INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Novel Front-End Features Based on Neural Graph Embeddings for DNN-HMM and LSTM-CTC Acoustic Modeling

Yuzong Liu [1*], *Katrin Kirchhoff* [2]

[1] Amazon.com, Seattle
[2]Department of Electrical Engineering, University of Washington, Seattle
{yzliu,kk2}@uw.edu

Abstract

In this paper we investigate neural graph embeddings as front-end features for various deep neural network (DNN) architectures for speech recognition. Neural graph embedding features are produced by an autoencoder that maps graph structures defined over speech samples to a continuous vector space. The resulting feature representation is then used to augment the standard acoustic features at the input level of a DNN classifier. We compare two different neural graph embedding methods, one based on a local neighborhood graph encoding, and another based on a global similarity graph encoding. They are evaluated in DNN-HMM-based and LSTM-CTC-based ASR systems on a 110-hour Switchboard conversational speech recognition task. Significant improvements in word error rates are achieved by both methods in the DNN-HMM system, and by global graph embeddings in the LSTM-CTC system.

Index Terms: acoustic modeling, graphs, semi-supervised learning, deep neural networks

1. Introduction

Deep neural networks (DNNs) [1, 2] are being widely used in acoustic modeling for large vocabulary conversational speech recognition (LVCSR) systems. A feedforward DNN is a multilayer perceptron with multiple hidden layers. It is used to map an acoustic feature vector (such as MFCC or fMLLR feature vectors) to target labels, typically context-dependent HMM states (or *senones*) that are derived from a forced alignment of the speech signal and training transcriptions. More recently, studies have shown that more advanced deep models, such as recurrent neural networks (RNNs) [3], can improve over feed-forward DNNs. An RNN has recurrent connections within the hidden layers, thus allowing temporal information to be stored inside the network. Among these, long short-term memory networks (LSTMS) have been shown to outperform conventional RNNs [4, 5]. LSTM is a special version of an RNN with memory cells that store temporal information. Each cell consists of multiple gate logic units, which are used to add new information, forget old information and decide on the output information.

In addition, recent attempts at end-to-end speech recognition using connectionist temporal classification (CTC) [6, 7, 8, 9] have received much attention. CTC uses an objective function [10] for sequence labeling problems involving variable-length input/output sequences that do not have to be of the same length. Unlike training DNNs or RNNs in combination with a hidden Markov model (HMM), CTC does not rely on frame-level training targets derived from a forced alignment and automatically learns the alignments between acoustic feature vectors and labels.

In a complementary line of work [11, 12, 13] we have shown that graph-based semi-supervised learning can provide useful additional information for DNN-based acoustic models. In an early lattice-based rescoring approach to integrating graph-based information [12], all training and test samples were represented as nodes on a weighted graph. To infer senone distributions of the test samples on the graph, label information was propagated from the training samples to the test sample by minimizing an energy function on the graph. The new senone posteriors were converted to likelihoods and were linearly interpolated with other scores for the purpose of lattice rescoring. However, the graph construction process required by this framework quickly becomes computationally prohibitive for LVCSR tasks. To alleviate computational overhead and achieve better performance we have recently proposed a novel neural graph embedding approach to acoustic modeling [13]. Instead of utilizing a huge graph directly, we map the neighborhood information for each sample into a compact feature representation using an autoencoder. These compact feature representations are then concatenated with the original acoustic feature vectors (i.e. MFCC or fMLLR feature), and are fed into the DNN input layer. We refer to this type of neural graph embedding features as **neighborhood graph embedding features** or NGE features.

In this paper, we present several advances over this method. First, we enrich the NGE features as described above with speaker-dependent information. Second, we propose an alternative type of neural graph embedding features called **similarity graph embedding features**, or SGE features. Unlike the NGE features, which only encode local information for each sample, the similarity graph embedding features capture the global manifold information for each sample. We show considerable improvements over the NGE features on LVCSR tasks. Third, in addition to DNN-HMM systems, we also test similarity graph embedding features as front-end features in an LSTM-CTC-based LVCSR system and again show improvements in word error rate over both standard acoustic features and NGE features.

The remainder of the paper is organized as follows: in Section 2 we describe our previous framework on NGE features and propose an improved NGE feature extraction using speaker-dependent information. In Section 3, we describe the related work on encoding similarity matrix and explain our novel SGE features. In Section 4, we describe the baseline system and datasets for this work and provide experimental results comparing both graph embedding methods. Section 5 concludes.

∗Work was performed while the author was with the University of Washington.

http://dx.doi.org/10.21437/Interspeech.2016-542

2. Local Neighborhood Graph Embeddings

In our previous work [13], we introduced neural graph embedding features that encode local neighborhood information for each sample (speech frame). The objective was to enrich the standard acoustic feature vectors with information about the most similar speech samples. Given an acoustic classifier that produces probability distributions over a set of labels (e.g., HMM states) for each frame, we define the **neighborhood encoding vector**, or n-vector, for a given frame i as follows:

$$\mathbf{n}_i = \mathbf{l}_1 \circ \mathbf{l}_2 \circ \cdots \circ \mathbf{l}_k$$

where $\mathbf{l}_j$ is the label distribution of the j-th nearest neighbor of sample i, and $\circ$ is a vector concatenation symbol. To generate the $\mathbf{l}$ vectors, we trained a simple MLP classifier on monophone targets. In order to select the k nearest neighbors for a given frame we chose a representative subset of "landmarks" from the training and test data and only performed kNN selection against these, rather than using the entire set of speech samples. Landmarks (see further Sec.3.2) are a subset of representative feature vectors that are intended approximate the entire data set. We describe the generation of NGE features as follows. Notationally, we define our training set as $\mathcal{L}$ and the test set as $\mathcal{U}$. We first obtain a set of landmarks from $\mathcal{L}$ and $\mathcal{U}$, and refer to these as $\mathcal{Z}_\mathcal{L}$ and $\mathcal{Z}_\mathcal{U}$ respectively. Landmarks are obtained by k-means clustering of the data and extracting the cluster centroids. We then create n-vectors for all frames in $\mathcal{L}$ using landmark set $\mathcal{Z}_\mathcal{L}$. These n-vectors are then used to train an autoencoder: each n-vector constructed for $\mathcal{L}$ is passed to an autoencoder (a neural network with a single hidden layer trained to minimize the reconstruction error between input and output), and the hidden layer output $\mathbf{g}$ is extracted from the autoencoder. For each frame in $\mathcal{U}$, we create two n-vectors - one created using landmark set $\mathcal{Z}_\mathcal{L}$ and another created using landmark set $\mathcal{Z}_\mathcal{U}$. We can extract two graph embedding feature vectors $\mathbf{g}^\mathcal{L}$ and $\mathbf{g}^\mathcal{U}$ for each sample. To use these embedding features for DNN training, the features are further enriched with the similarity values for the k nearest neighbors. Thus, the final feature vectors take the following form:

$$\mathbf{x} = \mathbf{a} \circ \mathbf{g}^\mathcal{L} \circ \mathbf{g}^\mathcal{L} \circ \mathbf{s}^\mathcal{L} \circ \mathbf{s}^\mathcal{L} \text{(for training samples)} \quad (1)$$

$$\mathbf{x} = \mathbf{a} \circ \mathbf{g}^\mathcal{L} \circ \mathbf{g}^\mathcal{U} \circ \mathbf{s}^\mathcal{L} \circ \mathbf{s}^\mathcal{U} \text{(for test samples)} \quad (2)$$

Here $\mathbf{a}$ is the original acoustic feature vector (MFCC or fMLLR features). $\mathbf{s}$ is a vector that consists of similarity values to the nearest landmark points. We show in [13] that these neighborhood graph embedding features yield significant improvements over speaker-independent (SI) and speaker-dependent (SD) DNN-HMM systems.

2.1. Speaker-Dependent NGE (SD-NGE) Features

In our previous work we created landmarks globally from the entire training set and test data. In this work we investigate selecting landmarks on a per-speaker basis to obtain better improvements. Formally, we define two types of landmark sets. A *global landmark set* $\mathcal{Z}_{global}$ is extracted from the entire training data. This set is identical to the landmark set $\mathcal{Z}_\mathcal{L}$ defined in the previous NGE framework. In addition, a *local landmark set* $\mathcal{Z}_{local}$ is extracted on a per-speaker basis. In either case, we use k-means and extract the centroids as landmark points. For each frame in either training and test samples, we can create two types of n-vectors: the first n-vector $\mathbf{n}^{global}$ is created using kNN search against $\mathcal{Z}_{global}$, and the second n-vector $\mathbf{n}^{local}$ is created using kNN search against $\mathcal{Z}_{local}$. The resulting n-vectors are passed into the autoencoder to extract two graph embeddings features $\mathbf{g}^{global}$ and $\mathbf{g}^{local}$. Similar to our previous work we also create two similarity vectors $\mathbf{s}^{global}$ and $\mathbf{s}^{local}$ which consist of the similarity values to the neighboring landmark points. Let the original acoustic input vector to the DNN be $\mathbf{a}$, the final input feature to the DNN takes the following form: $\mathbf{x} = \mathbf{a} \circ \mathbf{g}^{global} \circ \mathbf{g}^{local} \circ \mathbf{s}^{global} \circ \mathbf{s}^{local}$

Compared to the previous NGE features [13], which are referred to as the SI-NGE features, the SD-NGE features, when applied to the DNN input layer, provide both global (speaker-independent) and local (speaker-dependent) information.

3. Global Similarity Graph Embeddings

The neighborhood embedding features described above are extracted by mapping graph neighborhood information to a continuous feature representation using an autoencoder. The neighborhood information is *local* - i.e., we only use the label information of the k nearest neighbors for a given frame. In this section, we describe a similarity graph embedding (SGE) feature that encodes *global* information for each sample.

3.1. Encoding the Similarity Matrix

Encoding a global data similarity matrix into continuous space has previously been studied in fields other than speech recognition. In [14] a deep autoencoder was trained to map a k-nearest neighbor graph into a low-dimensional continuous vector space for the purpose of data clustering. The input to the autoencoder for each sample consisted of the row vectors of the normalized similarity matrix representing the graph, and the nonlinear activation outputs from the last hidden layer were extracted as the 'graph embedding features'. These features were then used as inputs to a standard k-means clustering algorithm. Clustering with graph embedding features yielded better results than direct k-means clustering with the similarity graph row vectors or spectral clustering on the similarity graph. In [15], the authors used neural embeddings of a semantic knowledge-graph for the purpose of semantic parsing. In [16], the authors proposed a 'generalized autoencoder' which extends the traditional autoencoder by taking manifold information into the reconstruction term of the autoencoder training.

Following [14] a direct way to encode the manifold information would be to use the row vectors in a similarity matrix to train the autoencoder. This would require constructing a similarity graph over all training and test samples, followed by autoencoder training and feature extraction from the hidden layer(s). This approach cannot be used directly in LVCSR tasks. First, constructing a similarity graph over all training and test samples jointly is computationally expensive, even when approximate kNN search procedures are used. More importantly, the graph would have to be re-built, and the autoencoder would have to be retrained, for every new test set, which is impractical.

3.2. Similarity Matrix Approximation using Landmarks

To address the computational complexity issue we again use the concept of landmarks. In [17, 18] it was shown formally how a large similarity matrix can be approximated using a set of "landmarks". Given a similarity matrix $\mathbf{W}$, and a set of landmarks $Z = \{\mathbf{z}_i\}, i = 1, \cdots, m$, the similarity matrix can be approximated as follows:

$$\tilde{\mathbf{W}} = Z \Lambda^{-1} Z^\intercal$$

where $\tilde{\mathbf{W}}$ is the approximated similarity matrix of $\mathbf{W}$, $Z \in \mathbb{R}^{n \times m}$ is a low rank matrix, and Λ is a diagonal matrix with $\Lambda_{kk} = \sum_{i=1}^{n} Z_{ik}$. Z is a design matrix with Z_{ij} measuring the similarity between sample i and landmark j. According to Nadaraya-Watson kernel regression [19], Z_{ij} is defined as follows:

$$Z_{ij} = \frac{K_h(\mathbf{x}_i, \mathbf{z}_j)}{\sum_{\mathbf{z}' \in \mathcal{Z}} K_h(\mathbf{x}_i, \mathbf{z}')} \quad (3)$$

where $\mathcal{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$ is a set of landmarks, $K_h(\mathbf{x}_i, \mathbf{z}_j)$ is a kernel function defined on a given sample $\mathbf{x}_i$ and a landmark $\mathbf{z}_j$ (in our case, we use the RBF kernel). In this way, we can represent a large similarity matrix $\mathbf{W}$ using a much smaller low rank matrix Z. More importantly, it is not necessary to train new models for each new test set - all we need is a set of pre-computed landmark points.

To train the autoencoder we can subsequently use the row vectors in Z. Compared to using the row vectors of a full a similarity matrix, the complexity is reduced from $O(n^2)$ to $O(mn)$, where $m \ll n$. In our LVCSR experiments n is usually in the range of 10^7 whereas m is usually in the range of 10^3, which reduces the data size by 4 orders of magnitude. After the autoencoder has been trained, we can create the graph embedding features $\mathbf{g}$. These are then concatenated with the existing acoustic features $\mathbf{a}$: $\mathbf{x} = \mathbf{a} \circ \mathbf{g}$.

3.3. Comparison to Other Methods

NGE features are a somewhat ad-hoc representation of local neighborhood information. By contrast, the SGE features have a more solid theoretical foundation and a tight connection to spectral clustering. Spectral clustering aims at finding k eigenvectors that correspond to the smallest non-zero eigenvalues of the graph similarity matrix. These eigenvectors can be viewed as a low-dimensional embedding of the similarity matrix; they can be used as inputs to a standard clustering algorithms such as k-means. A graph autoencoder similarly finds a low-dimensional embedding by virtue of being trained to reconstruct the similarity matrix. In [14] the authors proved that the similarity graph embeddings and spectral clustering share a similar objective based on the Eckart-Young-Mirsky theorem [20].

The SGE features also bear some similarity to i-vectors [21, 22], a feature representation that is widely used for speaker identification and DNN adaptation. I-vectors are usually extracted using a universal background model (UBM-GMM) and are used to measure how an UBM-GMM should be adapted in an affine subspace in order to capture the variability of the underlying speech segments. I-vectors can be extracted at either utterance or speaker level. To use i-vectors for DNN training, they are duplicated across all frames for each utterance/speaker and appended to the acoustic feature vectors. Similar to SGE and NGE, the i-vector extractor needs to compute a set of UBM centroids, which is analogous to the landmark set we are using for graph embedding features. The difference is that i-vectors capture speaker variability at the speaker/utterance level; whereas the graph embedding features capture the manifold information at the frame level.

4. Experiments and Results

4.1. Data and Systems

For comparability with our previous work [13] we first evaluate the two graph embedding features on the SVitchboard-II dataset [23], which 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 acoustically representative while having a limited vocabulary size. We use the largest-vocabulary sub-task in SVitchboard-II for this study, which has a vocabulary size of 9983. We refer to this set as SVB-10k. 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. In addition, we also evaluate the framework using the larger 110-hour Switchboard-I task and report the performance on the Switchboard part of the Hub5-00 test set. We train a trigram backoff language model using the Switchboard corpus.

We use two different ASR architectures for evaluation. The first is a DNN-HMM system trained using Kaldi [24]. We train both a speaker-independent (SI) and a speaker-dependent (SD) version. The DNNs consist of 4 hidden layers with the *tanh* activation functions; the output layer uses a softmax function. The total number of output classes (senones) in the DNN is 1864 for SVB-10k and 2390 for Switchboard-I. For the SI system we use spliced MFCC features with a LDA transformation; for the SD system, we first create a speaker adaptive trained (SAT) GMM-HMM system, and use spliced fMLLR features as inputs to the DNN. The targets for DNN training are created using forced alignments from a SI-GMM-HMM and SAT-GMM-HMM system, respectively. 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.

The second architecture is a LSTM-CTC system trained using EESEN [25]. In the LSTM-CTC system we replace the DNN by a deep bi-directional LSTM as the acoustic model. The baseline LSTM networks consist of 2 hidden layers. Each layer has a forward and a backward sub-layer, where each sub-layer consists of 320 memory cells. For the baseline system we use 40-dimensional fMLLR features as inputs to the LSTM, with mean and variance normalization applied on a per-speaker basis. For CTC training, we use 42 phonemes plus one blank symbol as targets for SVB-10k, and 45 phonemes plus one blank symbol for Switchboard. We use 20 epochs to train the LSTM, with an initial learning rate of 0.00004. The learning rate is halved when the improvements are no longer significant.

4.2. Autoencoder Training

For both NGE and SGE feature extraction, we select a subset of landmark points. For NGE feature extraction, we extract a set of landmarks from all frames in the training set. We partition these frames according to their phone labels derived from forced alignments and run k-means for each partition. For silence frames, we set $k = 64$; for the other phonemes, we set $k = 32$. The resulting centroids are used as landmarks (1408 and 1504 landmarks in total for each task). These landmarks are also used for the SGE feature extraction. For SI-NGE feature, we extract the same number of landmarks from the test set; for SD-NGE features, instead of selecting landmarks from an entire test set, we extract 256 landmarks for each speaker. To generate the probability distributions for the n-vectors, we train a separate MLP classifier to produce monophone label distributions, as before (more details can be found in [13]). The autoencoder is a simple feedforward neural network with one hidden layer. We train the autoencoder by minimizing the ℓ_2 reconstruction loss on the training data. The inputs to the autoencoder for NGE

and SGE features are the n-vectors and the row vectors in the Z matrix, respectively. The number of hidden units is 43 and 46, for SVB-10k and Switchboard, respectively.

4.3. Experimental Results

We show experimental results comparing NGE features and SGE features on different tasks and systems. We first compare the original SI-NGE features (Row 2 and 5) and the SD-NGE features (Row 3 and 6) in Table 1. Using the SI-NGE features we reduce the word error rate (WER) by 1.5% and 0.8% on both tasks. With the SD-NGE features, we further reduce the WER by 2% and 1.5% absolute compared to SI baseline system (Row 1). It can be observed that, in a SD baseline system (Row 4), the SI-NGE features do not lead to improvements on WER; on the other hand, the SD-NGE features further reduce the WER by 1% and 0.5% on two LVCSR tasks. This demonstrates that the SD-SGE features are more helpful in training the DNNs.

SVB-10k			SWB-110h		
System		WER	System		WER
1	SI-DNN (4/1024)	32.17	1	SI-DNN (4/1200)	23.7
2	+ SI-NGE	**30.59**	2	+ SI-NGE	**22.9**
3	+ SD-NGE	**30.18**	3	+ SD-NGE	**22.5**
4	SD-DNN (4/1024)	27.97	4	SD-DNN (4/1200)	20.0
5	+ SI-NGE	27.93	5	+ SI-NGE	20.2
6	+ SD-NGE	**26.90**	6	+ SD-NGE	**19.5**

Table 1: WER using SI and SD neighborhood graph embeddings. The parentheses indicate the number of layers and number of hidden units per layer. Numbers in bold-face indicate significant improvement at $p = 0.05$.

We then compare the SD-NGE vs. SGE features in DNN-HMM systems. In Table 2 we observe that the SGE features (Row 3 and 6) consistently outperform the SD-NGE (Row 2 and 5) features. The improvement over the SI baseline system (Row 1) is even more significant. In SI-DNN systems the SGE feature reduce the word error rate from 2% to 3.6% absolute on SVB-10k and 1.2% to 2.6% absolute on Switchboard. Moreover, the DNN system using SGE features usually has 10% fewer parameters than the system using NGE features. As the number of parameters increases, we can achieve further reduction in WERs. In the SD-DNN systems (Row 4), both SGE and NGE features yield similar performance. Again, the system using SGE features achieves similar results using fewer parameters.

SVB-10k			SWB-110h		
System		WER	System		WER
1	SI-DNN (4/1024)	32.17	1	SI-DNN (4/1200)	23.7
2	+ SD-NGE [6.1M]	**30.18**	2	+ SD-NGE [8.5M]	**22.5**
3	+ SGE [5.5M]	**28.60**	3	+ SGE [7.8M]	**21.1**
4	SD-DNN (4/1024)	27.97	4	SD-DNN (4/1200)	20.0
5	+ SD-NGE [6.5M]	**26.90**	5	+ SD-NGE [8.8M]	**19.5**
6	+ SGE [5.9M]	**26.80**	6	+ SGE [8.1M]	**19.5**

Table 2: WER using neighborhood graph embeddings and similarity graph embeddings. The parentheses indicate the number of layers and number of hidden units per layer. The brackets indicate the number of parameters in the network. Numbers in bold-face indicate significant improvement at $p = 0.05$.

We finally investigate different front-end features for LSTM-CTC system. In the LSTM-CTC system NGE features lead to severe overfitting. SGE features, on the other hand, are a more robust feature representation resulting in consistent reductions in WER. Figure 1 shows the CTC token (phoneme sequence) accuracy on the dev set on SVB-10k and Switchboard using two front-end features. The token error rate is the normalized edit distance between the prediction and target phone sequence labels. It can be observed that the token accuracy achieved with the SGE features is consistently higher than that with the fMLLR features. As shown in Table 3, the WER is reduced by around 1% and 0.4% absolute in both tasks.

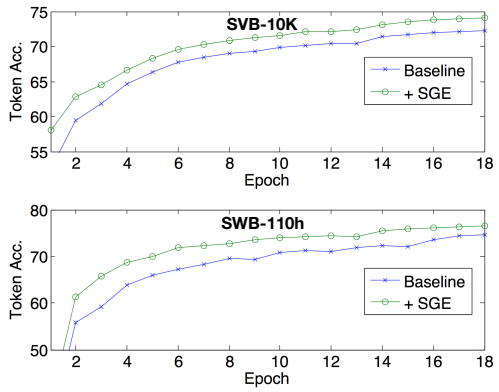

Figure 1: Token (phoneme) accuracy using CTC training on SVB-10k/ Switchboard dev set. Blue line shows baseline accuracy at each epoch; green line shows accuracy using SGE feature at each epoch.

SVB-10k		SWB-110h	
System	WER	System	WER
LSTM-CTC (2/320)	31.27	LSTM-CTC (2/320)	23.4
+ SGE	**30.20**	+ SGE	23.0

Table 3: WER using similarity graph embedding features on LSTM-CTC system. The parentheses indicate the number of LSTM layers and number of memory cell in each forward/backward sub-layer. Numbers in bold-face indicate significant improvement at $p = 0.05$.

5. Conclusion

In this paper we have proposed several types of neural graph embeddings as ASR front-end features. The first is an improved neighborhood graph embedding which integrates speaker-dependent information. The second feature is a similarity graph embedding that encodes global manifold information. We have shown how these features perform as front-end feature in DNN-HMM and LSTM-CTC based ASR systems, and how they can be scaled to LVCSR contexts using the concept of landmarks. In future work we will evaluate this framework on low-resourced ASR tasks where it may be difficult to train speaker-dependent systems; we also plan to work on applications where the training and test conditions are drastically mismatched. Further work will study how to dynamically update landmarks and the autoencoder feature extractor.

6. References

[1] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks." in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2011.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. 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] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2014.

[5] J. Li, A.-R. Mohamed, G. Zweig, and Y. Gong, "LSTM time and frequency recurrence for automatic speech recognition," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015.

[6] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

[7] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep Speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[8] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos *et al.*, "Deep Speech 2: End-to-end speech recognition in English and Mandarin," *arXiv preprint arXiv:1512.02595*, 2015.

[9] Y. Miao, M. Gowayyed, X. Na, T. Ko, F. Metze, and A. Waibel, "An empirical exploration of CTC acoustic models," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of International Conference on Machine Learning (ICML)*, 2006.

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

[12] ——, "Graph-based semi-supervised acoustic modeling in dnn based speech recognition," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[13] ——, "Acoustic modeling with neural graph embeddings," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015.

[14] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[15] L. Heck and H. Huang, "Deep learning of knowledge graph embeddings for semantic parsing of Twitter dialogs," in *Proceedings of GlobalSIP*, 2014.

[16] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014.

[17] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems*, 2002, pp. 705–712.

[18] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

[19] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[20] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[21] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[22] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2013.

[23] Y. Liu, R. Iyer, K. Kirchhoff, and J. Bilmes, "SVitchboard II and FiSVer I: High-quality limited-complexity corpora of conversational English speech," in *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, 2015.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2011.

[25] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015.