CONTEXT DEPENDENT STATE TYING FOR SPEECH RECOGNITION USING DEEP NEURAL NETWORK ACOUSTIC MODELS

Michiel Bacchiani, David Rybach

Google Inc., New York, NY

ABSTRACT

This paper proposes an algorithm to design a tied-state inventory for a context dependent, neural network-based acoustic model for speech recognition. Rather than relying on a GMM/HMM system that operates on a different feature space and is of a different model family, the proposed algorithm optimizes state tying on the activation vectors of the neural network directly. Experiments show the viability of the proposed algorithm reducing the WER from 36.3% for a context independent system to 16.0% for a 15000 tied-state system.

Index Terms— State Tying, Context Modeling, Deep Neural Networks, Acoustic Modeling.

1. INTRODUCTION

In recent years, use of neural networks as the acoustic model of a speech recognition system has gone from novel to the most common approach [1]. Two types of use of such models appears in the community. In the hybrid approach, the neural network replaces the previously common Gaussian Mixture Model (GMM) emission probabilities of the Hidden Markov Models (HMM) (eg. [2, 3, 4, 5]). In the other approach, the neural network is used to compute features and the rest of the recognition system is unchanged, ie. the HMMs still use GMMs in that case, but they operate on the features computed by the neural network (eg. [6, 7, 8]).

Given the large improvements of recognition accuracy using neural networks, either in the hybrid approach or as a feature extractor, there is a clear determination in the field to keep using neural network-based systems. However, given that the nature of these models is in stark contrast with the GMM/HMM-based systems, a lot of the commonly used algorithms like speaker adaptation do not directly apply unless the neural network is used as a feature extractor alone. Hence there is an interest in removing the dependency on GMMs (for hybrid systems), while retaining the algorithmic knowledge that has been developed in light of such models.

A hybrid neural network system relies on a GMM/HMM to allow training of the network. If training uses a cross entropy (CE) objective, the acoustic observations of the training set need to be associated with state identites of the HMMs. To derive such labels, a fully trained GMM/HMM system is used to force align the training material. More recent work alleviates that requirement by the introduction of sequence training [9, 10, 11] which replaces the frame-based CE objective with one that optimizes the discriminative sequence-based metrics commonly used in GMM/HMM systems (MMI, MPE, sMBR or similar). This effectively "ports" some of the algorithmic knowledge developed in light of the GMM/HMM model to the new neural network framework. Alignment statistics computed with the neural network itself are in that case used to guide the network optimization (although generally sites would still start with a CE trained network and it is not clear if the optimization would converge to a well performing system without that). However, even with sequence training, a dependency on the GMM/HMM system remains as the neural network is trained to predict probabilities of a tied-state inventory that is constructed using the Gaussian-based system, typically through decision-tree clustering [12]. In other words, for a CE-objective system the depency on a GMM/HMM is two-fold (alignment to label the neural network training data and the definition of the state inventory), for a sequence trained system it is only based on the state inventory definition.

Gaussian context-dependent state tying uses the acoustic observations that are used in a GMM/HMM system (eg. Perceptual Linear Predictive (PLP) features) which generally are not the same type as the feature representation used in a neural network-based system (most use log filterbank energies instead). Furthermore, decisions on which phonetic contexts to tie in the tied-state inventory are based on how well the grouped phonetic contexts are modeled jointly by a Gaussian distribution. Hence the tied-state definition used in the neural network replacement of the GMM model is constructed on a mismatched feature set using a mistmatched model family.

In this work we propose an algorithm to design the tied-state inventory using activations of an a priori trained context independent neural network. As such, the algorithm is directly matched to the features and modeling that is employed in the neural network and does not rely on a mismatched GMM/HMM system and distinct feature representation. Furthermore, since the state inventory is defined post hoc, the algorithm is more amenable to changes in the state inventory. Like sequence training, the algorithm "ports" some of the knowhow developed in light of context clustering for GMM/HMM systems in the sense that is still uses decision tree-based clustering to define the tied-state set. Retaining that aspect also allows the proposed algorithm to integrate well with the recognition search in testing. For the work presented here, we show the feasibility of such a tied-state set construction algorithm using a network optimized with a CE criterion. As such, it still represents a dependency on a baseline GMM/HMM system but when used with sequence training, this would alleviate the dependency entirely (assuming the CE network initialization for sequence training can be omitted).

The rest of this paper is organized as follows. In section 2 we described the tied-state inventory construction algorithm. In section 3 we show experimental results using the proposed algorithm and finally, in section 4 we present conclusions and future work directions.

2. ALGORITHM

The algorithm consists of three stages, the initialization stage described in section 2.1, the state tying stage described in section 2.2 and the post-training stage described in section 2.3.

2.1. Initialization

We assume the neural network consists of an input layer where a window of acoustic observation vectors are presented. That layer is connected with trainable weights to a stack of hidden layers and a final softmax layer. The hidden layers compute weighted sums of the activations it receives from its connections and a bias and outputs activations from a non-linear function applied to the sums. The final softmax layer produces the network output for N classes with the n-th class probability for the t-th acoustic input pattern $\mathbf{x}(t)$ defined as

$$\mathrm{P}(n \mid \mathbf{x}(t)) = \frac{\exp(\mathbf{w}_n^T \mathbf{a}(t) + b_n)}{\sum_{i=1}^{N} \exp(\mathbf{w}_i^T \mathbf{a}(t) + b_i)}, \quad (1)$$

where $\mathbf{w}_n$ denotes the weight vector associated with the output neuron for the n-th class, b_n denotes the bias of that neuron, $\mathbf{a}(t)$ denotes the activation output from the final hidden layer (the one connected to the output softmax layer) for the t-th input pattern and ^{T} denotes transposition.

For the initialization of the algorithm, the final network topology is defined for the input and hidden layers and a first softmax layer is defined for N Context Independent (CI) states. This network is then trained on the training set. Note that for CE training (as is used in the experiments here), the acoustic observations of the training set need to be labeled with CI state labels. Let the label for the t-th frame be denoted as $l(t)$.

2.2. State Tying

The state tying stage first computes activation vectors for all training set frames using the trained CI network from initialization. Specifically, it computes $\mathbf{a}(t)$ from equation (1) for all t in the training set.

Let $\mathbf{c} = c_1, \ldots, c_L$ denote an L symbol context. For example, for a triphone system, L would be 2 and c_1 encodes the left and c_2 encodes the right context. Furthermore, let $\mathbf{c}(t)$ denote the context for the t-th observation and let $c_p(t)$ denote the p-th symbol in that context. Now define $\Phi(n, p, s) = \{a(t) : l(t) = n \text{ and } c_p(t) = s\}$, the set of all activation vectors for CI state n that have symbol s in the p-th context position.

We then construct decision trees using the algorithm described in [13]. Starting from a single node tree for each CI state n, we greedily split leaf nodes that have the largest likelihood gain under a Gaussian model. A potential new left and right child node are defined by a Gaussian distribution so any leaf in the tree is characterized by a Gaussian centroid model. The use of a Gaussian distribution assumption of the activation vectors is for efficiency as it allows the use of sufficient statistics to implement the algorithm.

Potential splits are evaluated by partitioning the activation vectors in a parent node into a left and right child set and computing the likelihood gain from modeling the activation vectors in the left/right child partitioning as opposed to jointly using the parent distribution.

Partitioning the activation frames in a node of the tree for phone state n considers assigning the frames for a particular context symbol s in a particular context position p to either the left or right child. In other words, the activation vectors $\Phi(n, p, s)$ are assigned to either the left or right child, they cannot be divided in the assignment. Likelihood of such hypothesized partitionings are computed to assess the gain of a hypothesized split. The constraint that any $\Phi(n, p, s)$ is assigned to a single child means that a resulting hypothesized split can be expressed as a set symbol set question on a particular context position p.

The gain and definition of a hypothesized context position split is computed jointly by binary divisive, likelihood-based K-means clustering. The set of activation vectors for a given context symbol s in position p (ie. $\Phi(n, p, s)$) is assigned to either the right or left child based on the larger Gaussian likelihood of that data under the left or right child distribution. Once the activation vector sets are partitioned, the Gaussian centroid for the left and right child are re-estimated in a maximum likelihood sense and this process is iterated until converged. Treating the activation vector sets for a given context symbol s as atomic, the K-means clustering process computes the likelihood gain as well as the partitioning that produces that gain.

The algorithm considers the partitioning of any leaf in any tree on any context position and greedily implements the split with the largest gain. The newly formed children are then greedily evaluated for their best split parameters and this process is iterated until no further leaf splits can be found that lead to a likelihood gain.

Note that this process is very similar to the commonly used decision tree-based tying of context dependent states [12]. The difference is that here the splits are evaluated on the activation vectors that feed into the final softmax layer, not based on features that are mismatched with the neural network. Another contrast with previous work is that no pre-defined context sets are used, but the Chou clustering [13] algorithm is used instead to define the set questions. Since the clustering is performed on the output of one or more layers of non-linearities, we want to have a data driven design of the set questions. That said, experimental results show similar set questions are learned from data as would be used in clustering with manually designed classes.

Given the decision tree clustering, the trees can be pruned back using a gain threshold to a desired state set size. To allow use of this tied-state inventory in recognition experiments, we compile the pruned trees into a context dependency transducer using the splitting part of the algorithm described in [14] (the split optimization of that work is not used). Note that for the leaves of the trees, we have the Gaussian centroid providing an estimate of the mean activation vectors of frames that are assigned to the context dependent (CD) tied states.

2.3. Post Training

Given the tree clustering described in section 2.2, we initialize training of the final context dependent tied-state inventory network. Each of the acoustic observations $\mathbf{x}(t)$ of the training set are mapped to the new tied-state inventory. If an observation was previously labeled as CI state n, the observation is mapped to the tied CD state by mapping it using the tree for n and looking up the leaf corresponding to $\mathbf{c}(t)$.

The neural network itself is initialized by expanding the softmax layer number of neurons to the new CD state inventory size. Since the softmax is a maximum entropy model it is not clear what a good initialization is. Given that we know the mean activation vector for a particular node from the centroid of the leaf of the decision tree, we set the weights going into the neuron to the mean activation.

The hidden layer parameters are retained from the initialization stage. Subsequent post training can be limited to just the softmax layer or all weights in the network. Since the tied-state inventory is designed based on similarity of the activation vectors, they are matched to the weight estimates of the hidden layers, though optimized for the CI state inventory.

3. EXPERIMENTS

Experiments were conducted on a database of mobile speech recordings originating from a number of Android applications: voice

search, translation and the voice-based input method. These recordings are anonymized; we only retain the speech recording but remove all information indicating which device recorded the data. The training set consists of a sampling of about 3 million utterances containing about 2000 hours of speech. We obtained manual transcriptions for these utterances. Evaluation is performed on a test set that contains data sampled uniformly from all applications emphasizing the use frequency of each application. This test set contains about 23000 utterances or about 20 hours of speech.

We first trained a CI GMM/HMM system using a 42 phone inventory. Each phone is modeled with a 3-state left-to-right HMM giving the CI system a 126 state inventory. Training started out with a flat start and followed with incremental mixture training. State distributions in the final system were modeled with Gaussian mixtures of up to 64 components. The CI GMM/HMM system was then used to force align the training set, labeling the acoustic observations in the training set with CI state labels. One could use a CD system to potentially get better CI state labels, but given that the study focuses on removing dependency on a CD GMM/HMM system, we use only the CI system here.

For the initial neural network, 40 dimensional log filterbank energy features were computed at a rate of 100 frames per second. These frames were stacked with a context window of 20 frames to the left and 5 frames to the right. The resulting training set representation is about 680 million observations of 1040-dimensional vectors with associated CI state labels. In addition, we computed the triphone context for each observation, ie. for each training frame we encode the left and right neighboring phone label. With respect to the algorithm description in 2.2, we set L to 2 and define $\mathbf{c}$ to encode the triphone context.

We then trained an initial neural network as described in 2.1. This network consists of 7 fully connected hidden layers of 2560 neurons each. The softmax output layer has 126 outputs, one for each CI state label. For the hidden layers we use ReLU nonlinearities [5]. We trained this network using our distributed deep network training infrastructure [15], using asynchronous gradient descent with a fixed 0.003 learning rate and training steps (mini-batch updates) of 200 frames. The training minimizes a cross entropy loss. We trained this network for about 22 million steps (about 4.4B training frames or about 6.5 epochs). The averaged entropy loss over the training process is depicted in figure 1.

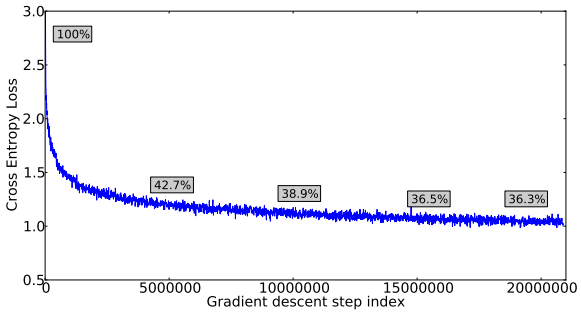

Fig. 1. Cross entropy loss of the CI state system as a function of the gradient descent step index and word error rates on the test set using the network estimates from sampled step indices.

For all experiments recorded here, in addition to tracking the training CE loss, we periodically ran a word error rate (WER) evaluation on the test set on intermediate network estimates. The system used in each of these evaluations uses a vocabulary of about 2.5 million words and a trigram language model with about 33 million ngrams. The language model and lexicon (and in case of a context dependent system, the context dependency definition) are combined into a single decoding graph using the approach described in detail in [16].

Besides the cross entropy loss, figure 1 shows the WER measurements of the network at different stages in the training process. The CI error rate of the converged network is 36.3%. Training the network up to that point using 100, 4-way partitioned model replicas to parallelize the training took about 95 hours.

We then computed 2560-dimensional activation vectors for the 680 million training frames from the 7-th hidden layer of the initial network. Gaussian statistics of those activation vectors for any observed CI state in unique context were derived. A total of 103518 unique statistics accumulators (CI state label in context) were found (silence states were modeled context independently). These statistics were then used to perform the likelihood based clustering described in section 2.2. In figure 2 we show the log-likelihood gains (on a double log scale) of successive greedy splits. The plot is not monotonically decreasing as tree topology constraints are taken into account (ie. a child can have a larger split gain than its parent, but that gain can only be implemented if the parent is split first). Since the activation vectors used in clustering are after application of various non-linearities, there is no guarantee that phones with phonetically similar production properties have similar activation patterns. However, observing some of the frequent context questions that were inferred by the data driven clustering algorithm we see phone groups like for example nasals `en n nx`.

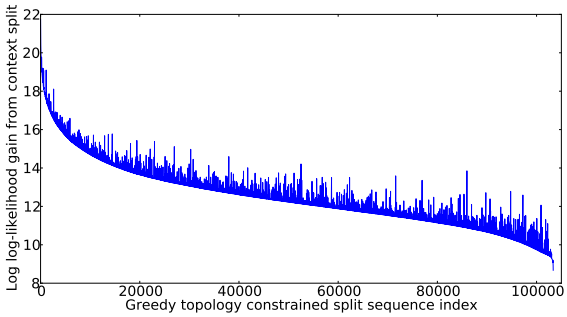

Fig. 2. Log log-likelihood gains from tree leaf splitting. Gains are ordered as chosen by the growing algorithm, ie. greedy but under the constraint of the topologies of the grown trees.

We then pruned the trees back to create state inventories of 400, 800, 2000, 4000, 8000, 12000 and 15000 tied states. Neural networks retaining the hidden layer weights from the initial network but with mutated softmax layers were constructed for the state clustering defined by the pruned trees as proposed in section 2.3. These networks were then trained for 15 million (200 frame mini batch) training steps. In the training we only optimize the weights and biases of the softmax layer, the hidden layer weights are kept fixed. The CE loss evolution of these networks through the training process and periodic WER evaluation are depicted in figure 3

Observe that the centroid mean initialization of the softmax weights appears a poor choice as the initial CE loss of the work is large. Training reduces the metric and WER measurements show that the tied-state systems gain over the CI system when trained for 15 million steps. The smaller state inventory systems converge more quickly. The best WER of 18.8% (a 48% WER reduction compared to the CI system) is obtained using the 2000 state inventory system.

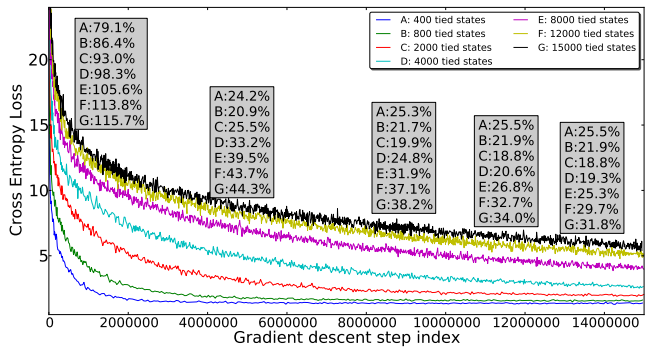

Fig. 3. Cross entropy loss of systems of various tied-state inventory sizes as function of the gradient descent step index and word error rates on the test set using the network estimates from sampled step indices.

It is unclear if the larger systems perform more poorly due to a lack of convergence or if the activation vectors feeding into the softmax layer are sub-optimal. Note that since the hidden layers are frozen in this run, the softmax of any of these systems operates on the same activations.

To investigate, we continued training the networks for the 4000 or larger state inventories, updating either the softmax layer weights only or allowing weights in the network to be trained jointly. The CE loss evolution and WER measurements of the softmax only updates are depicted in figure 4, the training evolutions for the runs that updated all network weights are depicted in figure 5.

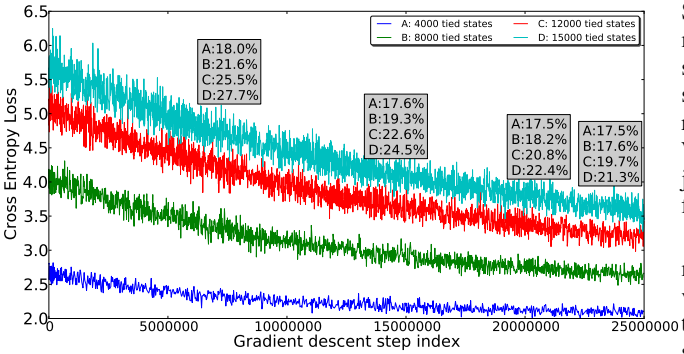

Fig. 4. Cross entropy loss of systems of various tied-state inventory sizes as function of the gradient descent step index and word error rates on the test set using the network estimates from sampled step indices. Initialization is with the network from the first 15 million step training phase and only softmax weights are updated.

The network training that allows the hidden layer weights to be updated outperforms the network training that only allows softmax layer updates. The additional gain is more pronounced for the larger networks. The 15000 state network reaches a word error rate of 16.0%.

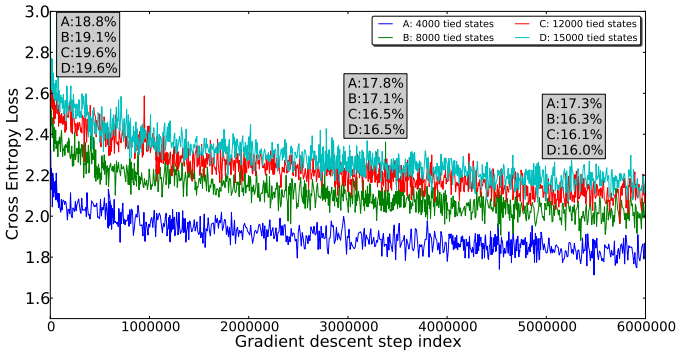

Fig. 5. Cross entropy loss of systems of various tied-state inventory sizes as function of the gradient descent step index and word error rates on the test set using the network estimates from sampled step indices. Initialization is with the network from the first 15 million step training phase and all layer weights are updated.

4. CONCLUSIONS AND FUTURE WORK

This paper proposes a context dependent state tying algorithm that directly optimizes the state set on the activations in the neural network. The proposed algorithm allows the design and training of such networks and use of them in recognition. Experiments show that training of a network up to 15000 tied states improves the word error rate from 36.3% for a context independent system to 16.0% for the tied-state context dependent system.

The state inventory tying definition is optimized on the activation vectors of the context independent trained system. For smaller state inventory networks, this presents a reasonable feature space. First, creating a larger state inventory results in WER reductions (eg. 25.5% for the 400 state system vs. 21.9% for the 800 state system). Second, comparing the subsequent constrained, softmax only, optimization vs. the joint optimization of all weights of the network, the smaller networks show little performance difference (eg. the 4000 state system achieves 17.6% vs. 17.3%). For larger state inventories, the CI activation vectors appear less well suited. We still see WER reduction for increased inventory sizes but the constrained vs. joint training shows larger WER differences (eg. 21.3% vs. 16.0% for the 15000 state system).

The proposed algorithm for tied-state inventory design is better matched to the neural network model as it directly optimizes the inventory based on the features and model family. In that sense it effectively reduces dependency on an external, mismatched GMM/HMM system. However, it still relies on the labeling of input frames based on a CI GMM/HMM system. Furthermore, the alignment of that system is fixed in this work limiting the ability of the training to optimize for string WER. For comparison, when training a 15000 state context dependent GMM/HMM system from the same CI system starting point, force aligning the training data to label the frames with the CD state labels and training a DNN with the same topology as used here results in a 14.5% WER. Given that in that case the state inventory has been optmized with a mismatched model one expects a performance degradation, but given that the better CD GMM/HMM system is used to segment and label the input, the resulting system is more accurate. Future work will hence focus on adding an iterative or adaptive aspect to the proposed algorithm and not rely on an early fixed segmentation for system design and training.

5. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A.Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] T. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making Deep Belief Networks Effective for Large Vocabulary Continuous Speech Recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[3] F. Seide, G. Li, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in *Proc. of Interspeech*, 2011.

[4] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application Of Pretrained Deep Neural Networks To Large Vocabulary Speech Recognition," in *Proc. of Interspeech*, 2011.

[5] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On Rectified Linear Units for Speech Processing," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013.

[6] T. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-Encoder Bottleneck Features Using Deep Belief Networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2012.

[7] Z. Yan, Q. Huo and J. Xu, "A Scalable Approach to Using DNN-Derived Features in GMM-HMM Based Acoustic," in *Proc. of Interspeech*, 2013, pp. 104–108.

[8] P. Bell, H. Yamamoto, P. Swietojanski, Y. Wu, F. McInnes and C. Hori, "A lecture transcription system combining neural network acoustic and language models," in *Proc. of Interspeech*, 2013, pp. 3087–3091.

[9] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable Minimum Bayes Risk Training of Deep Neural Network Acoustic Models Using Distributed Hessian-free Optimization," in *Proc. of Interspeech*, 2012.

[10] K. Vesely, A. Ghoshal, L.Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. of Interspeech*, 2013.

[11] H. Su, D. Yu and F. Seide, "Error Back Propagation for Sequence Training for Context-Dependent Deep Networks for Conversational Speech Transcrtiption," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013, pp. 6664–6668.

[12] S. Young, J. Odell, and P. Woodland, "Tree-Based State Tying for High Accuracy Acoustic Modelling," in *Proc. ARPA Human Language Technology Workshop*, 1994.

[13] P. Chou, "Optimal partitioning for classification and regression trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 340–354, 1991.

[14] D. Rybach, M. Riley, and C. Alberti, "Direct construction of compact context-dependency transducers from data," *Computer Speech and Language*, vol. 28, pp. 177–191, 2014.

[15] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. Neural Information Processing Systems*, 2012.

[16] M. Mohri, F. Pereira, and M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.