.

Shaofei Xue, Ossama Abdel-Hamid, Hui Jiang, Li-Rong Dai, and Qingfeng Liu. Fast adaptation of deep neural network based on discriminant codes for speech recognition. *Transactions on Audio, Speech and Language Processing*, 22(12):1713–1725, 2014.

Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 215–219, 2014.