arXiv:1507.08240v1 [cs.CL] 29 Jul 2015

EESEN: END-TO-END SPEECH RECOGNITION USING DEEP RNN MODELS AND WFST-BASED DECODING

Yajie Miao, Mohammad Gowayyed, Florian Metze

Language Technologies Institute, School of Computer Science, Carnegie Mellon University

ABSTRACT

The performance of automatic speech recognition (ASR) has improved tremendously due to the application of deep neural networks (DNNs). Despite this progress, building a new ASR system remains a challenging task, requiring various resources, multiple training stages and significant expertise. This paper presents our Eesen framework which drastically simplifies the existing pipeline to build state-of-the-art ASR systems. Acoustic modeling in Eesen involves learning a single recurrent neural network (RNN) predicting context-independent targets (phonemes or characters). To remove the need for pre-generated frame labels, we adopt the connectionist temporal classification (CTC) objective function to infer the alignments between speech and label sequences. A distinctive feature of Eesen is a generalized decoding approach based on weighted finite-state transducers (WFSTs), which enables the efficient incorporation of lexicons and language models into CTC decoding. Experiments show that compared with the standard hybrid DNN systems, Eesen achieves comparable word error rates (WERs), while at the same time speeding up decoding significantly.

***Index Terms*—** Recurrent neural network, connectionist temporal classification, end-to-end ASR

1. INTRODUCTION

Automatic speech recognition (ASR) has traditionally leveraged the hidden Markov model/Gaussian mixture model (HMM/GMM) paradigm for acoustic modeling. HMMs act to normalize the temporal variability, whereas GMMs compute the emission probabilities of HMM states. In recent years, the performance of ASR has been improved dramatically by the introduction of deep neural networks (DNNs) as acoustic models [1, 2, 3]. In the *hybrid* HMM/DNN approach, DNNs are used to classify speech frames into clustered context-dependent (CD) states (i.e., senones). On a variety of ASR tasks, DNN models have shown significant gains over the GMM models. Despite these advances, building a state-of-the-art ASR system remains a complicated, expertise-intensive task. First, acoustic modeling typically requires various resources such as dictionaries and phonetic questions. Under certain conditions (e.g., in low-resource languages), these resources may be unavailable, which restricts or delays the deployment of ASR. Second, in the hybrid approach, training of DNNs still relies on GMM models to obtain (initial) frame-level labels. Building GMM models normally goes through multiple stages (e.g., CI phone, CD states, etc.), and every stage involves different feature processing techniques (e.g., LDA, fMLLR, etc.). Third, the development of ASR systems highly relies on ASR experts to determine the optimal configurations of a multitude of hyper-parameters, for instance, the number of senones and Gaussians in the GMM models.

Previous work has made various attempts to reduce the complexity of ASR. In [4, 5], researchers propose to flat-start DNNs and thus get ride of GMM models. However, this GMM-free approach still requires iterative procedures such as generating forced alignments and decision trees. Meanwhile, another line of work [6, 7, 8, 9, 10] has focused on end-to-end ASR, i.e., modeling the mapping between speech and labels (words, phonemes, etc.) directly without any intermediate components (e.g., GMMs). On this aspect, Graves *et al.* [11] introduce the *connectionist temporal classification* (CTC) objective function to infer speech-label alignments automatically. This CTC technique is further investigated in [6, 7, 8, 12] on large-scale acoustic modeling tasks. Although showing promising results, research on end-to-end ASR faces two major obstacles. First, it is challenging to incorporate lexicons and language models into decoding. When decoding CTC-trained models, past work [6, 8, 10] has successfully constrained search paths with lexicons. However, how to integrate *word-level* language models efficiently still is an unanswered question [10]. Second, the community lacks a shared experimental platform for the purpose of benchmarking. End-to-end systems described in the literature differ not only in their model architectures but also in their decoding methods. For example, [6] and [8] adopt two distinct versions of beam search for decoding CTC models. These setup variations hamper rigorous comparisons not only across end-to-end systems, but also between the end-to-end and existing hybrid approaches.

In this paper, we resolve these issues by presenting and publicly releasing our Eesen framework. Acoustic modeling in Eesen is viewed as a sequence-to-sequence learning problem. We exploit deep recurrent neural networks (RNNs)

[13, 14, 15] as the acoustic models, and the Long Short-Term Memory (LSTM) units [16, 17, 18, 19] as the RNN building blocks. Using the CTC objective function, Eesen simplifies acoustic modeling into learning a single RNN over pairs of speech and context-independent (CI) label sequences. A distinctive feature of Eesen is a generalized decoding method based on weighted finite-state transducers (WFSTs). In this method, individual components (CTC labels, lexicons and language models) are encoded into WFSTs, and then composed into a comprehensive search graph. The WFST representation provides a convenient way of handling the CTC *blank* label and enabling beam search during decoding. Our experiments with the Wall Street Journal (WSJ) benchmark show that Eesen results in superior performance than the existing end-to-end ASR pipelines [6, 8]. The WERs of Eesen are on a par with strong hybrid HMM/DNN baselines. Moreover, the application of CI modeling targets allows Eesen to speed up decoding and reduce decoding memory usage. Eesen is released as an open-source project[1], and will undertake continuous expansion and optimization.

2. THE EESEN FRAMEWORK: MODEL TRAINING

Acoustic models in Eesen are deep bidirectional RNNs trained with the CTC objective function [11]. We describe the model structure in Section 2.1, and restate key points of CTC training in Section 2.2. Section 2.3 presents some practical considerations emerging from our GPU implementation.

2.1. Deep Bidirectional Recurrent Neural Networks

Compared to the standard feedforward networks, RNNs have the advantage of learning complex temporal dynamics on sequences. Given an input sequence $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, a recurrent layer computes the *forward* sequence of hidden states $\overrightarrow{\mathbf{H}} = (\overrightarrow{\mathbf{h}}_1, ..., \overrightarrow{\mathbf{h}}_T)$ by iterating from $t = 1$ to T:

$$\overrightarrow{\mathbf{h}}_t = \sigma(\overrightarrow{\mathbf{W}}_{hx}\mathbf{x}_t + \overrightarrow{\mathbf{W}}_{hh}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_h) \qquad (1)$$

where $\overrightarrow{\mathbf{W}}_{hx}$ is the input-to-hidden weight matrix, $\overrightarrow{\mathbf{W}}_{hh}$ is the hidden-to-hidden weight matrix. In addition to the inputs $\mathbf{x}_t$, the hidden activation $\mathbf{h}_{t-1}$ from the previous time step are fed to influence the hidden outputs at the current time step. In a bidirectional RNN, an additional recurrent layer computes the *backward* sequence of hidden outputs $\overleftarrow{\mathbf{H}}$ from $t = T$ to 1:

$$\overleftarrow{\mathbf{h}}_t = \sigma(\overleftarrow{\mathbf{W}}_{hx}\mathbf{x}_t + \overleftarrow{\mathbf{W}}_{hh}\overleftarrow{\mathbf{h}}_{t-1} + \overleftarrow{\mathbf{b}}_h) \qquad (2)$$

Our acoustic model is a deep architecture, in which we stack multiple bidirectional recurrent layers. At each frame t, the concatenation of the forward and backward hidden outputs $[\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]$ from the current layer are treated as inputs into the next recurrent layer.

[1]https://github.com/yajiemiao/eesen

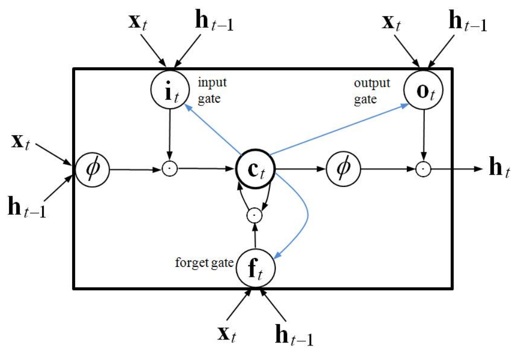

Fig. 1. A memory block of LSTM.

Learning of RNNs can be done using back-propagation through time (BPTT). In practice, training RNNs to learn long-term temporal dependency can be difficult due to the vanishing gradients problem [20]. To overcome this issue, we apply the LSTM units [16] as the building blocks of RNNs. LSTM contains memory cells with self-connections to store the temporal states of the network. Also, multiplicative gates are added to control the flow of information. Fig. 1 depicts the structure of the LSTM units we use. The blue curves represent peephole connections [21] that link the memory cells to the gates to learn precise timing of the outputs. Given the input sequence, a *forward* LSTM layer computes the gates and memory cells activation sequentially from $t = 1$ to T. The computation at the time step t can be formally written as follows. We omit the $\rightarrow$ arrow for uncluttered formulation.

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \qquad (3a)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \qquad (3b)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \qquad (3c)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \qquad (3d)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t) \qquad (3e)$$

where $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$ are the outputs of the input gate, output gate, forget gate and memory cells respectively. The $\mathbf{W}_{.x}$ weight matrices connect the inputs with the units, whereas the $\mathbf{W}_{.h}$ matrices connect the *previous* memory cell states with the units. The $\mathbf{W}_{.c}$ terms are diagonal weight matrices for peephole connections. Also, σ is the logistic sigmoid nonlinearity, and ϕ is the hyperbolic tangent nonlinearity. The computation of the *backward* LSTM layer can be represented similarly.

2.2. Training with Connectionist Temporal Classification

Unlike in the hybrid approach, the RNN model in our Eesen framework is not trained using frame-level labels with respect to the cross-entropy (CE) criterion. Instead, following [6, 8, 10], we adopt the CTC objective [11] to automatically learn the alignments between speech frames and their label sequences (e.g., phonemes or characters). Assume that the label sequences in the training data contain K unique labels. Normally K is a relatively small number, e.g., around 45 for English when the labels are phonemes. An additional *blank* label $\varnothing$, which means no labels being emitted, is added to the labels. For simplicity of formulation, we denote every label using its index in the label set. Given an utterance $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, its label sequence is denoted as $\mathbf{z} = (z_1, ..., z_U)$. In our implementation, we always index the blank as 0. Therefore $\mathbf{z}_u$ is an integer ranging from 1 to K. The length of $\mathbf{z}$ is constrained to be no greater than the length of the utterance, i.e., $U \leq T$. CTC aims to maximize $\ln Pr(\mathbf{z}|\mathbf{X})$, the log-likelihood of the label sequence given the inputs, by optimizing the RNN model parameters.

The final layer of the RNN is a softmax layer which has $K+1$ nodes that correspond to the $K+1$ labels (including $\varnothing$). At each frame t, we get the output vector $\mathbf{y}_t$ whose k-th element y_t^k is the posterior probability of the label k. However, since the labels $\mathbf{z}$ are not aligned to the frames, it is difficult to evaluate the likelihood of $\mathbf{z}$ given the RNN outputs. To bridge the RNN outputs with label sequences, an intermediate representation, the *CTC path*, is introduced in [11]. A CTC path $\mathbf{p} = (p_1, ..., p_T)$ is a sequence of labels at the frame level. It differs from $\mathbf{z}$ in that the CTC path allows occurrences of the blank label and repetitions of non-blank labels. The total probability of the CTC path is decomposed into the probability of the label p_t at each frame:

$$Pr(\mathbf{p}|\mathbf{X}) = \prod_{t=1}^{T} y_t^{p_t} \tag{4}$$

The label sequence $\mathbf{z}$ can then be mapped to its corresponding CTC paths. This is a one-to-multiple mapping because multiple CTC paths can correspond to the same label sequence. For example, both "A A $\varnothing$ $\varnothing$ B C $\varnothing$" and "$\varnothing$ A A B $\varnothing$ C C" are mapped to the label sequence "A B C". We denote the set of CTC paths for $\mathbf{z}$ as $\Phi(\mathbf{z})$. Then, the likelihood of $\mathbf{z}$ can be evaluated as a sum of the probabilities of its CTC paths:

$$Pr(\mathbf{z}|\mathbf{X}) = \sum_{p \in \Phi(\mathbf{z})} Pr(\mathbf{p}|\mathbf{X}) \tag{5}$$

However, summing over all the CTC paths is computationally intractable. A solution is to represent the possible CTC paths compactly as a trellis. To allow blanks in CTC paths, we add "0" (the index of $\varnothing$) to the beginning and the end of $\mathbf{z}$, and also insert "0" between every pair of the original labels in $\mathbf{z}$. The resulting *augmented label sequence* $\mathbf{l} = (l_1, ..., l_{2U+1})$ is leveraged in a forward-backward algorithm for efficient likelihood evaluation. Specifically, in a forward pass, the variable α_t^u represents the total probability of all CTC paths that end with label l_u at frame t. As with the case of HMMs [22], α_t^u can be recursively computed from α_{t-1}^u and α_{t-1}^{u-1}. Similarly, a backward variable β_t^u carries the total probability of all CTC paths that starts with label l_u at t and reaches the final frame T. The likelihood of the label sequence $\mathbf{z}$ can then be computed as:

$$Pr(\mathbf{z}|\mathbf{X}) = \sum_{u=1}^{2U+1} \frac{\alpha_t^u \beta_t^u}{y_t^{l_u}} \tag{6}$$

where t can be any frame $1 \leq t \leq T$. The objective $\ln Pr(\mathbf{z}|\mathbf{X})$ now becomes differentiable with respect to the RNN outputs $\mathbf{y}_t$. We define an operation on the augmented label sequence $\Upsilon(\mathbf{l}, k) = \{u | l_u = k\}$ that returns the elements of $\mathbf{l}$ which have the value k. The derivative of the objective with respect to y_t^k can be derived as:

$$\frac{\partial \ln Pr(\mathbf{z}|\mathbf{X})}{\partial y_t^k} = \frac{1}{Pr(\mathbf{z}|\mathbf{X})} \frac{1}{(y_t^k)^2} \sum_{u \in \Upsilon(\mathbf{l},k)} \alpha_t^u \beta_t^u \tag{7}$$

These errors are back-propagated through the softmax layer and further into the RNN to update the model parameters.

2.3. GPU Implementation

We implement the training of the RNN models on GPU devices. Model parameters are updated over entire utterances, without using the truncated version of BPTT. To fully exploit the capacity of GPUs, multiple utterances are processed at a time in parallel. This parallel processing speeds up model training by replacing matrix-vector multiplication over single frames with matrix-matrix multiplication over multiple frames. Within a group of parallel utterances, we pad every utterance to the length of the longest utterance in the group. These padding frames are excluded from gradients computation and parameter updating. For further acceleration, the training utterances are sorted by their lengths, from the shortest to the longest. The utterances in the same group then have approximately the same length, which minimizes the number of padding frames. To ensure training stability, the gradients of RNN parameters are clipped to the range of [-50, 50].

CTC learning is also expensive because the forward and backward vectors ($\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$) have to be computed sequentially, either from $t = 1$ to T or from $t = T$ to 1. Like in RNNs, our implementation of CTC also processes multiple utterances at the same time. Moreover, at a specific frame t, the elements of $\boldsymbol{\alpha}_t$ (and $\boldsymbol{\beta}_t$) are independent and thus can be computed in parallel.

3. THE EESEN FRAMEWORK: DECODING

3.1. Decoding with WFSTs

Previous work has introduced a variety of methods [6, 8, 10] to decode CTC-trained models. These methods, however, either fail to integrate word-level language models [10] or achieve the integration under constrained conditions (e.g., n-best list rescoring in [6]). In this work, we propose a generalized decoding approach based on WFSTs [23, 24]. A WFST

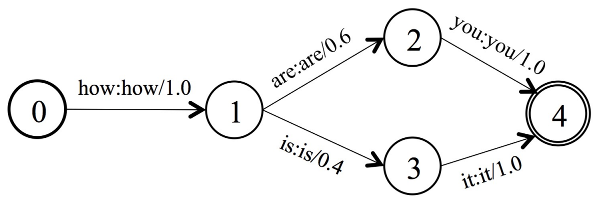

Fig. 2. A toy example of the grammar (language model) WFST. The arc weights are the probability of emitting the next word when given the previous word. The node 0 is the start node, and the double-circled node is the end node.

is a finite-state acceptor (FSA) in which each transition has an input symbol, an output symbol and a weight. A path through the WFST takes a sequence of input symbols and emits a sequence of output symbols. Our decoding method represents the CTC labels, lexicons and language models as separate WFSTs. Using highly-optimized FST libraries such as OpenFST [25], we can fuse the WFSTs efficiently into a single search graph. Building of the individual WFSTs is described as follows. Although exemplified in the scenario of English, the same procedures hold for other languages.

Grammar. A grammar WFST encodes the permissible word sequences in a language/domain. The WFST shown in Fig. 2 represents a toy language model which permits two sentences "how are you" and "how is it". The WFST symbols are the words, and the arc weights are the language model probabilities. With this WFST representation, CTC decoding in principle can leverage any language models that can be converted into WFSTs. Following conventions in the literature [24], the language model WFST is denoted as G.

Lexicon. A lexicon WFST encodes the mapping from sequences of lexicon units to words. Depending on what labels our RNN has modeled, there are two cases to consider. If the labels are phonemes, the lexicon is a standard dictionary as we normally have in the hybrid approach. When the labels are characters, the lexicon simply contains the spellings of the words. A key difference between these two cases is that the *spelling lexicon* can be easily expanded to include any out-of-vocabulary (OOV) words. In contrast, expansion of the *phoneme lexicon* is not so straightforward. It relies on some grapheme-to-phoneme rules/models, and is potentially subject to errors. The lexicon WFST is denoted as L. Fig. 3 and 4 illustrate these two cases of building L.

For the spelling lexicon, there is another complication to deal with. With characters as CTC labels, we usually insert an additional *space* character between every pair of words, in order to model word delimiting in the original transcripts. During decoding, we allow the space character to optionally appear at the beginning and end of a word. This complication can be handled easily by the WFST shown in Fig. 4.

Token. The third WFST component maps a sequence of frame-level CTC labels to a single lexicon unit (phoneme or character). For a lexicon unit, its *token WFST* is designed to subsume all of its possible label sequences at the frame level. Therefore, this WFST allows occurrences of the blank label ∅, as well as repetitions of any non-blank lables. For example, after processing 5 frames, the RNN model may generate 3 possible label sequences "AAAAA", "∅ ∅ A A ∅", "∅ A A A ∅". The token WFST maps all these 3 sequences into a singleton lexicon unit "A". Fig. 5 shows the WFST structure for the phoneme "IH". We denote the token WFST as T.

Search Graph. After compiling the three individual WFSTs, we compose them into a comprehensive search graph. The lexicon and grammar WFSTs are firstly composed. Two special WFST operations, *determinization* and *minimization*, are performed over the composition of them, in order to compress the search space and thus speed up decoding. The resulting WFST LG is then composed with the token WFST, which finally generates the search graph. Overall the oder of the FST operations is:

$$S = T \circ min(det(L \circ G)) \tag{8}$$

where $\circ$, det and min denote composition, determinization and minimization respectively. The search graph S encodes the mapping from a sequence of CTC labels emitted on speech frames to a sequence of words.

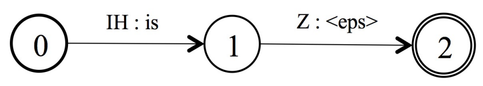

Fig. 3. The WFST for the phoneme-lexicon entry "is IH Z". The "<eps>" symbol means no inputs are consumed or no outputs are emitted.

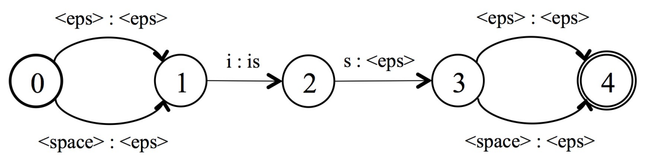

Fig. 4. The WFST for the spelling of the word "is". We allow the word to optionally start and end with the space character "<space>".

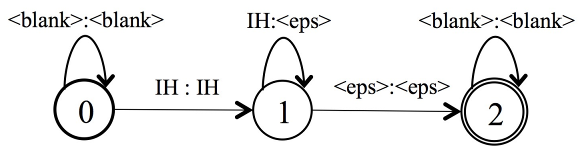

Fig. 5. An example of the token WFST which depicts the phoneme "IH". We allow the occurrences of the blank label "<blank>" and the repetitions of the non-blank label "IH".

3.2. Posterior Normalization

When decoding the hybrid DNN models, we need to scale the states posteriors from the DNNs using states priors. The priors are usually estimated from the forced alignments of the training data. During decoding of the CTC-trained models, we adopt a similar procedure. Specifically, we run the final RNN model over the training set for a propagation pass. Labels with the largest posteriors are picked as the frame-level alignments, from which priors of the labels are estimated. However, this method does not perform well in our experiments. Part of the reason is that the softmax-layer outputs from a CTC-trained model display a highly peaky distribution [11, 12]. That is, a majority of the frames have the blank as their labels. The activation of the non-blank labels only appears in a narrow region along the time axis. This causes the prior estimates to be dominated by the count of the blank.

Alternatively, we propose to estimate more robust label priors from the label sequences in the training data. As mentioned in Section 2.2, the label sequences actually used by CTC training are the augmented label sequences, which insert the blank at the beginning, at the end, and between every label pair in the original label sequences. We compute the priors from the augmented label sequences (e.g., "$\varnothing$ IH $\varnothing$ Z $\varnothing$"), instead of the original ones (e.g., "IH Z"), through simple counting. In our experiments, this simple method gives better recognition accuracy than both the aforementioned frame-alignment method and also the proposal described in [12].

4. EXPERIMENTS

4.1. Experimental Setup

The experiments are conducted on the Wall Street Journal (WSJ) corpus that can be obtained from LDC under the catalog numbers LDC93S6B and LDC94S13B. Data preparation gives us 81 hours of transcribed speech, from which we select 95% as the training set and the remaining 5% for cross validation. As discussed in Section 2, we apply deep RNNs as the acoustic models. Inputs of the RNNs are 40-dimensional filterbank features together with their first and second-order derivatives. The features are normalized via mean subtraction and variance normalization on the speaker basis.

Initial values of the model parameters are randomly drawn from a uniform distribution with the range [−0.1, 0.1]. The model is trained with BPTT, in which the errors are backpropagated from CTC. Utterances in the training set are sorted by their lengths, and 10 utterances are processed in parallel at a time. The error rate of the *hypothesized labels* is monitored to determine learning rates. The hypothesized labels are formed by firstly picking the frame-level labels (the label with the largest probability at every frame), and then removing blanks and label repetitions. The *label error rate* (LER) can be obtained in the same manner as WER, i.e., computing the edit distance between the hypothesized labels and the reference. We adopt a decaying "newbob" learning rate schedule based on LERs. Specifically, the learning rate starts from 0.00004 and remains unchanged until the drop of LER on the validation set between two consecutive epochs falls below 0.5%. Then the learning rate is decayed by a factor of 0.5 at each of the subsequent epochs. The whole learning process terminates when the LER fails to decrease by 0.1% between two successive epochs.

Our decoding follows the WFST-based approach in Section 3. After posterior normalization, the acoustic model scores need to be scaled down. The scaling factor lies between 0.5 and 0.9, and its optimal value is decided empirically. A pruned trigram language model in the standard ARPA format is used. To be consistent with previous work [6, 8], we report our results on the *eval02* set. Our experimental setup has been released together with Eesen, which enables the readers to reproduce our numbers easily.

4.2. Phoneme-based Systems

We explore the optimal RNN configurations on the phoneme-based systems. When phonemes are taken as CTC labels, we employ the CMU dictionary[2] as the lexicon. Due to the lack of forced alignments, CTC training cannot handle multiple pronunciations for the same word. For every word, we only keep its first pronunciation in the lexicon and remove all the other alternatives. From the lexicon, we extract 72 labels including phonemes, noise marks and the blank. Our best-performing model has 4 bi-directional LSTM layers. At each layer, both the forward and the backward sub-layers contain 320 memory cells. Model training ends up to reach the LER (phone error rate in this setting) of 8.8% on the validation set. On the eval92 testing set, the Eesen end-to-end system finally achieves the WER of 7.87%, with both the lexicon and the language model used in decoding. When only the lexicon is used, our decoding behaves similarly as the beam search in [6]. In this case, the WER rises quickly to 26.92%. This obvious degradation reveals the effectiveness of our decoding approach in integrating language models.

Table 1 shows a comparison between Eesen and a hybrid HMM/DNN system. The hybrid system is constructed by following the standard Kaldi recipe "s5" [24]. Inputs of the DNN model are 11 neighboring frames of filterbank features. The DNN has 6 hidden layers and 1024 units at each layer. This DNN model contains slightly more parameters (9.2 vs 8.5 million) than the Eesen RNN model. Parameters of the DNN are initialized with restricted Boltzmann machines (RBMs) that are pre-trained in a greedy layerwise fashion [26]. The DNN is fine-tuned to optimize the CE objective with respect to 3421 senones. For fair evaluations, we decode the DNN model using the original lexicon, rather than the expanded lexicon used by the Kaldi recipe. From Table 1, we observe that the performance of the Eesen system is still behind the

[2]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

hybrid HMM/DNN system. We expect to close this gap with further optimizations and improvements to Eesen.

A major advantage of Eesen compared with the hybrid approach is the decoding speed. The acceleration comes from the drastic reduction of the number of states, i.e., from thousands of senones to tens of phonemes/characters. To verify this, Table 2 compares the decoding speed of the Eesen and the hybrid HMM/DNN systems under their best decoding settings. From their real-time factors, we observe that decoding in Eesen is 3.2$\times$ faster than that of HMM/DNN. Also, the decoding graph (TLG) in Eesen is significantly smaller than the graph (HCLG) used by HMM/DNN, which saves the disk space for storing the graphs.

Table 1. *Performance of the phoneme-based Eesen system, and its comparison with the hybrid HMM/DNN system built with Kaldi. "#Param" means the number of parameters.*

Model	LM	#Param	WER%
Eesen RNN	lexicon	8.5M	26.92
Eesen RNN	trigram	8.5M	7.87
Kaldi Hybrid HMM/DNN	trigram	9.2M	7.14

Table 2. *Comparisons of decoding speed between the phoneme-based Eesen system and the hybrid HMM/DNN system. "RTF" refers to the real-time factor in decoding. "Graph Size" means the size of the decoding graph in terms of megabytes.*

Model	RTF	Graph Size
Eesen RNN	0.64	263
Kaldi Hybrid HMM/DNN	2.06	480

4.3. Character-based Systems

We apply the same RNN architecture discussed in Section 4.2 to modeling characters. We take the word list from the CMU dictionary as our vocabulary, ignoring the word pronunciations. CTC training deals with 59 labels including letters, digits, punctuation marks, etc. As mentioned in Section 3.1, a nice property of modeling characters is that we can include arbitrary new words into the vocabulary. To demonstrate this advantage, we expand the basic vocabulary by adding new words appearing in the training transcripts. Specifically, the OOV words that occur at least twice in the training transcripts are added into the vocabulary. A new trigram language model is built (and then pruned) with the expanded vocabulary for decoding. Table 3 shows that with the basic vocabulary, the character-based system gets the WER of 9.07%. Vocabulary expansion reduces the WER to 7.34%, a number very close to the performance of the hybrid system (7.14%).

Table 3 also lists the results of end-to-end ASR systems that have been reported in the previous work [6, 8] and on the same dataset. Our Eesen framework outperforms both [6] and [8] in terms of WERs on the testing set. It is worth pointing out that the 8.7% WER reported in [6] is obtained not in a purely end-to-end manner. Instead, the authors of [6] generate a n-best list of hypotheses from a hybrid DNN model, and apply the CTC model to rescore the hypotheses candidates. Our Eesen numbers, in contrast, come from a completely end-to-end pipeline, without any intervention from GMM or hybrid DNN models.

Table 3. *Performance of the character-based Eesen system, and its comparison with results presented in previous work.*

System	Vocabulary	WER%
Eesen	Original	9.07
Eesen	Expanded	7.34
Graves *et al.* [6]	Expanded	8.7
Hannun *et al.* [8]	Original	14.1

5. CONCLUSIONS AND FUTURE WORK

In this work, we present our Eesen framework to build end-to-end ASR systems. Eesen exploits deep RNNs as the acoustic models and CTC as the training objective function. We train the RNN models in a single step, and thus are able to reduce the complexity of ASR system development. The WFST-based decoding enables efficient and effective incorporation of lexicions and language models. Because of its open-source property, Eesen can serve as a shared benchmark platform for research on end-to-end ASR.

In our future work, we plan to further improve the WERs of Eesen systems via more advanced learning techniques (e.g., expected transcription loss in [6]) and better parameter initialization. Also, we are interested to apply Eesen to various languages and different types (e.g., noisy, far-field) of speech, and investigate how end-to-end ASR performs under these conditions. Moreover, due to the removal of GMMs, acoustic modeling in Eesen cannot leverage speaker adapted front-ends. We will study new speaker adaptation [27, 28] and adaptive training [29, 30] techniques for the CTC models.

6. ACKNOWLEDGEMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575. This research was performed as part of the Speech Recognition Virtual Kitchen project, which is supported by the United States National Science Foundation under grant number CNS-1305365.

7. REFERENCES

[1] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[3] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.

[4] Andrew Senior, Georg Heigold, Michiel Bacchiani, and Hank Liao, "GMM-free DNN training," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5639–5643.

[5] Michiel Bacchiani, Andrew Senior, and Georg Heigold, "Asynchronous, online, GMM-free training of a context dependent acoustic model for speech recognition," in *Fifteenth Annual Conference of the International Speech Communication Association*. ISCA, 2014.

[6] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

[7] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al., "Deepspeech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[8] Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," *arXiv preprint arXiv:1408.2873*, 2014.

[9] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[10] Andrew L Maas, Ziang Xie, Dan Jurafsky, and Andrew Y Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[11] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[12] Hasim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Francoise Beaufays, and Johan Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4280–4284.

[13] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2012.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[15] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[16] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] Hasim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[18] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.

[19] Yajie Miao and Florian Metze, "On speaker adaptation of long short-term memory recurrent neural net-

works," in *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH) (To Appear)*. ISCA, 2015.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[21] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber, "Learning precise timing with LSTM recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[22] Lawrence R Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[23] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[24] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý, "The Kaldi speech recognition toolkit," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 1–4.

[25] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Implementation and Application of Automata*, pp. 11–23. Springer, 2007.

[26] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[27] Hank Liao, "Speaker adaptation of context dependent deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7947–7951.

[28] Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012.

[29] Yajie Miao, Hao Zhang, and Florian Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[30] Yajie Miao, Lu Jiang, Hao Zhang, and Florian Metze, "Improvements to speaker adaptive training of deep neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014.