GMM-FREE DNN ACOUSTIC MODEL TRAINING

Andrew Senior, Georg Heigold, Michiel Bacchiani, Hank Liao

Google Inc.,
New York
{andrewsenior,heigold,michiel,hankliao}@google.com

ABSTRACT

While deep neural networks (DNNs) have become the dominant acoustic model (AM) for speech recognition systems, they are still dependent on Gaussian mixture models (GMMs) for alignments both for supervised training and for context dependent (CD) tree building. Here we explore bootstrapping DNN AM training without GMM AMs and show that CD trees can be built with DNN alignments which are better matched to the DNN model and its features. We show that these trees and alignments result in better models than from the GMM alignments and trees. By removing the GMM acoustic model altogether we simplify the system required to train a DNN from scratch.

***Index Terms*—** Deep neural networks, hybrid neural network speech recognition, Voice Search, mobile speech recognition, flat start, Viterbi forced-alignment, context dependent tree-building.

1. INTRODUCTION

In recent years, neural networks have become a fundamental part of Hidden Markov Model (HMM) speech recognition systems. Neural networks may be used either as nonlinear feature extractors, where the features are used in a conventional Gaussian Mixture Model (GMM) likelihood estimator [1, 2], or as so-called "hybrid" models where a neural network estimates class posterior probabilities [3].

In this paper, we investigate hybrid models. While the use of hybrid models removes the need for a GMM model at inference time, such systems still have a dependence on the GMM systems that preceded them in two main areas:

1. **Alignment:** A neural network is trained using a state alignment that provides a label for each frame in the training set. Once a DNN model is available, it can be used to force align the training set, but in all the systems that we know of, the original alignment for a language was generated with a GMM system. In contrast, GMM systems are often "flat started" [4] from a phonetic transcription without any initial time alignment.

2. **Context dependency trees:** modelling context dependent (CD) states results in a significant accuracy gain compared to context independent (CI) states both in GMM-HMM systems and DNN-HMM hybrid systems. Again, in all the cases we know of, the context-dependency tree is constructed using an alignment generated with a GMM-HMM. Further, these context dependency trees are built with the features used by GMMs, not those used by the DNNs, although Bacchiani [5] recently proposed building trees for DNNs with DNN activations as features.

In addition, DNN systems that use constrained maximum likelihood linear regression [6] for speaker adaptation also use a GMM to compute the alignment from frames to mixture components, and hence the linear transform to apply to the features.

In this paper we show that DNNs for acoustic modelling can be flat started (Section 2) and their alignments used for building CD state-tying trees (Section 3) which can be used for training CD DNNs (Section 4). We further show that on-the-fly realignment can be used to refine DNN models. Experiments and conclusions are presented in Sections 5 and 6 respectively.

2. FLAT STARTING

A GMM-HMM AM is typically trained on a set of acoustic utterances, each of which is accompanied by a human- or machine-generated transcription. The word-level transcription can be converted to a state sequence using a lexicon that provides the pronunciation(s) of each word; as well as context and HMM transducers that generate a sequence of context-dependent states for the transcription. Initially such a state sequence has no timing information and we do not know which parts of the acoustic signal correspond to which states in the state sequence. In the expectation-maximization (EM) [7] training algorithm, we alternate finding the maximum likelihood state alignment given the current GMM AM with finding the expected parameters of the GMM AM given that alignment.

When building a GMM-HMM speech recognition system from scratch, the iterative EM algorithm is repeated many times in phases, between which the model complexity or fea-

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

tures are changed. For instance, early phases will use CI states, simple features (such as perceptual linear predictive features (PLPs) with deltas and delta-deltas, denoted PLPDA) and few mixture components per modelled distribution. Subsequently, discriminatively trained features such as linear discriminant analysis (LDA) features may be used; CI states may be replaced with a much larger inventory of CD states; and each state is modelled with an increasing numbers of mixture components.

In this sequence of increasing model complexity, the model's accuracy increases and the alignment from acoustic frames to states is progressively refined. It should be noted, however that when CI states are replaced with CD states, the state boundary definitions become less clear since there is no acoustic constraint on whether a frame should be classified as A followed by B vs B preceded by A.

To bootstrap the training of such a model, we must construct either a state alignment without a model or a model without a state alignment, in a "flat start" initialization. One way of doing this is to initialize a one-Gaussian-per-state CI model with the global dataset mean and covariance. With the constraint of the state sequence for each utterance, such a model can generate a crude alignment which is sufficiently accurate for the EM algorithm to build upon, eventually producing an accurate speech recognition system.

To our knowledge, all hybrid DNN systems (such as [8, 9, 10]) are built using an alignment that can ultimately be traced back to a GMM model, or in some cases [11] to the manual phone alignment of the TIMIT dataset. These DNN models thus have implicit dependency on the original GMM used, as well as the features and context window size of that model.

While we are unaware of any DNN-based speech systems that use a flat start, neural network handwriting systems [12] have used flat start, for instance using an "equal-length" segmentation where an alignment is constructed without a model assuming that all the characters are equal length.

2.1. DNN Flat Start

In our work, we adapt a GPU-based neural network trainer that has been used to train large-scale production speech models using alignments from GMM-HMMs and DNN-HMMs. We randomly initialize a neural network with zero-mean, small variance weights and construct a CI state sequence (*without* any timing information) for each utterance from the word-level manual transcription. The untrained model is used to force-align a batch of data (10,000 frames, or roughly 25 utterances). The labelled frames are shuffled randomly and the model is trained with stochastic gradient descent on mini-batches of 200 frames and a cross entropy criterion.

Adopting this approach means that, except for the first minibatch of each batch, the model used at training time is not the model used for alignment, so there is a certain amount of "staleness" in the alignments, but the alignments are constrained to match the provided state sequence and they will not change significantly, except at the start. After this, the batch size can be increased for greater computational efficiency and better shuffling.

The simplest alternative is to align each utterance and backpropagate its frames, which results in no staleness. However, training on many frames from a single utterance (which are highly correlated in speaker, channel and background conditions as well as in which state symbols are observed) leads to a poorer gradient estimate (less representative of the ideal gradient computed on the whole training set) and hence slower convergence than from a minibatch of random frames. A disadvantage is that each frame must be forward-propagated twice — once for alignment and once to compute the gradients for backpropagation. Activations could be stored, but this introduces a further staleness and additional storage and system complexity.

Alignments are made with the Viterbi algorithm on the GPU using custom kernels. One difficulty with performing on-the-fly alignment is that it is hard to compute an objective measure of the performance of the network during training since the conventional measures of frame accuracy, cross entropy or squared error are computed relative to the targets from the alignment which are taken as ground truth. In this case the alignments are given by the network, and will inevitably be consistent with the frame-level predictions resulting in high frame accuracies. For instance a network tending to give high scores for silence can produce alignments consisting almost entirely of silence, and relative to these targets it will appear to have very high frame accuracy.

2.2. State prior estimation

The alignment is also used to compute the state prior distribution, $P(s_i)$ which is essential for computing the scaled posteriors used during HMM decoding ($P(x_t|s_i) \propto P(s_i|x_t)/P(s_i)$ for states s_i and observations x_t at time t). Scaled posteriors are also required for computing the forced alignments during training. When flat starting, the priors must be bootstrapped along with the model. State priors can be estimated by forced-aligning the whole training set and using the normalized state frequencies as the prior distribution. To estimate the empirical class frequency with minimum computation, and update it as the model changes, we count the state observations $c_i(\tau)$ after each alignment, τ, and update a running average $c_i^*(\tau)$ of state observations.

$$P(s_i) \approx \frac{c_i^*}{\sum_j c_j^*} \quad (1)$$

$$c_i^*(\tau) = \gamma c_i^*(\tau - 1) + c_i(\tau) \quad (2)$$

This average is initialized with equal counts (corresponding to a uniform prior which matches the posteriors given by the randomly initialized model's outputs), and updated with a decay factor γ. Choosing γ too small means that rarely observed

symbols will be forgotten. Choosing it too large means that the network's posteriors are stale, leading (before the prior has converged) to bias during decoding — particularly affecting silence and resulting in speech frames being aligned to silence or silence being aligned to speech labels. In our experiments we have found that $\gamma = 0.995$ worked well. When considering flat start with a CI state inventory, no symbol is too rare, and later in training we can use a reasonable model to compute a prior once over a large dataset. Subsequently the empirical prior changes more slowly and online updating is less important.

3. CONTEXT-DEPENDENT TREE BUILDING

It is well known that modelling phonetic states separately, according to the phonetic context in which they occur, can lead to better modelling of the state distribution. Because of the very large number of possible contexts, many of which may not be observed in the training set, data-driven tree-based state-tying algorithms [13, 14] have been developed to cluster together sets of contexts for a given CI state, with each such cluster or CD state being modelled separately.

Having bootstrapped a CI DNN model, this can be used to compute an alignment of the training set which is then used to build a context clustering tree for each CI state, using the method of Young *et al.* [14]. For each CI state, sufficient statistics of observations $\mathbf{x}$ are gathered for each possible context. In this case we build triphone trees, considering the cross-word context of the preceding and following phone. We accumulate sufficient statistics to model each context dependent state with a diagonal covariance Gaussian. (Count, mean and sum-of-squares of $\mathbf{x}$.) Given the context labellings and sufficient statistics, a greedy hierarchical divisive clustering is applied, choosing the split which maximizes the likelihood gain from a set of predetermined, linguistically-motivated questions. Splits are applied recursively with the constraint that no leaf may have fewer than 10,000 observations. The 126 trees are then aggregated (having a total of around 40,000 leaves) and pruned back by iteratively merging the minimum likelihood split until the set of trees has the desired number of states (2000 or 16,000 for the experiments described here), yielding a nested hierarchy of state inventories.

4. CD TRAINING

The resulting context dependency clustering can then be used to relabel the CI alignment from the flat start model, producing an alignment with the same state boundaries but with CD labels. This can then be used to train a new model with the new CD state inventory. For a network which differs only in the output state inventory, the new network can be initialized with the weights of the old network, copying the weights of the CI states for all the CD state outputs that are descended from that CI state, though in the experiments presented here we always train a new network from a random initialization.

Once we have trained the CD model, we can use it to generate a new forced-alignment of the training data. Such an alignment will be better-matched to the current model and potentially lead to better performance, particularly for a more complex model than that used to generate the original alignment.

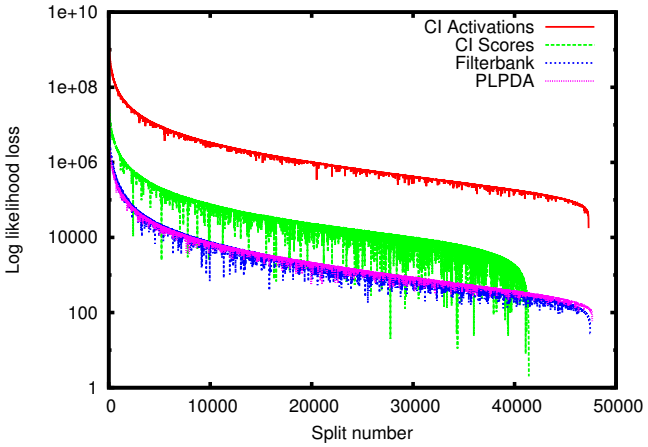

Fig. 1: A graph of log likelihood losses (log scale) as splits in the trees are greedily merged, from the $\sim 40,000$ state full trees on the right, back to the 126 CI states at the left. The high (1024) dimensional CI activations give the highest split gains and the smoothest curve.

5. EXPERIMENTS

5.1. Flat start

First, we show the results of training a flat start model as described in Section 2. We train a 6-hidden-layer network with 1024 sigmoid units per layer and 126 CI softmax outputs on a 1900 hour corpus of anonymized, hand-transcribed US English voice search and dictation utterances. WER evaluations were carried out on a disjoint 23,000 word test set of similar utterances. The network inputs are stacked 40-dimensional $25ms$ log mel filterbank energies with a context window of 16 past and 5 future frames. This flat started CI model achieves a WER of 21.3% on the test set, compared to a model with the same configuration trained using the best alignment available (from a large, 85M parameter ReLU DNN with 14,000 CD states) which achieves a WER of 16.4%. This seems to be a reasonable WER given the limited alignment and a reasonable starting point for CD tree building.

5.2. Tree-building

We then build CD trees as described in Section 3 using a variety of features. Traditionally GMM CD trees are built using

Tree-building Features	WER (2.7M)	WER (56M)
GMM `plpda`	14.3	10.4
DNN `plpda`	14.7	10.5
DNN `fb`	14.6	10.6
DNN `fbda`	14.5	10.5
DNN `ciact`	14.3	10.3
DNN `ciscore`	**14.2**	10.4

Table 1: WERs for small (2.7M parameter, 2,000 output) and large (56M parameter, 16,000 output) networks after training with the state inventories for different trees using the alignment of the best available model (85M parameter, 14,000 outputs, using GMM `plpda` trees).

39-dimensional PLPDA features. Here we build trees with these features (`plpda`), with single-frame filterbank energies (`fb`: 40 dimensions); with `fb` features and their velocities and accelerations (`fbda`: 120 dimensions); with the CI log posteriors from the aligning network (`ciscore`: 126 dimensions) and the activations of the penultimate layer of the DNN (`ciact`: 1024 dimensions). These trees all have roughly 40,000 leaves. Figure 1 shows the log likelihood gains of the splits chosen by the greedy algorithm.

5.3. Training with DNN-derived trees

As described in Section 4 we then train DNN models on this alignment. Here we train models with 6 hidden layers of 512 rectified linear units (ReLUs) [15] using labels from the 2000 state inventory from each of the trees. To compare these models with different CD state inventories, we measure the CI state frame accuracy, FA_{CI}, is the fraction of development set frames, $x(t)$, for which the correct CI state $CI^*(t)$ has the greatest posterior, summed over the child CD states CD_j:

$$FA_{CI} = \frac{1}{N}\sum_{t=1}^{N} \delta(CI^*(t), \arg\max_i P(CI_i|x(t))) \quad (3)$$

$$P(CI_i|x(t)) = \sum_{CD_j \in CI_i} P(CD_j|x(t)). \quad (4)$$

We first train models using the alignment from our best, large DNN. The results, in Table 1 show that models using trees built with the DNN features (`ciscore` and `ciact`) perform very well, even outperforming GMM `plpda` trees that match the model used to generate this alignment.

We then train models on the GMM-free, DNN CI alignment. The WERs for these models are shown in Table 2. Among the models with the CI alignment (among which the WER variation is small), the model with the GMM state inventory performs relatively poorly. For reference, training equivalent models with the best model alignment, a WER of 14.3% is achieved.

Tree-building Features	Dimension	WER	FA_{CI}
GMM `plpda`	39	15.3	67.0
DNN `plpda`	39	15.2	66.8
DNN `fb`	40	15.1	66.8
DNN `fbda`	120	**15.0**	66.8
DNN `ciscore`	126	15.1	66.9
DNN `ciact`	1024	15.2	67.0

Table 2: WERs and CI state frame accuracies for 2000-output models trained on the alignment from the bootstrap CI DNN with different CD state-tying trees. Here models using trees built with the DNN-alignment all outperform the original GMM-alignment-based trees.

If we train a large (55M parameter, 16,000 `ciscore` CD states) model on the CI alignments, and realign the data, then retrain the small model, it achieves a WER of 14.8%.

5.4. Realignment

We trained a large network (6 hidden layers of 2176 ReLUs with 14000 outputs) with a fixed alignment generated from a network of the same shape. By 7 billion frames the network reaches a WER of 9.4%. (WERs in this paragraph were on a different test set yielding lower WERs than in the previous section.) However, if the same network is cloned at 2 billion frames, when the WER was 9.8%, and training is continued with on-the-fly batch Viterbi realignment, the WER reached 9.3%.

6. CONCLUSIONS

We have shown that DNNs can be bootstrapped without an initial alignment by on-the-fly Viterbi alignment using a randomly-initialized model. Further, we have demonstrated that context dependent DNNs can be successfully trained without the use of GMMs, either for flat start or for the alignment for CD tree building. By building the CD trees on an alignment from a DNN and using features better suited to a DNN, we have shown that such GMM-free training can result in better models than when using conventional, GMM-based flat starting and CD tree building. Among the features that we have explored for tree-building, the CI scores of the bootstrap network seem to be best, though filterbank features and the penultimate layer activations also outperform the original trees. Realignment of well-trained models has shown small gains, although larger gains can be expected from sequence training [16].

In future work, we plan to investigate alternative features, and the impact of iteratively realigning with larger, state-of-the-art models from flat start CI alignments.

7. REFERENCES

[1] H. Hermansky, D.P.W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*, 2000.

[2] T.N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proc. ICASSP*, 2012.

[3] N. Morgan and H. Bourlard, "Continuous speech recognition: An introduction to the hybrid HMM/connectionist approach," *IEEE Signal Processing Magazine*, vol. 12, no. 3, pp. 25–42, 1995.

[4] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *HTK Manual*, 3.1 edition, 2000.

[5] M. Bacchiani, "Context dependent state tying for speech recognition using deep neural network acoustic models," in *Proc. ICASSP*, 2014.

[6] A. Mohamed, T.N. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Proc. ICASSP*, 2011.

[7] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–39, 1977.

[8] G. Hinton, L. Deng, D. Yu, G.E. Dahl, Mohamed A., 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, pp. 82–97, November 2012.

[9] O. Abdel-Hamid, A-R. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*. IEEE, 2012, pp. 4277–4280.

[10] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012.

[11] M.M. Hochberg, S.J. Renals, and A.J. Robinson, "Abbot: The CUED hybrid connectionist-HMM large-vocabulary recognition system," *Spoken Language Systems Technology Workshop*, pp. 102–5, 1994.

[12] A.W. Senior and F. Fallside, "Offline handwriting recognition by recurrent error propagation networks," *International Workshop on Frontiers in Handwriting Recognition*, vol. 93, 1993.

[13] P. Chou, "Optimal partitioning for classication and regression trees," *IEEE PAMI*, vol. 13, no. 4, pp. 340–354, 1991.

[14] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. ARPA Human Language Technology Workshop*, 1994.

[15] 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, "On rectified linear units for speech processing," in *Proc. ICASSP*, 2013.

[16] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*, 2009.