Lexicon-Free Conversational Speech Recognition with Neural Networks

Andrew L. Maas[*], **Ziang Xie**[*], **Dan Jurafsky**, **Andrew Y. Ng**
Stanford University
Stanford, CA 94305, USA
{amaas, zxie, ang}@cs.stanford.edu, jurafsky@stanford.edu

Abstract

We present an approach to speech recognition that uses only a neural network to map acoustic input to characters, a character-level language model, and a beam search decoding procedure. This approach eliminates much of the complex infrastructure of modern speech recognition systems, making it possible to directly train a speech recognizer using errors generated by spoken language understanding tasks. The system naturally handles out of vocabulary words and spoken word fragments. We demonstrate our approach using the challenging Switchboard telephone conversation transcription task, achieving a word error rate competitive with existing baseline systems. To our knowledge, this is the first entirely neural-network-based system to achieve strong speech transcription results on a conversational speech task. We analyze qualitative differences between transcriptions produced by our lexicon-free approach and transcriptions produced by a standard speech recognition system. Finally, we evaluate the impact of large context neural network character language models as compared to standard n-gram models within our framework.

1 Introduction

Users increasingly interact with natural language understanding systems via conversational speech interfaces. Google Now, Microsoft Cortana, and Apple Siri are all systems which rely on *spoken language understanding* (SLU), where transcribing speech is a single step within a larger system. Building such systems is difficult because spontaneous, conversational speech naturally contains repetitions, disfluencies, partial words, and out of vocabulary (OOV) words (De Mori et al., 2008; Huang et al., 2001). Moreover, SLU systems must be robust to transcription errors, which can be quite high depending on the task and domain.

Modern systems for large vocabulary continuous speech recognition (LVCSR) use hidden Markov models (HMMs) to handle sequence processing, word-level language models, and a pronunciation lexicon to map words into phonetic pronunciations (Saon and Chien, 2012). Traditional systems use Gaussian mixture models (GMMs) to build a mapping from sub-phonetic states to audio input features. The resulting speech recognition system contains many sub-components, linguistic assumptions, and typically over ten thousand lines of source code. Within the past few years LVCSR systems improved by replacing GMMs with deep neural networks (DNNs) (Dahl et al., 2011; Hinton et al., 2012), drawing on early work on with hybrid GMM-NN architectures (Bourlard and Morgan, 1993). Both HMM-GMM and HMM-DNN systems remain difficult to build, and nearly impossible to efficiently optimize for downstream SLU tasks. As a result, SLU researchers typically operate on an n-best list of possible transcriptions and treat the LVCSR system as a black box.

Recently Graves and Jaitly (2014) demonstrated an approach to LVCSR using a neural network trained with the connectionist temporal classification (CTC) loss function (Graves et al., 2006). Us-

[*]Authors contributed equally.

ing the CTC loss function the authors built a neural network which directly maps audio input features to a sequence of characters. By re-ranking word-level n-best lists generated from an HMM-DNN system the authors obtained competitive results on the Wall Street Journal corpus.

Our work builds upon the foundation introduced by Graves and Jaitly (2014). Rather than reasoning at the word level, we train and decode our system by reasoning entirely at the character-level. By reasoning over characters we eliminate the need for a lexicon, and enable transcribing new words, fragments, and disfluencies. We train a deep bidirectional recurrent neural network (DBRNN) to directly map acoustic input to characters using the CTC loss function introduced by Graves and Jaitly (2014). We are able to efficiently and accurately perform transcription using only our DBRNN and a character-level language model (CLM), whereas previous work relied on n-best lists from a baseline HMM-DNN system. On the challenging Switchboard telephone conversation transcription task, our approach achieves a word error rate competitive with existing baseline HMM-GMM systems. To our knowledge, this is the first entirely neural-network-based system to achieve strong speech transcription results on a conversational speech task.

Section 2 reviews the CTC loss function and describes the neural network architecture we use. Section 3 presents our approach to efficiently perform first-pass decoding using a neural network for character probabilities and a character language model. Section 4 presents experiments on the Switchboard corpus to compare our approach to existing LVCSR systems, and evaluates the impact of different language models. In Section 5, we offer insight on how the CTC-trained system performs speech recognition as compared to a standard HMM-GMM model, and finally conclude in Section 6.

2 Model

We address the complete LVCSR problem. Our system trains on utterances which are labeled by word-level transcriptions and contain no indication of when words occur within an utterance. Our approach consists of two neural networks which we integrate during a beam search decoding procedure. Our first neural network, a DBRNN, maps acoustic input features to a probability distribution over characters at each time step. Our second system component is a neural network character language model. Neural network CLMs enable us to leverage high order n-gram contexts without dramatically increasing the number of free parameters in our language model. To facilitate further work with our approach we make our source code publicly available. [1]

2.1 Connectionist Temporal Classification

We train neural networks using the CTC loss function to do maximum likelihood training of letter sequences given acoustic features as input. This is a direct, discriminative approach to building a speech recognition system in contrast to the generative, noisy-channel approach which motivates HMM-based speech recognition systems. Our application of the CTC loss function follows the approach introduced by Graves and Jaitly (2014), but we restate the approach here for completeness.

CTC is a generic loss function to train systems on sequence problems where the alignment between the input and output sequence are unknown. CTC accounts for time warping of the output sequence relative to the input sequence, but does not model possible re-orderings. Re-ordering is a problem in machine translation, but is not an issue when working with speech recognition – our transcripts provide the exact ordering in which words occur in the input audio.

Given an input sequence X of length T, CTC assumes the probability of a length T character sequence C is given by,

$$p(C|X) = \prod_{t=1}^{T} p(c_t|X). \tag{1}$$

This assumes that character outputs at each timestep are conditionally independent given the input. The distribution $p(c_t|X)$ is the output of some predictive model.

CTC assumes our ground truth transcript is a character sequence W with length τ where $\tau \leq T$. As a result, we need a way to construct possibly shorter output sequences from our length T sequence of

[1] Available at: `deeplearning.stanford.edu/lexfree`

character probabilities. The CTC *collapsing function* achieves this by introducing a special *blank* symbol, which we denote using "_", and collapsing any repeating characters in the original length T output. This output symbol contains the notion of *junk* or *other* so as to not produce a character in the final output hypothesis. Our transcripts W come from some set of symbols ζ' but we reason over $\zeta = \zeta' \cup \_$.

We denote the collapsing function by $\kappa(\cdot)$ which takes an input string and produces the unique collapsed version of that string. As an example, here are the set of strings Z of length $T = 3$ such that $\kappa(z) = \texttt{hi},\ \forall z \in Z$:

$$Z = \{\texttt{hhi}, \texttt{hii}, \texttt{\_hi}, \texttt{h\_i}, \texttt{hi\_}\}.$$

There are a large number of possible length T sequences corresponding to a final length τ transcript hypothesis. The CTC objective function $\mathcal{L}_{\text{CTC}}(X, W)$ is a likelihood of the correct final transcript W which requires integrating over the probabilities of all length T character sequences $C_W = \{C : \kappa(C) = W\}$ consistent with W after applying the collapsing function,

$$\begin{aligned} \mathcal{L}_{\text{CTC}}(X, W) &= \sum_{C_W} p(C|X) \\ &= \sum_{C_W} \prod_{t=1}^{T} p(c_t|X). \end{aligned} \tag{2}$$

Using a dynamic programming approach we can exactly compute this loss function efficiently as well as its gradient with respect to our probabilities $p(c_t|X)$.

2.2 Deep Bi-Directional Recurrent Neural Networks

Our loss function requires at each time t a probability distribution $p(c|x_t)$ over characters c given input features x_t. We model this distribution using a DBRNN because it provides an expressive model which explicitly accounts for the sequential relationships that should exist in our task. Moreover, the DBRNN is a relatively straightforward neural network architecture to specify, and allows us to learn parameters from data rather than more explicitly specifying how to convert audio features into characters. Figure 1 shows a DBRNN with two hidden layers.

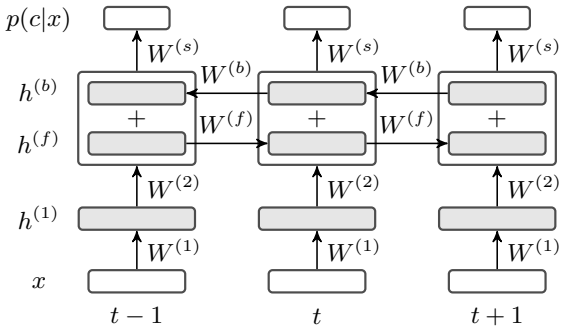

Figure 1: Deep bi-directional recurrent neural network to map input audio features X to a distribution $p(c|x_t)$ over output characters at each timestep t. The network contains two hidden layers with the second layer having bi-directional temporal recurrence.

A DBRNN computes the distribution $p(c|x_t)$ using a series of hidden layers followed by an output layer. Given an input vector x_t the first hidden layer activations are a vector computed as,

$$h^{(1)} = \sigma(W^{(1)T} x_t + b^{(1)}), \tag{3}$$

where the matrix $W^{(1)}$ and vector $b^{(1)}$ are the weight matrix and bias vector. The function $\sigma(\cdot)$ is a point-wise nonlinearity. We use $\sigma(z) = \min(\max(z, 0), \mu)$. This is a rectified linear activation function clipped to a maximum possible activation of μ to prevent overflow. Rectified linear hidden units have been show to work well in general for deep neural networks, as well as for acoustic modeling of speech data (Glorot et al., 2011; Zeiler et al., 2013; Dahl et al., 2013; Maas et al., 2013)

We select a single hidden layer j of the network to have temporal connections. Our temporal hidden layer representation $h^{(j)}$ is the sum of two partial hidden layer representations,

$$h_t^{(j)} = h_t^{(f)} + h_t^{(b)}. \tag{4}$$

The representation $h^{(f)}$ uses a weight matrix $W^{(f)}$ to propagate information forwards in time. Similarly, the representation $h^{(b)}$ propagates information backwards in time using a weight matrix $W^{(b)}$. These partial hidden representations both take input from the previous hidden layer $h^{(j-1)}$ using a weight

matrix $W^{(j)}$,

$$h_t^{(f)} = \sigma(W^{(j)T} h_t^{(j-1)} + W^{(f)T} h_{t-1}^{(f)} + b^{(j)}),$$
$$h_t^{(b)} = \sigma(W^{(j)T} h_t^{(j-1)} + W^{(b)T} h_{t+1}^{(b)} + b^{(j)}). \quad (5)$$

Note that the recurrent forward and backward hidden representations are computed entirely independently from each other. As with the other hidden layers of the network we use $\sigma(z) = \min(\max(z, 0), \mu)$.

All hidden layers aside from the first hidden layer and temporal hidden layer use a standard dense weight matrix and bias vector,

$$h^{(i)} = \sigma(W^{(i)T} h^{(i-1)} + b^{(i)}). \quad (6)$$

DBRNNs can have an arbitrary number of hidden layers, but we assume that only one hidden layer contains temporally recurrent connections.

The model outputs a distribution $p(c|x_t)$ over a set of possible characters ζ using a *softmax* output layer. We compute the softmax layer as,

$$p(c = c_k|x_t) = \frac{\exp(-(W_k^{(s)T} h^{(:)} + b_k^{(s)}))}{\sum_{j=1}^{|\zeta|} \exp(-(W_j^{(s)T} h^{(:)} + b_j^{(s)}))}, \quad (7)$$

where $W_k^{(s)}$ is the k'th column of the output weight matrix $W^{(s)}$ and $b_k^{(s)}$ is a scalar bias term. The vector $h^{(:)}$ is the hidden layer representation of the final hidden layer in our DBRNN.

We can directly compute a gradient for all weights and biases in the DBRNN with respect to the CTC loss function and apply batch gradient descent.

3 Decoding

Our decoding procedure integrates information from the DBRNN and language model to form a single cohesive estimate of the character sequence in a given utterance. For an input sequence X of length T our DBRNN produces a set of probabilities $p(c|x_t),\ t = 1, \ldots, T$. Again, the character probabilities are a categorical distribution over the symbol set ζ.

3.1 Decoding Without a Language Model

As a baseline, we use a simple, greedy approach to decoding the DBRNN outputs (Graves and Jaitly, 2014). The simplest form of decoding does not employ the language model and instead finds the highest probability character transcription given only the DBRNN outputs. This process selects a transcript hypothesis W^* by making a greedy approximation,

$$W^* = \arg\max_W p(W|X) \approx \kappa(\arg\max_C p(C|X))$$
$$= \kappa(\arg\max_C \prod_{t=1}^{T} p(c_t|X)). \quad (8)$$

This decoding procedure ignores the issue of many time-level character sequences mapping to the same final hypothesis, and instead considers only the most probable character at each point in time. Because our model assumes the character labels for each timestep are conditionally independent, C^* is simply the most probable character at each timestep in our DBRNN output. As a result, this decoding procedure is very fast to compute, requiring only time $O(T|\zeta|)$.

3.2 Beam Search Decoding

To decode while taking language model probabilities into account, we use a beam search to combine a character language model and the outputs of our DBRNN. This search-based decoding method does not make a greedy approximation and instead assigns probability to a final hypothesis by integrating over all character sequences consistent with the hypothesis under our collapsing function $\kappa(\cdot)$. Algorithm 1 outlines our decoding procedure.

We note that our decoding procedure is significantly simpler, and in practice faster, than previous decoding procedures applied to CTC models. This is due to reasoning at the character level without a lexicon so as to not introduce difficult multi-level constraints to obey during the decoding search procedure. While a softmax over words is typically the bottleneck in neural network language models, a softmax over possible characters is comparatively cheap to compute. Our character language model is applied at every time step, while word models can only be applied when we consider adding a space or by computing the likelihood of a sequence being the prefix of a word in the lexicon (Graves and Jaitly, 2014). Additionally, our lexicon-free approach re-

Algorithm 1 Beam Search Decoding: Given the likelihoods from our DBRNN and our character language model, for each time step t and for each string s in our current previous hypothesis set Z_{t-1}, we consider extending s with a new character. Blanks and repeat characters with no separating blank are handled separately. For all other character extensions, we apply our character language model when computing the probability of s. We initialize Z_0 with the empty string $\varnothing$. Notation: ζ': character set excluding "_", $s + c$: concatenation of character c to string s, $|s|$: length of s, $p_{\text{b}}(c|x_{1:t})$ and $p_{\text{nb}}(c|x_{1:t})$: probability of s ending and not ending in blank conditioned on input up to time t, $p_{\text{tot}}(c|x_{1:t})$: $p_{\text{b}}(c|x_{1:t}) + p_{\text{nb}}(c|x_{1:t})$

Inputs CTC likelihoods $p_{\text{ctc}}(c|x_t)$, character language model $p_{\text{clm}}(c|s)$
Parameters language model weight α, insertion bonus β, beam width k
Initialize $Z_0 \leftarrow \{\varnothing\}$, $p_{\text{b}}(\varnothing|x_{1:0}) \leftarrow 1$, $p_{\text{nb}}(\varnothing|x_{1:0}) \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  $Z_t \leftarrow \{\}$
  for s **in** Z_{t-1} **do**
    $p_{\text{b}}(s|x_{1:t}) \leftarrow p_{\text{ctc}}(\_|x_t)p_{\text{tot}}(s|x_{1:t-1})$ ▷ Handle blanks
    $p_{\text{nb}}(s|x_{1:t}) \leftarrow p_{\text{ctc}}(c|x_t)p_{\text{nb}}(s|x_{1:t-1})$ ▷ Handle repeat character collapsing
    Add s to Z_t
    for c **in** ζ' **do**
      $s^+ \leftarrow s + c$
      if $c \neq s_{t-1}$ **then**
        $p_{\text{nb}}(s^+|x_{1:t}) \leftarrow p_{\text{ctc}}(c|x_t)p_{\text{clm}}(c|s)^\alpha p_{\text{tot}}(c|x_{1:t-1})$
      else
        $p_{\text{nb}}(s^+|x_{1:t}) \leftarrow p_{\text{ctc}}(c|x_t)p_{\text{clm}}(c|s)^\alpha p_{\text{b}}(c|x_{1:t-1})$ ▷ Repeat characters have "_" between
      end if
      Add s^+ to Z_t
    end for
  end for
  $Z_t \leftarrow k$ most probable s by $p_{\text{tot}}(s|x_{1:t})|s|^\beta$ in Z_t ▷ Apply beam
end for
Return $\arg\max_{s \in Z_t} p_{\text{tot}}(s|x_{1:T})|s|^\beta$

moves the difficulties of handling OOV words during decoding, which is typically a troublesome issue in speech recognition systems.

4 Experiments

We perform LVCSR experiments on the 300 hour Switchboard conversational telephone speech corpus (LDC97S62). Switchboard utterances are taken from approximately 2,400 conversations among 543 speakers. Each pair of speakers had never met, and converse no more than once about a given topic chosen randomly from a set of 50 possible topics. Utterances exhibit many rich, complex phenomena that make spoken language understanding difficult. Table 2 shows example transcripts from the corpus.

For evaluation, we report word error rate (WER) and character error rate (CER) on the HUB5 Eval2000 dataset (LDC2002S09). This test set consists of two subsets, Switchboard and CallHome. The CallHome subset represents a mismatched test condition as it was collected from phone conversations among family and friends rather than strangers directed to discuss a particular topic. The mismatch makes the CallHome subset quite difficult overall. The Switchboard evaluation subset is substantially easier, and represents a better match of test data to our training corpus. We report WER and CER on the test set as a whole, and additionally report WER for each subset individually.

4.1 Baseline Systems

We build two baseline LVCSR systems to compare our approach to standard HMM-based approaches.

Method	CER	EV	CH	SWBD
HMM-GMM	23.0	29.0	36.1	21.7
HMM-DNN	17.6	21.2	27.1	15.1
HMM-SHF	NR	NR	NR	12.4
CTC no LM	27.7	47.1	56.1	38.0
CTC+5-gram	25.7	39.0	47.0	30.8
CTC+7-gram	24.7	35.9	43.8	27.8
CTC+NN-1	24.5	32.3	41.1	23.4
CTC+NN-3	24.0	30.9	39.9	21.8
CTC+RNN	24.9	33.0	41.7	24.2
CTC+RNN-3	24.7	30.8	40.2	21.4

Table 1: Character error rate (CER) and word error rate results on the Eval2000 test set. We report word error rates on the full test set (EV) which consists of the Switchboard (SWBD) and CallHome (CH) subsets. As baseline systems we use an HMM-GMM system and HMM-DNN system. We evaluate our DBRNN trained using CTC by decoding with several character-level language models: 5-gram, 7-gram, densely connected neural networks with 1 and 3 hidden layers (NN-1, and NN-3), as well as recurrent neural networks s with 1 and 3 hidden layers. We additionally include results from a state-of-the-art HMM-based system (HMM-DNN-SHF) which does not report performance on all metrics we evaluate (NR).

First, we build an HMM-GMM system using the Kaldi open-source toolkit[2] (Povey et al., 2011). The baseline recognizer has 8,986 sub-phone states and 200K Gaussians trained using maximum likelihood. Input features are speaker-adapted MFCCs. Overall, the baseline GMM system setup largely follows the existing `s5b` Kaldi recipe, and we defer to previous work for details (Vesely et al., 2013).

We additionally built an HMM-DNN system by training a DNN acoustic model using maximum likelihood on the alignments produced by our HMM-GMM system. The DNN consists of five hidden layers, each with 2,048 hidden units, for a total of approximately 36 million (M) free parameters in the acoustic model.

Both baseline systems use a bigram language model built from the 3M words in the Switchboard transcripts interpolated with a second bigram language model built from 11M words on the Fisher English Part 1 transcripts (LDC2004T19). Both LMs are trained using interpolated Kneser-Ney smoothing. For context we also include WER results from a state-of-the-art HMM-DNN system built with quinphone phonetic context and Hessian-free sequence-discriminative training (Sainath et al., 2014).

[2] `http://kaldi.sf.net`

4.2 DBRNN Training

We train a DBRNN using the CTC loss function on the entire 300hr training corpus. The input features to the DBRNN at each timestep are MFCCs with context window of ± 10 frames. The DBRNN has 5 hidden layers with the third containing recurrent connections. All layers have 1824 hidden units, giving about 20M trainable parameters. In preliminary experiments we found that choosing the middle hidden layer to have recurrent connections led to the best results.

The output symbol set ζ consists of 33 characters including the special blank character. Note that because speech recognition transcriptions do not contain proper casing or punctuation, we exclude capital letters and punctuation marks with the exception of "–", which denotes a partial word fragment, and "′", as used in contractions such as "`can′t`."

We train the DBRNN from random initial parameters using the gradient-based Nesterov's accelerated gradient (NAG) algorithm as this technique is sometimes beneficial as compared with standard stochastic gradient descent for deep recurrent neural network training (Sutskever et al., 2013). The NAG algorithm uses a step size of 10^{-5} and a momentum of 0.95. After each epoch we divide the learning rate by 1.3. Training for 10 epochs on a single GTX 570 GPU takes approximately one week.

4.3 Character Language Model Training

The Switchboard corpus transcripts alone are too small to build CLMs which accurately model general orthography in English. To learn how to spell words more generally we train our CLMs using a corpus of 31 billion words gathered from the web (Heafield et al., 2013). Our language models use sentence start and end tokens, `<s>` and `</s>`, as

well as a <null> token for cases when our context window extends past the start of a sentence.

We build 5-gram and 7-gram CLMs with modified Kneser-Ney smoothing using the KenLM toolkit (Heafield et al., 2013). Building traditional n-gram CLMs is for $n > 7$ becomes increasingly difficult as the model free parameters and memory footprint become unwieldy. Our 7-gram CLM is already 21GB; we were not able to build higher order n-gram models to compare against our neural network CLMs.

Following work illustrating the effectiveness of neural network CLMs (Sutskever et al., 2011) and word-level LMs for speech recognition (Mikolov et al., 2010), we train and evaluate two variants of neural network CLMs: standard feedfoward deep neural networks (DNNs) and a recurrent neural network (RNN). The RNN CLM takes one character at a time as input, while the non-recurrent CLM networks use a context window of 19 characters. All neural network CLMs use the rectified linear activation function, and the layer sizes are selected such that each has about 5M parameters (20MB).

The DNN models are trained using standard backpropagation using Nesterov's accelerated gradient with a learning rate of 0.01 and momentum of 0.95 and a batch size of 512. The RNN is trained using backpropagation through time with a learning rate of 0.001 and batches of 128 utterances. For both model types we halve the learning rate after each epoch. The DNN models were trained for 10 epochs, and the RNN models for 5 epochs.

All neural network CLMs were trained using a combination of the Switchboard and Fisher training transcripts which in total contain approximately 23M words. We also performed experiments with CLMs trained from a large corpus of web text, but found these CLMs to perform no better than transcript-derived CLMs for our task.

4.4 Results

After training the DBRNN and CLMs we run decoding on the Eval2000 test set to obtain CER and WER results. For all experiments using a CLM we use our beam search decoding algorithm with $\alpha = 1.25$, $\beta = 1.5$ and a beam width of 100. We found that larger beam widths did not significantly improve performance. Table 1 shows results for the DBRNN as well as baseline systems.

The DBRNN performs best with the 3 hidden layer DNN CLM. This DBRNN+NN-3 attains both CER and WER performance comparable to the HMM-GMM baseline system, albeit substantially below the HMM-DNN system. Neural networks provide a clear gain as compared to standard n-gram models when used for DBRNN decoding, although the RNN CLM does not produce any gain over the best DNN CLM.

Without a language model the greedy DBRNN decoding procedure loses relatively little in terms of CER as compared with the DBRNN+NN-3 model. However, this 3% difference in CER translates to a 16% gap in WER on the full Eval2000 test set. Generally, we observe that small CER differences translate to large WER differences. In terms of character-level performance it appears as if the DBRNN alone performs well using only acoustic input data. Adding a CLM yields only a small CER improvement, but guides proper spelling of words to produce a large reduction in WER.

5 Analysis

To better see how the DBRNN performs transcription we show the output probabilities $p(c|x)$ for an example utterance in Figure 2. The model tends to output mostly blank characters and only spike long enough for a character to be the most likely symbol for a few frames at a time. The dominance of the blank class is not forced, but rather learned by the DBRNN during training. We hypothesize that this spiking behavior results in more stable results as the DBRNN only produces a character when its confidence of seeing that character rises above a certain threshold. Note that this a dramatic contrast to HMM-based LVCSR systems, which, due to the nature of generative models, attempt to explain almost all timesteps as belonging to a phonetic substate.

Next, we qualitatively compare the DBRNN and HMM-GMM system outputs to better understand how the DBRNN approach might interact with SLU systems. This comparison is especially interesting because our best DBRNN system and the HMM-GMM system have comparable WERs, removing the confound of overall quality when comparing hypotheses. Table 2 shows example test set utterances along with transcription hypotheses from the HMM-

#	Method	Transcription
(1)	Truth	yeah i went into the i do not know what you think of *fidelity* but
	HMM-GMM	yeah when the i don't know what you think of fidel it even them
	CTC+CLM	yeah i went to i don't know what you think of fidelity but um
(2)	Truth	no no speaking of weather do you carry a altimeter slash *barometer*
	HMM-GMM	no i'm not all being the weather do you uh carry a uh helped emitters last brahms her
	CTC+CLM	no no beating of whether do you uh carry a uh a time or less barometer
(3)	Truth	i would ima- well yeah it is i know you are able to stay home with them
	HMM-GMM	i would amount well yeah it is i know um you're able to stay home with them
	CTC+CLM	i would ima- well yeah it is i know uh you're able to stay home with them

Table 2: Example test set utterances with a ground truth transcription and hypotheses from our method (CTC+CLM) and a baseline HMM-GMM system of comparable overall WER. The words *fidelity* and *barometer* are not in the lexicon of the HMM-GMM system.

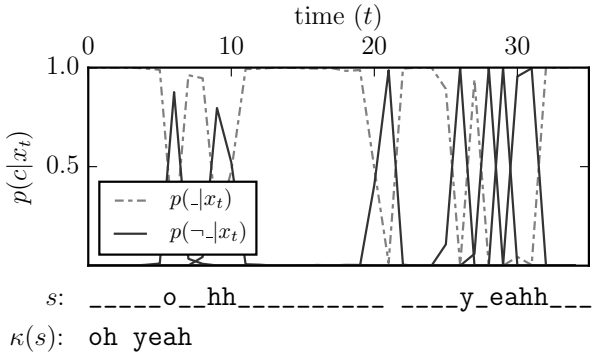

Figure 2: DBRNN character probabilities over time for a single utterance along with the per-frame most likely character string s and the collapsed output $\kappa(s)$. Due to space constraints we only show a distinction in line type between the blank symbol _ and non-blank symbols.

GMM and DBRNN+NN-3 systems.

The DBRNN sometimes correctly transcribes OOV words with respect to our audio training corpus. We find that OOVs tend to trigger clusters of errors in the HMM-GMM system, an observation that has been systematically explored in previous work (Goldwater et al., 2010). As shown in example utterance (3), HMM-GMM errors can introduce word substitution errors which may alter meaning whereas the DBRNN system outputs word fragments or non-words which are phonetically similar and may be useful input features for SLU systems. Unfortunately the Eval2000 test set does not offer a rich set of utterances containing OOVs or fragments to perform a deeper analysis. The HMM-GMM and best DBRNN system achieve identical WERs on the subset of test utterances containing OOVs and the subset of test utterances containing fragments.

Finally, we quantitatively compare how character probabilities from the DBRNN align with phonetic segments from the HMM-GMM system. We generate HMM-GMM forced alignments on a large sample of the training set, and separate utterances into monophone segments. For each monophone, we compute the average character probabilities from the DBRNN by aligning the beginning of each monophone segment, treating it as time 0. We measure time using feature frames rather than seconds. Figure 3 shows character probabilities over time for the phones *k*, *sh*, *w*, and *uw*.

Although the CTC model does not explicitly compute a forced alignment as part of training, we see significant rises in character probabilities corresponding to particular phones during HMM-GMM-aligned monophone segments. This indicates that the CTC model automatically learns a reasonable alignment of characters to the audio. Generally, the CTC model tends to produce character spikes towards the beginning of monophone segments. This is especially evident in plosive consonants such as *k* and *t*. For liquids and glides (*r*, *l*, *w*, *y*), the CTC model does not produce characters until later in the monophone segment. For vowels the CTC character

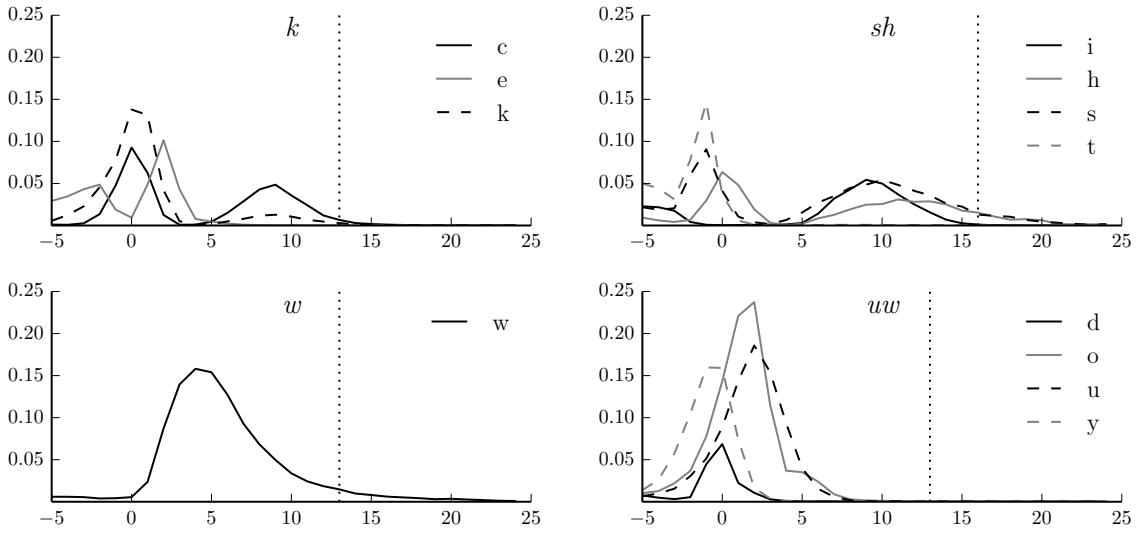

Figure 3: Character probabilities from the CTC-trained neural network averaged over monophone segments created by a forced alignment of the HMM-GMM system. Time is measured in frames, with 0 indicating the start of the monophone segment. The vertical dotted line indicates the average duration of the monophone segment. We show only characters with non-trivial probability for each phone while excluding the blank and space symbols.

probabilities generally rise slightly later in the phone segment as compared to consonants. This may occur to avoid the large contextual variations in vowel pronunciations at phone boundaries. For certain consonants we observe CTC probability spikes before the monophone segment begins, as is the case for *sh*. The probabilities for *sh* additionally exhibit multiple modes, suggesting that CTC may learn different behaviors for the two common spellings of the sibilant *sh*: the letter sequence "sh" and the letter sequence "ti".

6 Conclusion

We presented an LVCSR system consisting of two neural networks integrated via beam search decoding that matches the performance of an HMM-GMM system on the challenging Switchboard corpus. We built on the foundation of Graves and Jaitly (2014) to vastly reduce the overall complexity required for LVCSR systems. Our method yields a complete first-pass LVCSR system with about 1,000 lines of code — roughly an order of magnitude less than high performance HMM-GMM systems. Operating entirely at the character level yields a system which does not require assumptions about a lexicon or pronunciation dictionary, instead learning orthography and phonics directly from data. We hope the simplicity of our approach will facilitate future research in improving LVCSR with CTC-based systems and jointly training LVCSR systems for SLU tasks. DNNs have already shown great results as acoustic models in HMM-DNN systems. We free the neural network from its complex HMM infrastructure, which we view as the first step towards the next wave of advances in speech recognition and language understanding.

Acknowledgments

We thank Awni Hannun for his contributions to the software used for experiments in this work. We also thank Peng Qi and Thang Luong for insightful discussions, and Kenneth Heafield for help with the KenLM toolkit. Our work with HMM-GMM systems was possible thanks to the Kaldi toolkit and its contributors. Some of the GPUs used in this work were donated by the NVIDIA Corporation. AM was supported as an NSF IGERT Traineeship Recipient under Award 0801700. ZX was supported by an NDSEG Graduate Fellowship.

References

H. Bourlard and N. Morgan. 1993. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA.

G. E. Dahl, D. Yu, and L. Deng. 2011. Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In *Proc. ICASSP*.

G. E. Dahl, T. N. Sainath, and G. E. Hinton. 2013. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout. In *ICASSP*.

R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. 2008. Spoken language understanding. *Signal Processing Magazine, IEEE*, 25(3):50–58.

X. Glorot, A. Bordes, and Y Bengio. 2011. Deep Sparse Rectifier Networks. In *AISTATS*, pages 315–323.

S. Goldwater, D. Jurafsky, and C. Manning. 2010. Which Words are Hard to Recognize? Prosodic, Lexical, and Disfluency Factors That Increase Speech Recognition Error Rates. *Speech Communications*, 52:181–200.

A. Graves and N. Jaitly. 2014. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *ICML*.

A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376. ACM.

K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL-HLT*, pages 690–696, Sofia, Bulgaria.

G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(November):82–97.

X. Huang, A. Acero, H.-W. Hon, et al. 2001. *Spoken language processing*, volume 18. Prentice Hall Englewood Cliffs.

A. Maas, A. Hannun, and A. Ng. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, K. Veselý, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, and G. Stemmer. 2011. The kaldi speech recognition toolkit. In *ASRU*.

T. N. Sainath, I. Chung, B. Ramabhadran, M. Picheny, J. Gunnels, B. Kingsbury, G. Saon, V. Austel, and U. Chaudhari. 2014. Parallel deep neural network training for lvcsr tasks using blue gene/q. In *INTERSPEECH*.

G. Saon and J. Chien. 2012. Large-vocabulary continuous speech recognition systems: A look at some recent advances. *IEEE Signal Processing Magazine*, 29(6):18–33.

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating text with recurrent neural networks. In *ICML*, pages 1017–1024.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. 2013. On the Importance of Momentum and Initialization in Deep Learning. In *ICML*.

K. Vesely, A. Ghoshal, L. Burget, and D. Povey. 2013. Sequence-discriminative training of deep neural networks. In *Interspeech*.

M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton. 2013. On Rectified Linear Units for Speech Processing. In *ICASSP*.