ACOUSTIC MODELING WITH NEURAL GRAPH EMBEDDINGS

Yuzong Liu, Katrin Kirchhoff

Department of Electrical Engineering
University of Washington, Seattle, WA 98195

ABSTRACT

Graph-based learning (GBL) is a form of semi-supervised learning that has been successfully exploited in acoustic modeling in the past. It utilizes manifold information in speech data that is represented as a joint similarity graph over training and test samples. Typically, GBL is used at the *output* level of an acoustic classifier; however, this setup is difficult to scale to large data sets, and the graph-based learner is not optimized jointly with other components of the speech recognition system. In this paper we explore a different approach where the similarity graph is first embedded into continuous space using a neural autoencoder. Features derived from this encoding are then used at the *input* level to a standard DNN-based speech recognizer. We demonstrate improved scalability and performance compared to the standard GBL approach as well as significant improvements in word error rate on a medium-vocabulary Switchboard task.

Index Terms— Acoustic modeling, deep neural networks, graph-based learning

1. INTRODUCTION

Deep neural network (DNN) based classification has been shown to achieve superior performance on large-scale conversational automatic speech recognition (ASR) tasks by many research groups[1, 2, 3, 4, 5, 6, 7], and has become the state-of-the-art paradigm for acoustic modeling. In our own recent work [8, 9], we have explored complementary machine learning techniques that can further improve DNN-based acoustic modeling. In particular, we have proposed the integration of graph-based semi-supervised learning into DNN-based acoustic modeling, which has consistently resulted in significant improvements in frame classification accuracy and reductions in word error rates on several acoustic modeling and ASR tasks.

Graph-based learning (GBL) jointly models labeled (training) and unlabeled (test) data as a weighted graph. The nodes of the graph represent data samples and the edge weights encode pairwise similarities between samples. Graph-based semi-supervised learning (SSL) methods assume that the data distribution follows a low-dimensional manifold (represented by the graph) and constrain the label assignment to vary smoothly along the manifold. This means that samples that are close to each other in the graph (connected by an edge with a high weight) are encouraged to receive the same labels, whereas samples that are distant from each other are more likely to receive different labels. By exploiting smoothness constraints we can often achieve more accurate classifications as well as an implicit adaptation to the test data. Unlike standard supervised classification models used in acoustic modeling (including DNN-based classification), this method seeks to exploit *similarities between different test samples* in addition to similarities between training and test samples. It thus provides complementary information beyond that contained in the training data.

Previous applications of GBL to acoustic modeling [10, 11, 12, 13, 8] have shown considerable improvements in frame-level classification accuracy. More recently [9] we reported improvements in both frame-level accuracy and word error rate over a standard DNN baseline system. However, our previous work has also identified two main drawbacks of the standard GBL setup: First, although the frame-level classification accuracy consistently improves it does not always result in decreases in word error rate; rather, word error rates only decrease when the frame-level improvement is substantial. In the standard setup the graph-based learner is trained and applied independently at the output level of a DNN-based HMM state classifier. The resulting modified posterior distributions are then combined with the original posteriors and are used for lattice rescoring. Thus, the GBL is not optimized jointly with other system components. Second, graph construction incurs a high computational cost, which limits the scalability to large data sets.

In this paper, we therefore propose a different framework for integrating GBL, where the information provided by the graph is used at the *input level* to the DNN-based classifier: by embedding the similarity graph into continuous space through an autoencoder we learn novel features that are then used directly in the input layer to the DNN. The DNN learner can thus implicitly combine different information sources in the best possible way. While neural graph embeddings have been utilized previously in clustering [14] and in natural language understanding [15], their use in acoustic modeling is to our knowledge novel. In addition, we address the computational issues by landmark-based graph construction, which reduces

the cost by orders of magnitude while yielding good performance.

The paper is organized as follows: in Section 2, we give a brief overview of the standard GBL framework for DNN-based acoustic modeling; in Section 3, we explain the graph embedded features; in Section 4, we describe the data and baseline systems; in Section 5 we show the experimental results of the proposed method, and we conclude in Section 6.

2. GRAPH-BASED SSL FOR ACOUSTIC MODELING

In this section, we give a brief review of standard graph-based SSL as used in [9]. We define the following notation: We have a set of labeled data $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{r}_i)\}_{i=1}^{l}$, and a set of unlabeled data $\mathcal{U} = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$. $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional acoustic feature vector for a given sample (frame) i. For acoustic modeling, $\mathbf{x}$ is usually an MFCC or PLP feature vector, possibly with additional preprocessing steps such as splicing, LDA or MLLR transformations, etc. $\mathbf{r}_i \in \mathbb{R}^m$ is the reference distribution over class labels (e.g., phone labels, HMM state labels, etc.) for sample i. l and u are the number of frames in the labeled (training) and unlabeled (test) set, respectively. We construct a graph $G = \{V, E, W\}$ over all samples from $\mathcal{L}$ and $\mathcal{U}$, where V is the set of nodes in the graph (representing samples in $\mathcal{L} \cup \mathcal{U}$), E is the set of edges that connect a pair of nodes i and j, and each edge is associated with a weight w_{ij} that measures the similarity between them (an example is shown in Figure 1). The goal is to infer the class label distributions for all samples in $\mathcal{U}$ according to an objective function defined on the graph G.

We use a learning procedure called *prior-regularized measure propagation (pMP)* [16, 8], which minimizes the following objective function:

$$\begin{aligned}\mathcal{L}(G) \quad = \quad & \sum_{i=1}^{l} D_{KL}(\mathbf{r}_i||\mathbf{p}_i) + \mu \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} w_{ij} D_{KL}(\mathbf{p}_i||\mathbf{p}_j) \\ & + \nu \sum_{i=1}^{n} D_{KL}(\mathbf{p}_i||\tilde{\mathbf{p}}_i) \qquad (1)\end{aligned}$$

Here, $\{\mathbf{r}_i\}$ is reference label distribution, $\{\mathbf{p_i}\}$ is the predicted label distribution we want to optimize, and $\{\tilde{\mathbf{p}}_i\}$ is a prior label distribution. $D_{KL}(\cdot, \cdot)$ is the Kullback-Leibler divergence for a pair of probability distribution. w_{ij} is the similarity between sample i and j. $\mathcal{N}_i$ is the set of nearest neighbor of sample i. The first term of this objective function ensures that the predicted probability distribution matches the given distribution on labeled vertices; the second term enforces a smooth label assignment along the graph, and the third term penalizes divergence of the predicted label distribution from some prior distribution. Inference can be achieved using alternating minimization.

The graph is a k nearest-neighbor (kNN) graph constructed using fast kNN search as described in [17, 10] and symmetrization, such that if sample i is one of the nearest neighbors of sample j, then we also add j to the nearest neighbors of sample i. Nearest neighbors are obtained from both $\mathcal{L}$ and $\mathcal{U}$. In [9] we compared various feature representations for the kNN search and showed that bottleneck layer features obtained from a DNN yielded the best frame-level HMM state classification accuracy as well as word error rate. After graph construction we run inference on the graph and derive a new set of HMM state posteriors. We convert the posteriors into likelihoods by dividing by the prior HMM state distribution and combine the scores with the regular acoustic and language model scores for lattice rescoring. In [9] we showed that this procedure resulted in reduced word error rates on small to medium-vocabulary recognition tasks, and that it compared favorably to other semi-supervised learning schemes like self-training.

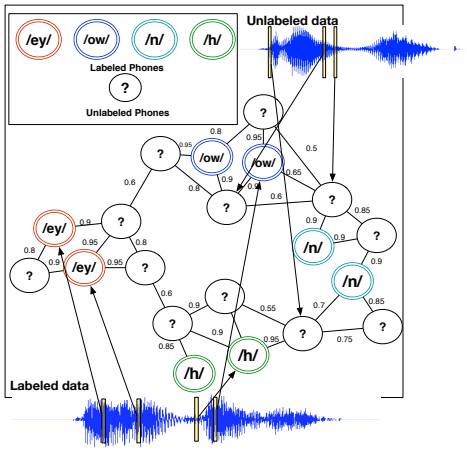

Fig. 1. Example of a data graph for GBL in acoustic modeling.

However, there are several important issues worth pointing out. First, graph-based SSL in its basic form is computationally complex since it requires the construction and storage of a huge graph built over all training and testing samples. For instance, a 100-hour speech task contains 40 million frames which is a severe bottleneck, even with fast and approximate nearest neighbor search approaches [18, 19]. Large graphs also lead to inefficient inference during test-time. The naive approach to solve graph-based SSL has $O(n^3)$ complexity; even with iterative solvers, inference can still be prohibitive. Approaches to alleviating the computational cost include selecting only a subset of the training data, either randomly or according to a principled objective function (e.g., [20, 21, 22]) or clustering data points and selecting cluster centroids as representative samples. Second, standard graph-based SSL is applied to the output of a DNN classifier and attempts to improve its accuracy. In past experiments we have observed that this usually leads to significant improvements in frame-level accuracy; however, the word error rate does not always reflect these improvements since not all system components

are optimized jointly.

3. GRAPH-EMBEDDING FEATURES

In this section we describe how the drawbacks described above can be addressed by utilizing "graph-embedding features", i.e., continuous features derived from a similarity graph that can be used directly as input to a standard DNN-based classifier in a speech recognition system.

3.1. Related Work

Graph embeddings have recently been proposed in several related fields. In [14], a deep stacked autoencoder is trained to learn compact graph-embedding features for the purpose of clustering. In that work, the input to the autoencoder consists of the row vectors of the normalized matrix representing the similarity graph: define the similarity matrix as $\mathbf{S} \in \mathbb{R}^{n \times n}$, and the degree matrix as $\mathbf{D} = diag\{d_1, d_2, \cdots, d_n\}$, where d_i is the degree of node i. The resulting input vector for sample i is given as $\frac{1}{d_i}\mathbf{S}_i$, where $\mathbf{S}_i$ is the ith row vector of $\mathbf{S}$. After training, the hidden layers of the autoencoder contain a shared, compact representation of the entire graph. The nonlinear activation outputs from the last hidden layer are extracted as the graph-embedding features, which are then used as inputs to a k-means clustering algorithm. The authors also show a theoretical connection between deep neural graph embeddings and spectral clustering. In [15], the authors use neural embeddings of a semantic knowledge-graph for the purpose of semantic parsing.

The framework in [14] is not directly scalable to our problem because of the huge construction overhead of the similarity matrix $\mathbf{S}$. First, as described above using the dense (or even the kNN) similarity graph would require too much computation for large data sets. Second, the similarity graph in [14] is built over one global data set. However, we would like to avoid building a similarity graph and subsequently retraining the autoencoder and the DNN classifier for every new test data set that we encounter, which would become impractical in real world acoustic modeling task.

3.2. Training Data Set Compression

Our goal is to learn graph-embedding features in a computationally feasible way. In previous work on manifold learning [23] it was shown how large-scale graph construction can be rendered manageable by selecting a much smaller set of "landmark" samples instead of the entire set of training samples.[1] Landmarks can be obtained by uniform subsampling of the original data set, or by k-means clustering and using the resulting cluster centroids as representative samples. We use the latter in this study. Thus, given a query sample we perform nearest neighbor search using a set of landmarks in each set rather than searching in the entire labeled training set and unlabeled set.

[1]The use of the term "landmark" here is unrelated to acoustic-phonetic landmarks.

3.3. Graph-Embedding Features for Acoustic Modeling

Rather than training graph embeddings from the raw similarity matrix, we provide additional information about the nearest neighbors to the autoencoder. Starting from the concept of a kNN graph, we denote the size-k set of nearest neighbors of node v_i as $\mathcal{N}_{v_i}$. We define a **neighborhood encoding vector** $\mathbf{n}_i$ that contains information about $\mathcal{N}_{v_i}$ as:

$$\mathbf{n}_i = [\mathbf{l}_1, \mathbf{l}_2, \cdots, \mathbf{l}_k] \;\; \forall j[j = 1, ..., k \rightarrow v_j \in \mathcal{N}_{v_i}] \qquad (2)$$

Here, $\mathbf{l}_j$ is an encoding vector of the class label distribution associated with node v_j. We compare the following two versions for $\mathbf{l}$:

One-hot vector representation: This is a binary vector representation with 1 representing the top-scoring class and 0 for all other classes. For training samples, the top-scoring class is determined by forced alignment of the training transcriptions with the acoustic data; for test samples, they are determined by a first-pass decoding output. Rather than utilizing HMM state labels, which would result in vectors of high dimensionality, we use phone classes.

Soft-label representation: This is the full probability distribution over class labels obtained from a supervised classifier. We can either use the DNN classifier from our baseline ASR system to compute the posteriors, or we can train a separate classifier. In our case we train a separate simple phone classifier using a subset of the entire data, which is run concurrently to the DNN.

Having obtained the neighborhood encoding vectors for the training samples we train an autoencoder to map them to a more compact representation (see Figure 2). The input to the autoencoder consists of the neighborhood encoding vectors $\mathbf{n}_1, ..., \mathbf{n}_l$, i.e., the autoencoder is trained on the training samples only. We thus depart from previous graph embedding approaches in that the autoencoder is not trained over all samples jointly; this is to avoid having to retrain the encoder for each new test set. The autoencoder maps an input vector to a hidden representation via a nonlinear transformation, defined as $\mathbf{g} = s(\mathbf{W}^{(1)}\mathbf{n} + \mathbf{b}^{(1)})$. The hidden representation is then passed through another nonlinear transformation to reconstruct the input, i.e. $\tilde{\mathbf{n}} = s(\mathbf{W}^{(2)}\mathbf{g} + \mathbf{b}^{(2)})$. $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ are the parameters of the network and s is a nonlinear activation function. To train the autoencoder, we minimize the reconstruction error $||\mathbf{n} - \tilde{\mathbf{n}}||_2$. The nonlinear activation output $\mathbf{g}$ provides the graph-embedding features. For the test samples, the nearest neighbors are drawn from the training and test set and their neighborhood encoding vectors are then passed through the trained autoencoder separately to derive $\mathbf{g}^{\mathcal{L}}$ and $\mathbf{g}^{\mathcal{U}}$.

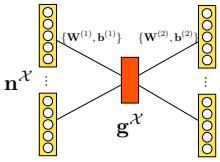

Fig. 2. Autoencoder mapping of neighborhood encoding vectors.

Two possible methods for using the graph-embedding features as inputs to the DNN in a DNN-HMM ASR system are shown in Figure 3. In this figure, $\mathbf{a}$ corresponds to the original acoustic features; $\mathbf{g}^{\mathcal{L}}$ and $\mathbf{g}^{\mathcal{U}}$ correspond to the graph-embedding features for the labeled data (training) and unlabeled data (test). For each test sample we find k nearest neighbors in the labeled set $\mathcal{L}$, and k nearest neighbors in the unlabeled set $\mathcal{U}$. We extract graph-embedding features $\mathbf{g}^{\mathcal{L}}$ and $\mathbf{g}^{\mathcal{U}}$ separately and append them as shown in Figure 3. In addition, we append $\mathbf{s}^{\mathcal{L}}$ and $\mathbf{s}^{\mathcal{U}}$, which are the similarity values for the nearest neighbors in sets $\mathcal{L}$ and $\mathcal{U}$. For training samples, $\mathbf{g}^{\mathcal{L}}$ and $\mathbf{s}^{\mathcal{L}}$ are simply duplicated to achieve the same dimensionality.

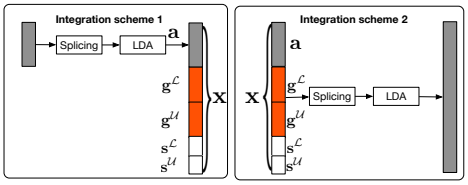

Fig. 3. Two different ways of integrating graph-embedding features during DNN training.

Figure 4 summarizes the proposed framework. Given training data $\mathcal{L}$ and test data $\mathcal{U}$, we first extract separate sets of landmarks using k-means. We also select a subset of the training data (10%) to train a simple MLP to produce the soft-label representation of $\mathbf{n}$. For the training data, kNN search is performed using the landmarks of $\mathcal{L}$; for the test data, kNN search is performed using landmark sets of both $\mathcal{L}$ and $\mathcal{U}$. The autoencoder is trained using $\mathbf{n}_1, ..., \mathbf{n}_l$ and is applied to $\mathbf{n}_1, ..., \mathbf{n}_{l+u}$.

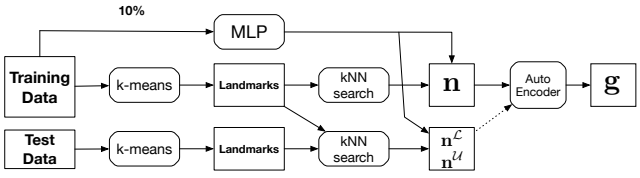

Fig. 4. Procedure for extracting graph-embedding features.

Table 1. Word error rates of DNN-based baseline system for the SVitchboard-II 10k task.

System	Dev	Eval
Baseline System	32.04%	32.17%

4. DATASETS

We evaluate the proposed framework on the SVitchboard-II dataset [24], which can be downloaded from `http://tinyurl.com/hqlc-speech`. SVitchboard-II is a set of high-quality, low-complexity conversational English speech corpora created from the original Switchboard-I dataset. To create these corpora, subsets of Switchboard-I were selected that are of high quality (acoustically representative) and low complexity (i.e., smaller vocabulary size). We use the largest-vocabulary task in SVitchboard-II for this study, which has a vocabulary size of 9983. The training, development, and test sizes are 67642, 8491, and 8503 utterances (69.1 hours, 8.8 hours and 8.8 hours), respectively. A trigram backoff language model built on the training data is used for decoding.

To train the baseline system we first train a triphone GMM-HMM system. We train a monophone GMM with flat start, using 13 MFCCs and their deltas and delta-deltas (MFCC+Δ+$\Delta\Delta$). Cepstral mean normalization is performed for each conversation side. After the monophone system has been trained, we use it to train a context-dependent GMM triphone model on top of it. The baseline DNN-HMM system is then bootstrapped from this triphone GMM-HMM system: we create a 4-layer network with 1024 nodes per layer. The total number of senones in the DNN is 1864. We perform greedy layer-wise supervised training [25, 1] on multiple CPUs and GPUs using the Kaldi toolkit [26]. The input features are spliced MFCCs (with a context window size of 4), followed by an LDA transformation (without dimensionality reduction) which is used to decorrelate the input features. The resulting feature vector has 117 dimensions in total. We use 20 epochs to train the DNN, with a mini-batch size of 256. For the first 15 epochs, we decrease the learning rate from 0.01 to 0.001 and fix the learning rate at 0.001 for the last 5 epochs. Table 1 shows the baseline DNN-HMM system's word error rate.

5. EXPERIMENTAL RESULTS

5.1. Standard GBL Baseline

We first conduct an experiment using the standard GBL framework as described in Section 2. In line with [9] we use the bottleneck layer features from a trained DNN and splice them with a context window of 9 (i.e. $\pm$ 4). The resulting features are used to build a kNN graph (excluding silence frames) of $\mathcal{L}$ and $\mathcal{U}$. Because the number of non-silence

frames in $\mathcal{L}$ is around 17 millions, we perform uniform subsampling to select 10% of the samples from $\mathcal{L}$. For each sample in the unlabeled set $\mathcal{U}$ we select 10 nearest neighbors in $\mathcal{L}$ and 10 in $\mathcal{U}$. The pMP algorithm is then run on the resulting graph, and a new set of HMM state posteriors is obtained. The new posteriors are converted to acoustic likelihoods by dividing the HMM state prior distribution. These new likelihoods are then interpolated with the original acoustic and language model scores for lattice rescoring. The parameters of pMP are $\mu = 10^{-6}$ and $\nu = 8 \times 10^{-6}$; the linear score combination weights are 0.2, 0.8, 8 for the GBL score, acoustic model score and language model score, respectively. All parameters are tuned on the development data.

5.2. Graph-Embedding Experiments

For the graph-embedding experiment we use the same bottleneck layer feature representation of the data as in the preceding section. The number of frames in the training data is around 25 million. We extract landmarks from this set using k-means clustering. First, we assign each frame a phone label based on the forced alignment of training transcriptions. For silence frames we run k-means with $k = 64$. For the remaining non-silence frames, we run k-means with $k = 32$ for each of the 42 non-silence phone classes. The resulting centroids are used as landmarks, resulting in 1408 landmarks, which is 4 orders of magnitude less than the original graph size. For each set (training, development and test), we create a separate set of landmarks. We perform kNN search using the landmark sets. The similarity measure we used is an RBF kernel with Euclidean distance. For the training samples we select 10 nearest neighbors in the landmark set of $\mathcal{L}$. For the test data we select 10 nearest neighbors in the landmark sets of $\mathcal{L}$ and $\mathcal{U}$, respectively.

To extract the neighborhood encoding vectors we find the 10 nearest neighbors in the landmark sets of $\mathcal{L}$ and $\mathcal{U}$, respectively. To generate the label-encoding vector $\mathbf{l}$, we experiment with both a one-hot and a soft-label vector representation. The labels are those of the 43 phone classes. For a one-hot encoding the frame-level forced alignment information is used for the training data and the first-pass decoding output for the test data. To create soft-label neighborhood encoding vectors, we train a simple 3-layer multi-layer perceptron (MLP) with ℓ_2 regularization for phone classification using the Theano [27] toolkit. The hidden layer of the MLP consists of 2000 nodes. The nonlinearity function in the MLP is the *tanh* function. We train the MLP on a random 10% subset of the entire training data using stochastic gradient descent with a minibatch size of 128. The learning rate of the MLP is 0.01 and the total number of epochs is 1000. The regularization constant is set to 0.001. We use early stopping during training. The MLP has a classification accuracy of 67% on a 1.2 million held-out dataset. The MLP outputs are then used to generate phone label distributions for the landmark samples. The dimensionality of the resulting neighborhood encoding vectors is 430 (43 phone classes and 10 nearest neighbors). After obtaining the neighborhood encoding vectors for the training samples we train the autoencoder and extract the hidden layer activation outputs as the graph-embedding features for $\mathcal{L}$. The graph-embedding features for the test set $\mathcal{U}$ are generated by passing the neighborhood encoding vectors for $\mathcal{U}$ through the same autoencoder (without retraining). We experimented with different dimensionalities of the graph-embedding features (43 vs. 100 nodes in the hidden layer, respectively). The DNN classifier for the ASR system is then trained using either of the two integration methods described above in Section 3.

5.3. Results

Table 2 shows the results for different experiments. We first compared the DNN baseline system with the standard GBL method. We then compared graph-embedding features with different nearest neighbor encoding schemes, viz. (1) one-hot label distribution vector (experiments #1, #2, #3) and (2) soft-encoding label distribution vector (experiments #5, #6). We also compared different sizes for the hidden layer in the autoencoder (43 vs. 100 hidden units). As another point of comparison, we ran an experiment where we directly used the neighborhood encoding vectors in the input layer of the DNN, instead of first passing them through the autoencoder.

Table 2. Experiment Setup.

Setup	Description
#1	One-hot; 43-d; integration scheme 2
#2	One-hot; 100-d; integration scheme 2
#3	One-hot; 43-d; integration scheme 1
#4	kNN w/o autoencoder; integration scheme 2
#5	Soft label; 43-d; integration scheme 2
#6	Soft label; 100-d; integration scheme 2

Table 3 shows the word error rates (WER) on the development and/or test sets using the different setups. Boldface numbers indicate statistically significant improvements ($p < 0.05$). The standard output-level GBL method only resulted in a trivial decrease in word error rate, although the frame-level HMM state classification accuracy (measured on all non-silence frames in the development data) increased from 32.5% in the baseline system to 38.2%. As discussed above, a gain in frame accuracy under this scheme does not always lead to a reduced WER. In Experiments 1, 2 and 3, we obtained larger but non-significant improvements in word error rate by using the graph-embedding features learned from the one-hot label-encoding vectors. The reason why improvements were not larger may be due to the way we derived the one-hot encodings: the labels of the training data were derived using forced alignments whereas the labels of the test

data were derived using first-pass decoding output. The discrepancy between the label accuracies could be a contributing factor. In Experiment #4, which utilizes neighborhood encoding vectors in the DNN input layer directly, we again have only a slight improvement. Using the soft label-encoding vectors, however, (#5 and #6), we achieved an absolute improvement in word error rate of 1.5%. The dimensionality of the hidden layer in the autoencoder (Experiment #1 vs. #2 and #5 vs. #6) does not have a consistent effect.

Table 3. Word error rates for standard GBL framework vs. graph-embedding features.

System	Dev	Test
DNN Baseline	32.04%	32.17%
+standard GBL	31.95%	32.13%
#1	**31.65%**	32.05%
#2	31.76%	31.98%
#3	31.76%	32.00%
#4	31.75%	**31.79%**
#5	**30.57%**	**30.59%**
#6	**30.61%**	**30.71%**

To further evaluate the performance of the graph-embedding features we subsequently used several stronger baseline systems (Table 4). First, we increased the number of hidden units per layer in the DNN from 1024 to 2048. In a separate experiment we also used maxout [28], which tends to yield better performance. For the maxout network, we used a group size of 5 and set the number of groups to 800. The value of p of the p-norm was set to 2. For both systems, we augmented the original acoustic features with the graph-embedding features following the setup in Experiment #5. We again see strong improvements from utilizing graph-embedding features with larger networks (first 2 rows of Table 4), with larger relative improvements than before. Maxout by itself already significantly improves over the baseline DNN performance; however, the combination with graph-embedding features still reduces the WER further by about 0.5% absolute. We also evaluate the performance of graph-embedding features on top of a speaker-dependent DNN system. The input features to the DNN system are 40-dimensional fMLLR features. First, the spliced MFCC features ($\pm$ 4 frames) are processed using LDA followed by a semi-tied covariance transform. Then, the fMLLR transform is applied to the features for each conversation side. Table 4 shows that graph-embedding features on top of the speaker-dependent system also achieve consistent WER reductions.

Finally, Table 5 shows the results of an experiment where we controlled for the overall number of parameters. The input layer of the DNN when using the graph-embedding features is generally larger than when using only acoustic features, resulting in more parameters in the new DNN system. Hence, we increased the size of the baseline DNN by increasing the number of nodes in the hidden layer, and made the number of parameters roughly the same for both networks. To do this, we increased the number of nodes per layer from 1024 to 1145, resulting in 6.16 million parameters in total for the baseline DNN. The DNN system in Experiment #5 is 6.15 million. Comparing the two systems, the system using graph embedding gave a 1.4% absolute improvement on the test set.

Table 4. WER comparison with different baseline system.

	Dev	Test
2048, 4-layer DNN	31.23%	31.31%
+ graph embedding	**29.59%**	**29.72%**
Maxout	30.63%	30.86%
+ graph embedding	**30.18%**	**30.32%**
SD-DNN	29.96%	30.01%
+ graph embedding	**29.40%**	**29.60%**

Table 5. WER comparison controlled for the number of parameters.

	Dev	Test
1145, 4-layer DNN, # params: 6.16M	31.89%	32.01%
+ graph embedding, # params: 6.15M	**30.57%**	**30.59%**

Computational efficiency: We also highlight the efficiency of the framework. Graph construction for the standard output-level GBL method can take several hours or up to days depending on the computational resources. The landmark-based construction method proposed in this paper uses only thousands of landmarks and thus achieves a speed-up of 3-4 orders of magnitude. Although several steps are involved in generating graph-embedding features (Figure 4), there is very little computational overhead: the training time of the MLP on one GPU is around 19 minutes; the training time of the autoencoder on one GPU is 110 minutes. Furthermore, these steps are only done once as part of the training process. The efficiency makes this framework appealing for real-world acoustic modeling task.

6. CONCLUSION

We have shown that graph-embedding features learned from nearest-neighbor encodings yield significant improvements on a 10k-vocabulary conversational ASR task. Future directions include and scaling this approch to larger vocabularies and exploring deep neural embeddings using stacked autoencoders. Our framework might also be used to improve recognition performance on low-resource languages, or as an adaptation method to novel acoustic environments.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of Tesla K40 GPUs used for this research.

7. REFERENCES

[1] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks.," in *Proceedings of Interspeech*, 2011.

[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*, vol. 29, no. 6, pp. 82–97, 2012.

[3] 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.

[4] Tara N. Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proceedings of ICASSP*. IEEE, 2013, pp. 8614–8618.

[5] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of ICASSP*. IEEE, 2013, pp. 6645–6649.

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

[7] George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny, "The IBM 2015 English conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[8] Yuzong Liu and Katrin Kirchhoff, "Graph-based semi-supervised learning for phone and segment classification," in *Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013.

[9] Yuzong Liu and Katrin Kirchhoff, "Graph-based semi-supervised acoustic modeling in DNN-based speech recognition," in *Proceedings of the IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 177–182.

[10] A. Alexandrescu and K. Kirchhoff, "Graph-based learning for phonetic classification," in *Proceedings of ASRU*, 2007.

[11] A. Subramanya and J.A. Bilmes, "Entropic graph regularization in non-parametric semi-supervised classification," in *Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2009.

[12] A. Alexandrescu and K. Kirchhoff, "Phonetic classification by controlled random walks," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2011.

[13] M. Orbach and K. Crammer, "Transductive phoneme classification using local scaling and confidence," in *Proceedings of IEEE 27th Convention of Electric and Electronics Engineers in Israel*, 2012.

[14] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu, "Learning deep representations for graph clustering," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[15] Larry Heck and Hongzhao Huang, "Deep learning of knowledge graph embeddings for semantic parsing of Twitter dialogs," in *Proceedings of the 2nd IEEE Global Conference on Signal and Information Processing*, 2014.

[16] A. Subramanya and J. Bilmes, "Semi-supervised learning with measure propagation," Tech. Rep. UWEE-TR-2010-0004, University of Washington, 2010.

[17] Andrew B Goldberg and Xiaojin Zhu, "Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization," in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, 2006, pp. 45–52.

[18] Jie Chen, Haw-ren Fang, and Yousef Saad, "Fast approximate k-nn graph construction for high dimensional data via recursive Lanczos bisection," *Journal of Machine Learning Research*, vol. 10, pp. 1989–2012, 2009.

[19] Wei Dong, Charikar Moses, and Kai Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th International Conference on World Wide Web*. ACM, 2011, pp. 577–586.

[20] Yuzong Liu, Kai Wei, Katrin Kirchhoff, Yisong Song, and Jeff Bilmes, "Submodular feature selection for high-dimensional acoustic score space," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013.

[21] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes, "Submodular subset selection for large-scale speech training data," *Proceedings of ICASSP*, 2014.

[22] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes, "Unsupervised submodular subset selection for speech data," in *Proceedings of ICASSP*, 2014.

[23] Vin De Silva and Joshua B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems*, 2002, pp. 705–712.

[24] Yuzong Liu, Rishabh Iyer, Katrin Kirchhoff, and Jeff Bilmes, "SVitchboard II and FiSVer I: High-quality limited-complexity corpora of conversational English speech," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Dresden, Germany, September 2015.

[25] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al., "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153, 2007.

[26] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011, pp. 1–4.

[27] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[28] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2014.