ACOUSTIC CLASSIFICATION USING SEMI-SUPERVISED DEEP NEURAL NETWORKS AND STOCHASTIC ENTROPY-REGULARIZATION OVER NEAREST-NEIGHBOR GRAPHS

Sunil Thulasidasan[⋆,†], *Jeffrey Bilmes*[†]

[⋆]Los Alamos National Laboratory
[†]Department of Electrical Engineering, University of Washington

ABSTRACT

We describe a graph-based semi-supervised learning method for acoustic data that uses a Deep Neural Network (DNN) combined with a stochastic graph-based entropic regularizer to favor smooth solutions over a graph induced by the data. We consider graph embeddings constructed from the input features and also from dimensionality-reduced encodings obtained from the bottleneck layer of a separate deep auto-encoder. We use a computationally efficient, stochastic graph-regularization technique that uses mini-batches that are consistent with the graph structure but that also provide enough data diversity for the convergence of stochastic gradient descent methods to good solutions. For this work, we focus on results of frame-level phone classification accuracy on the TIMIT speech corpus but our method is general and scalable to much larger data sets. Results indicate that our method significantly improves classification accuracy compared to the fully-supervised case when the fraction of labeled data is low, and it is competitive with other methods in the fully labeled case.

Index Terms— semi-supervised learning, graph-based learning, deep learning,speech recognition

1. INTRODUCTION

Semi-supervised learning (SSL) methods use both labeled and unlabeled data to improve learning performance [1] and are especially useful in situations where labeled data is scarce. Since unlabeled data can usually be collected in a fully automated, scalable way, SSL methods aim to leverage unlabeled data to improve prediction performance by exploiting the similarity between labeled and unlabeled data. A natural way to capture this relationship is via graphs where the nodes represent both labeled and unlabeled points and the weights of the edges reflect the similarity between the nodes [2]. The main idea behind graph-based SSL methods is that given a similarity metric, the objective function in graph-based SSL methods encourages similar (i.e., nearby) nodes to have the same label by imposing a graph-neighbor regularization; this is effective because it prefers the labels to be consistent with the graph structure (and the underlying manifold represented thereby). Graph-based SSL algorithms have been successfully applied to tasks such as phone and word classification in automatic speech recognition (ASR) [3, 4, 5, 6], part-of-speech tagging [7], statistical machine translation [8], sentiment analysis in social media [9], text categorization [10] and many others.

In this work, [1] we describe algorithmic improvements for efficient and scalable graph regularization that can be applied to any parametric graph-based SSL framework. We use a fully parametric learner – a deep neural network – with an entropy regularizer over the graph induced by the data, a method that was first described in [3] in the context of a multi-layered perceptron (MLP) with one hidden layer. By sampling the data using graph partitioning, but at the same time preserving the statistical properties of the data distribution, and by stochastically regularizing over the graph, we are able to significantly outperform the original results even on an MLP, and make further improvements using a DNN. For the results reported in this paper, we limit our data-set to fixed length speech frames, only reporting frame-level phone classification accuracy on the TIMIT[12] speech corpus without using HMM-based decoding and n-gram language models. Our aim in this work is not to beat state-of-the-art ASR systems (which all use language models[13, 14, 15] and typically will have higher accuracy than the results presented here) but to demonstrate the efficacy of a computationally efficient technique that can potentially be used to improve ASR and other machine learning systems in a semi-supervised setting.

[1] An earlier version of this work appeared in [11]

2. PARAMETRIC OBJECTIVE FOR GRAPH-BASED SSL CLASSIFIERS

Graph-based SSL techniques assume that data are embedded in some low-dimensional manifold in a higher dimensional ambient space, and that nearby nodes will likely have the same labels (the manifold and smoothness assumptions, respectively); the objective function in these methods thus impose a penalty when the output on nearby nodes differ. The general form of the loss function in graph-based SSL has the following form

$$\sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \sum_{(i,j)\in E(\mathcal{G})} \omega_{i,j} g(f(x_i), f(x_j)) \qquad (1)$$

where $f : \mathcal{X} \rightarrow \mathcal{Y}$ is the classifier mapping from input to output space. The first term in Equation 1 the supervised loss function calculated on the labeled points. $l(.)$ can be a squared loss, hinge-loss or some measure of divergence between predictions and ground truth. The second term is the graph regularizer, where $\omega_{i,j}$ captures the similarity between points x_i and x_j, and where $E(\mathcal{G})$ are the edges (pairs of nodes) of a graph $\mathcal{G}$. The function $g(\cdot, \cdot)$ captures the discrepancy between output $f(x_i)$ and $f(x_j)$, incurring a large penalty when similar nodes have different output. Additional regularizers (such as the standard ℓ_1 or ℓ_2) can also be applied to prevent overfitting.

Concretely, let $\{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\ell}$ be the labeled training data and $\{\mathbf{x_i}\}_{\mathbf{i=\ell+1}}^{\ell+\mathbf{u}}$ be the unlabeled training data, where $n = \ell + u$ so that we have n points in total. We denote by $\triangle_M$ the M-dimensional probability simplex (i.e., the set of all distributions over M class labels). Let $\mathbf{p}_\theta(\mathbf{x}_i) \in \triangle_M$ represent the output vector of posterior probabilities dictated by θ, the parameters of the classifier and $\mathbf{t}_i \in \triangle_M$ for $1 \leq i \leq \ell$ denote a probabilistic label vector for the i-th training

sample. We also assume that the samples $\{\mathbf{x}_i\}_i$ are used to produce a weighted undirected graph $\mathcal{G} = (V, E, \mathbf{W})$, where $\omega_{i,j} \in \mathbf{W}$ is the similarity (edge weight) between samples (vertices) $\mathbf{x}_i$ and $\mathbf{x}_j$ (i and j). We use the objective function defined in [16, 3], namely:

$$J(\theta) = \sum_{i=1}^{l} \mathbf{D}(\mathbf{t}_i \| \mathbf{p}_\theta(\mathbf{x}_i)) + \gamma \sum_{i,j=1}^{n} \omega_{i,j} \mathbf{D}(\mathbf{p}_\theta(\mathbf{x}_i) \| \mathbf{p}_\theta(\mathbf{x}_j)) + \kappa \sum_{i=1}^{n} \mathbf{D}(\mathbf{p}_\theta(\mathbf{x}_i) \| \mathbf{u}) + \lambda \|\theta\|_2 , \quad (2)$$

where $\mathbf{u} \in \triangle_M$ is the uniform distribution and $J(\theta)$ is the loss calculated over all samples. The weights $\omega_{i,j}$ themselves are sparse, so that $\omega_{i,j} = 0$ when $(i, j) \notin E(\mathcal{G})$. We use KL-divergence (denoted by $\mathbf{D}(\cdot\|\cdot)$) as our loss function since our output is a probability distribution over classes. The first term in the above equation is the supervised loss over the training samples, and the second term is the penalty imposed by the graph regularizer over neighboring pairs of nodes that favors smooth solutions over the graph. The third term is an entropy regularizer and favors higher entropy distributions since MLPs and DNNs are often very confident in their predictions which can lead to degenerate solutions; favoring higher entropy solutions counters this and is especially useful near decision boundaries. An alternative to regularizing against the uniform distribution is to regularize against a prior $\tilde{\mathbf{p}}(\mathbf{x_i})$ as done in [4], where $\tilde{\mathbf{p}}(\mathbf{x_i})$ are the outputs from a first-pass classifier trained in a supervised manner. We can easily incorporate this into our framework, though the work described in this report regularizes against only the uniform distribution. The final term in Equation 2 is the standard ℓ_2 regularizer to discourage overfitting.

3. RELATED WORK

There have been several graph-based learning algorithms that make use of some version of the objective function described in the previous section [4, 16, 17, 18]. Label propagation, described in [17] forces f to agree with labeled instances by minimizing squared loss between predictions of nearby points. Measure propagation, described in [16] uses essentially the same objective function as in Equation 2 but in a non-parametric setting. Prior-regularized measure propagation [4] substitutes the uniform distribution in Equation 2 with a prior $\tilde{\mathbf{p}}_\mathbf{i}$ that comes from a supervised classifier prior to the SSL process, and has shown to perform well on speech data. One of the early works to use a graph regularizer in a deep learning context is described in [19], where squared loss is used instead of KL-divergence. The algorithms described in this paper are most related to [3] but significantly improve upon the graph construction and parallel programming methodologies and also apply it to deep models. Our methods will generally work on any objective function with a graph regularizer.

4. GRAPH REGULARIZATION VIA GRAPH PARTITIONING

Like other graph-based SSL methods we induce a graph on the data by constructing a k-nearest neighbor (k-NN) graph where the edge weights are the Euclidean distance between the feature vectors. Since we are dealing with a non-convex objective function, and a moderately large data set ($\approx$ 1 million training samples), we use stochastic gradient descent (SGD) to optimize our objective function. We also use mini-batches to improve the gradient quality, and further, use larger mini-batches (size set to 1024) for better computational efficiency. Traditional SGD methods require randomly shuffling of the data for good convergence before constructing the mini-batches; this, however, poses a problem for our objective function. To see this, consider the terms involving graph regularization from our objective function, calculated over each point:

$$G_i = \gamma \sum_{j=1}^{n} \omega_{i,j} D(\mathbf{p}_\theta(\mathbf{x}_i) \| \mathbf{p}_\theta(\mathbf{x}_j))$$

For the graph regularization term to have any effect at all, the w_{ij}'s corresponding to the points in the mini-batch have to be non-zero. For a randomly shuffled data-set, given that the k-NN graph is very sparse (since each of the $\approx$ 1 million points only has a little more than $k = 10$ neighbors), the chunk of the affinity matrix corresponding to the mini-batch will be extremely sparse, implying that graph regularization will fail to take place on most computations. A naive way to address this would be to loop over all the neighbors for each point in the mini-batch, but this would prevent us from doing efficient matrix-matrix multiplications and would severely degrade performance negating any benefits of using fast processors like GPUs.

Thus, for the graph regularizer to be effective in a computationally efficient way, our mini-batches need to reflect the structure of the graph. To do this, we partition our affinity graph into k balanced parts by minimizing edge-cut (i.e, given k, we want to minimize the number of edges between partitions). The resulting re-permuted affinity matrix has a dense block-diagonal structure; during mini-batch gradient descent, each mini-batch tends to correspond to points inside one of the partition blocks. The corresponding relatively dense submatrices of the affinity matrix are used for the graph regularization computation over the mini-batches.

4.1. SGD on Graph-Based Mini-Batches

Theoretically, SGD, gives us an unbiased estimate of the true gradient, but only if the data is sampled from the true distribution. If our entire data set approximates this true distribution reasonably well, then a randomly sampled mini-batch will also be faithful to this distribution. However, for a graph partitioned mini-batch this argument no longer holds since the data points that comprise a mini-batch are not randomly sampled, but on the contrary, reflect relatively homogeneous regions on some low dimensional manifold (since we are partitioning by minimizing edge-cut on a k-NN graph). Thus our gradient estimate is no longer unbiased, leading to poor convergence of SGD. On the other hand we have also seen that randomly shuffled batches will cause the graph regularizer to become ineffective due to poor within-batch neighbor connectivity, unless one accepts extremely long computational times (and communication costs in a parallel implementation).

Constructing a mini-batch that gives good SGD convergence, and good neighbor connectivity represents a trade-off between two somewhat mutually opposing properties: diversity (for SGD convergence, also found to be the case in [20, 21]) and good neighbor-connectivity (for efficient graph regularization), which usually implies homogeneity. The full batch (i.e., the entire data set) however, has both these properties; perfect neighbor connectivity (since it contains all the points) as well as diversity[2] that mimics the diversity within the complete training data (assuming a large enough, well sampled training set). Indeed, if we are allowed to increase the size of the mini-batches as we please, we could presumably capture a more diverse set of points as well as a significant fraction of their neighbors, but computational and memory constraints prevent us from doing so. Note that

[2]We use Shannon entropy, calculated on the labels in a mini-batch, as a measure of diversity, but we anticipate better diversity measures exist.

the global structure of the affinity graph, owing to its sparsity, consists of a large number of small tightly connected clusters, with relatively few edges between the clusters. Thus a mini-batch that somehow captures this structure, but on a smaller scale, will be expected to have reasonably good connectivity as well as high entropy. This suggests a possible heuristic for the construction of improved mini-batches:

1. Given N data points, a batch size B (that represents our memory constraint) and M classes, partition the entire graph into $\frac{NM}{B}$ mini-blocks, where each mini-block is approximately balanced at size B/M.
2. Construct N/B "meta" batches of size B from the mini-blocks as follows:
   (a) For each batch b_i, randomly choose M mini-blocks from the set of $\frac{NM}{B}$ mini-partitions that were created in Step 1.
   (b) Group these M mini-blocks into one larger meta-batch. Since each mini-block is approximately size B/M, our meta-batch will be approximately of size B, satisfying our memory constraint.

At the end of this process we have meta-batches which are of the same size B as the earlier graph-based batches, but which are qualitatively different. Each meta-batch is now composed of many small homogeneous mini-blocks which, due to random sampling, are likely to be of a different class. We omit the proof here due to space constraints, but intuitively we expect that the resulting entropy from grouping together M such randomly chosen mini-blocks (of approximately equal size) to approach the entropy of the training set.

To see the effect of this process on the within-batch neighbor connectivity of the meta-batch, let $\mathcal{N}_i$ represent the set of neighbors of node i and $\mathcal{C}_i \subseteq \mathcal{N}_i$ be the set of neighbors of a node i that are within the same batch. Let $\mathcal{M}_j$ be the set that represents mini-batch j. We define the within-batch connectivity of $\mathcal{M}_j$ as

$$c_j = \frac{\sum |\mathcal{C}_i|}{\sum |\mathcal{N}_i|}, \forall i \in \mathcal{M}_j, j = 1, 2, 3 \dots k \tag{3}$$

Let C_{mini} and C_{meta} denote the random variables that represent the within-batch connectivity of a mini-block and meta-batch respectively. One can show that grouping K mini-blocks to form a meta-batch does not adversely impact the connectivity score, i.e., $E[C_{meta}] \geq E[C_{mini}]$. Further, using the Central Limit Theorem, we can show that the variance of c_{meta} is given by $\sigma^2_{c_{meta}} = \frac{1}{K}\sigma^2_{c_{mini}}$.

4.2. Stochastic Regularization over Graphs

Even though a meta-batch constructed using the procedure described in the previous section has much better neighbor-connectivity than a randomly shuffled batch, for a given node, a significant number of neighbors still lie outside the meta-batch.[3] As we argued earlier, regularizing against all neighbors is computationally inefficient. To preserve efficiency while still regularizing against out-of-batch neighbors, at each step, we *randomly* pick one additional meta-batch and regularize against this neighbor as follows: consider the graph induced by the meta-batches, $\mathcal{G}_M = (V^M, E^M)$, $V^M = \{M_1, M_2 ... M_{\lfloor N/B \rfloor}\}$ where each M_i is a meta-batch, and edge $e^M_{i,j} \in E^M$ exists between M_i and M_j if there exist some edge $e_{s,t}$ between nodes v_s and v_t in the affinity graph $\mathcal{G}$, such that $v_s \in M_i$ and $v_t \in M_j$. That is, meta-batches are connected if their member nodes are connected in the original affinity graph. Let $\mathcal{C}_{i,j}$ denote the set consisting of all such unique pairs v_s, v_t. Then we can define an edge-weight on each of the edges in E^M as $|\mathcal{C}_{i,j}|$. For a given meta-batch M_j, during each epoch, the probability of picking a neighboring meta-batch M_j is given by

$$p_{i,j} = \frac{|\mathcal{C}_{i,j}|}{\sum_j |\mathcal{C}_{i,j}|}. \tag{4}$$

Thus, a neighboring batch M_j of batch M_i is more likely to be picked during an epoch if there are a relatively large number of edges between the member nodes that comprise M_i and M_j. Over a large number of epochs, graph regularization is likely to take place against all neighboring batches; this enables labels to propagate via a stochastic diffusion process within the connected components of the affinity graph.

5. EXPERIMENTS

For all our experiments in this work we use the TIMIT speech corpus [12] and just report the frame-level phone classification accuracy. Features consist of 39-d vectors consisting of MFCC coefficients, and first and second derivatives. All data is normalized for zero mean and unit variance. We apply a sliding window of radius 4, resulting in a 351 dimensional feature vector. The output is a distribution over 49 classes, which is collapsed to 39 classes during scoring. We use the 362 speaker set for training and experiment with label ratios of 2%, 5%, 10%, 30%, 50% and 100% by randomly dropping labels from our training set. Hyper-parameters were tuned using parallel grid search on a validation set. We implemented all our models using the Theano toolkit [22]. For the results reported here we used the AdaGrad [23] variant of gradient descent and use a hold-out set for early stopping. For the k-NN graph construction, we set $k = 10$ for all the experiments and use the Scikit machine learning library [24] that constructs the graphs using a fast ball-tree search. After symmetrization, affinities are computed by applying a radial basis function (RBF) kernel, such that each entry w_{ij} in the affinity matrix W, $w_{ij} = e^{-\frac{||x_i - x_j||}{2\sigma^2}}$. σ controls the width of the kernel and determines how quickly the influence of a neighbor node decays with distance. As in [3], we tune σ over the set $\{d_i/3\}$ where $i \in \{1, 2, 3, 4, 5\}$ and d_i is the average distance between a node and it's i-th nearest neighbor. For graph partitioning, we use the METIS graph partitioning library [25] that uses a recursive multi-way partitioning to give approximately balanced blocks.

We initially tested the benefit of the meta-batches and stochastic graph regularization on a shallow neural network – a multi-layer perceptron (MLP) having one hidden layer of 2000 units and a softmax output layer. These results are shown in Figure 1. A graph-regularized MLP that uses mini-batches based on purely graph partitions, and without additional out-of-batch neighbor regularization performs the worst (red curve in Figure 1); this is not surprising considering the biased gradients when using relatively homogeneous graph-based mini-batches. Using meta-batches, both with and without stochastic out-of-batch regularization (the blue and green curves respectively), noticeably improves performance, the former beating the base MLP (a supervised learner) at all scenarios except the fully-labeled case. Next, we conducted experiments on a DNN with four hidden layers, each 2000 units wide, using Rectified Linear Units [26] as the non-linear activation function, and a softmax output layer. We also exploited the feature-learning ability of DNNs – which have been shown to be good at highly non-linear dimensionality reductions [27], and improve graph clustering [28] – to produce graph embeddings

[3] This fraction will depend on the mini-block size; for most of the experiments in this paper about 30% of the neighbors lie within a meta-batch.

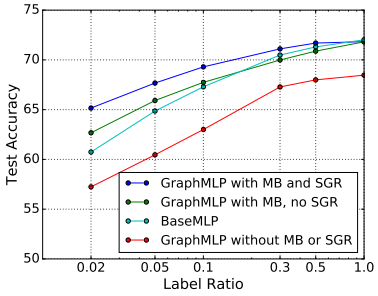
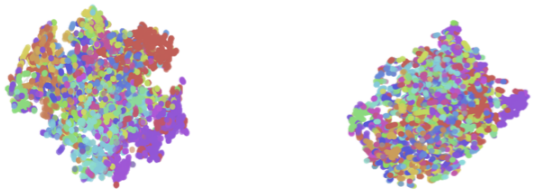


Fig. 1: Effect of meta-batches and stochastic graph regularization (SGR) on the performance of a graph-regularized MLP. Also shown are results of a base-line MLP, a fully-supervised learner that only uses labeled data.



(a) DNN-based embedding

(b) Input feature-based embedding

Fig. 2: 3-d embedding of TIMIT data using bottleneck representations learnt via supervised training of a DNN (left) compared to embedding produced from the input (right). The colors indicate class membership

which are then used to train the learner. To test the quality of embeddings produced from the bottleneck layer, we use a deep autoencoder with a 40-hidden-unit bottleneck layer and then project the encodings into 3-d space using an implementation of the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique[29]; we also plot the projection from the MFCC feature space. Figure 2 shows the embeddings from t-SNE. Visually, the embeddings resulting from the bottleneck layer of the DNN appear to be more consistent with the manifold, since points from the same class appear clustered together in the DNN embeddings (Figure 2a) compared to embeddings using input features (Figure 2b). Thus the DNN is learning a better similarity metric, and we would expect that the graph constructed using this metric would be more consistent with the actual manifold.

For the semi-supervised DNN training, we used dropout reporting the results for the case when dropout probability is 0.2, for which we saw the best performance. Dropout is essentially a stochastic regularization technique and admittedly changes our objective function, but it is interesting to note that even in this setting, the graph regularization still significantly improves classification performance over the baseline DNN at the lower label ratios as shown in Figure 3. We also compare against the results reported in [3], which was the main reference point in this work. In addition to the optimizations described in this paper, compared to [3] we can also train for significantly longer number of epochs owing to both greater computational capacity and better adaptive gradient methods, relative to 2009, which allows us to improve over the results in that paper. We also provide a comparison against a somewhat similar (although non-parametric) graph-based

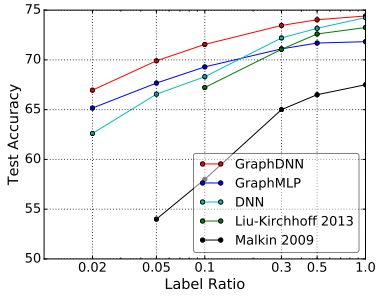


Fig. 3: Results for Graph Regularized DNN vs base-line DNN. The DNN used four hidden layers, each 2000 units, with dropout. Also shown are results for similar experiments in the literature that used the same TIMIT training and test data.

SSL framework reported in [4]. Compared to the latter work, for the shallow classifier (GraphMLP), we generally get better phone accuracy rates although at higher label ratios [4] is better. This is probably due to regularizing against a prior distribution output from a first-pass classifier, which provides better priors at higher label ratios. When moving to DNNs, however, we are able to improve performance over [4]; the graph-enabled DNN gives the best performance at all labeled ratios both due to the depth of the classifier as due to the added benefit of using a better embedding obtained from the deep auto-encoder. Any embedding can be used with the semi-supervised DNN in general, and one of the future directions of research is to compare the effect of different graph-based representations.

6. CONCLUDING REMARKS

We presented a method for graph-based semi-supervised learning with DNNs for classifying acoustic data. The training method samples data that enables regularization over the graph (which is a proxy for the manifold) but also preserves diversity in the training samples for convergence of SGD to good solutions. The stochastic graph regularization technique allows efficient out-of-batch regularization and though we only report frame-level classification accuracy on the TIMIT speech data set, this is a general, scalable technique that can be applied to much larger and different kinds of data sets, other types of parametric classifiers and can also be extended to online and distributed learning settings.

7. ACKNOWLEDGEMENTS

Thulasidasan was supported by the LDRD Program of the Los Alamos National Laboratory under U.S. Department of Energy Contract No. DE-AC52-06NA25396. Bilmes was supported by the National Science Foundation under Grant No. IIS-1162606, and by a Google, a Microsoft, a Facebook, and an Intel research award, and also in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

8. REFERENCES

[1] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al., *Semi-supervised learning*, MIT press Cambridge, 2006.

[2] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld, *Semi-supervised learning with graphs*, Carnegie Mellon University,

Language Technologies Institute, School of Computer Science, 2005.

[3] Jonathan Malkin, Amarnag Subramanya, and Jeff A Bilmes, "On the semi-supervised learning of multi-layered perceptrons.," in *INTERSPEECH*, 2009, pp. 660–663.

[4] Yuzong Liu and Katrin Kirchhoff, "Graph-based semi-supervised learning for phone and segment classification.," in *INTERSPEECH*, 2013, pp. 1840–1843.

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

[6] Yuzong Liu and Katrin Kirchhoff, "Acoustic modeling with neural graph embeddings," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 581–588.

[7] Amarnag Subramanya, Slav Petrov, and Fernando Pereira, "Efficient graph-based semi-supervised learning of structured tagging models," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 167–176.

[8] Andrei Alexandrescu and Katrin Kirchhoff, "Graph-based learning for statistical machine translation," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 119–127.

[9] Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald, "Sentiment summarization: evaluating and learning user preferences," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 514–522.

[10] Amarnag Subramanya and Jeff Bilmes, "Soft-supervised learning for text classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008, pp. 1090–1099.

[11] Sunil Thulasidasan and Jeff Bilmes, "Semi-supervised phone classification using deep neural networks and stochastic graph-based entropic regularization," in *2016 Workshop on Machine Learning in Speech and Language Processing*, San Francisco, CA, September 2016.

[12] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, 1993.

[13] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4277–4280.

[14] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009, vol. 1, p. 39.

[15] Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," *arXiv preprint arXiv:1304.1018*, 2013.

[16] Amarnag Subramanya and Jeff A Bilmes, "Entropic graph regularization in non-parametric semi-supervised classification," in *Advances in Neural Information Processing Systems*, 2009, pp. 1803–1811.

[17] Xiaojin Zhu and Zoubin Ghahramani, "Learning from labeled and unlabeled data with label propagation," Tech. Rep., Citeseer, 2002.

[18] Partha Pratim Talukdar and Koby Crammer, "New regularized algorithms for transductive learning," in *Machine Learning and Knowledge Discovery in Databases*, pp. 442–457. Springer, 2009.

[19] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

[20] K. Wei, R. Iyer, S. Wang, W. Bai, J. Bilmes, "How to intelligently distribute training data to multiple compute nodes: Distributed machine learning via submodular partitioning," in *Neural Information Processing Society (NIPS) Workshop*, Montreal, Canada, December 2015, LearningSys Workshop, http://learningsys.org.

[21] Kai Wei, Rishabh Iyer, Shengjie Wang, Wenruo Bai, and Jeff Bilmes, "Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications," in *Neural Information Processing Society (NIPS)*, Montreal, Canada, December 2015.

[22] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a cpu and gpu math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Austin, TX, 2010, vol. 4, p. 3.

[23] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] George Karypis and Vipin Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[26] Matthew D Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al., "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[27] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[28] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu, "Learning deep representations for graph clustering.," in *AAAI*, 2014, pp. 1293–1299.

[29] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, pp. 85, 2008.