Validating Efficacy and Robustness of Neural Structured Learning

Dheeraj Kumar Kondaparthi, Nishank Lakkakula, Sudheer Kumar Reddy Beeram, Abhinav Reddy Nadimpally
dkonda2@uic.edu, nlakka2@uic.edu, sbeera2@uic.edu, anadim4@uic.edu
Computer Science Department
University of Illinois at Chicago

Abstract — Neural Networks have proved to be powerful in many supervised learning tasks and on the other hand label propagation powerful technique used in semi-supervised learning on graphs. Neural Graph Learning proposes a training framework with graph regularized objective that combines power of neural networks and label propagation. This objective allows neural networks to train network in supervised setting and to bias the network to learn similar hidden representations for neighboring nodes of graph. Adversarial Learning on the other hand aims to increase neural networks robustness by learning if neighbor samples are induced by adversarial perturbation. Neural Structured Learning aims to generalize to Neural Graph Learning and Adversarial Learning by learning from structed signals in data, synthesizing the structure and learning from adversarial samples. This new model architecture brings three main advantages: high accuracy, robustness and training with less labeled. In this project we intend to show these advantages on variety of text and image classification tasks on different datasets.

1. INTRODUCTION

The need for Semi-supervised learning arises in many problems of computer vision, natural language process and social networks in proportion of unlabeled data and very abundant compared to labelled data. Bootstrapping techniques can be used generate pseudo labels for unlabeled data from system trained on labeled data, but these techniques suffer from label error feedbacks [1]. Autoencoder and transductive SVMs [4] can also be used, but they are computationally expensive to be used on large datasets and often inaccurate. Methods based on generative models work better for images and videos, but it is not immediately clear as in how well they can work for multi-model data. So, in contrast to above methods, graph-based techniques can be used for label propagation [2, 3] to provide versatile, scalable and effective solution to wide range of problems. These methods construct a smooth graph over the unlabeled and labeled data. Graphs are also often a natural way to describe the relationships between nodes, such as similarities between embeddings, phrases or images, or connections between entities on the web or relations in a social network. Edges in the graph connect semantically similar nodes or datapoints, and if present, edge weights reflect how strong such similarities are. By providing a set of labeled nodes, such techniques iteratively refine the node labels by aggregating information from neighbors and propagate these labels to the nodes' neighbors. In practice, these methods often converge quickly and can be scaled to large datasets with a large label space [5]. Neural Structure Learning is built upon this principle for label propagation.

The second motivation to Neural Structured Learning is to harness state of the art semi supervised learning techniques, to jointly train neural networks using limited labeled data and unlabeled data to improve its performance.

The third motivation to Neural Structured learning is to jointly train the network by generating new labels as adversarial samples thereby to check if the model can be robust to adversarial attacks.

Contributions: In this paper, we investigate and validate the advantages of Neural Structured Learning on various image and text classification tasks. The new training objective has a regularization term for generic neural network architectures to enforce similarity between nodes in graph, which is similar to training objective of label propagation. In particular, through our experiments, we found that,

- Neural Structure Learning using graph augmented networks can be applied to wide variety of neural networks such a feed forward neural networks, convolution neural networks, recurrent neural networks.
- The framework can handle either natural graphs or synthesized graphs.
- The framework is robust to adversarial attacks thereby generalizing to adversarial learning

We experimentally show that Neural Structured Learning framework performs favorably on variety of prediction tasks and datasets involving text features and/or graph inputs and on different neural network architectures.

2. BACKGROUND AND RELATED WORK

In this section we lay foundations to the Neural Structure Learning training objective in section.3

2.1 Neural Network Learning

Neural networks are a class of non-linear mapping from inputs to outputs and are composed of multiple layers that can potentially learn useful representations for predicting the outputs. Various models such as feed-forward neural networks, recurrent neural networks and convolutional networks are viewed under the same umbrella. Given a set of N training input-output pairs $\{x_n, y_n\}_{n=1}^{N}$, such neural networks are often trained by performing maximum likelihood learning, that is, tuning their parameters so that the networks' outputs are close to the ground truth under some criterion,

$$C_{NN}(\theta) = \sum_n c(g_\theta(x_n), y_n) \quad (1)$$

where $g_\theta(.)$ denotes the overall mapping, parameterized by θ and c(.) denotes a loss function such as *l*-2 for regression and cross entropy for classification. The cost function c and mapping g are typically differentiable w.r.t θ which facilitates optimization via gradient descent. This can also be scaled to a large number of training instances by employing stochastic training using mini batches of data. However, it is not clear how unlabeled data, if available, can be treated using this objective, or if extra information about the training set, such as relational structures can be used.

2.2 Graph Based Semi Supervised Learning

In this section, a concise introduction to graph based semi-supervised learning using label propagation and its training objective. Suppose a graph *G = (V, E, W)* where V is a set of nodes, E the set of edges and W the edge weight matrix is given. Let V_l, V_u be labeled and unlabeled nodes in the graph. The goal is to predict a soft assignment of labels for each node in the graph, $\hat{Y}$, given the training label distribution for the see nodes, Y. Mathematically, label propagation performs minimization of following convex objective function, for L labels,

$$C_{LP}(\hat{y}) = \mu_1 \sum_{v\in V_l} \|\hat{y_v} - y_v\|_2^2 + \mu_2 \sum_{v\in V, u\in N(v)} w_{u,v} \|\hat{y_v} - \hat{y}_u\|_2^2 + \mu_3 \sum_{v\in V} \|\hat{y_v} - U\|_2^2 \qquad (2)$$

subject to $\sum_{l=1}^{L} \widehat{Y_{vl}} = 1$, where *N(v)* is neighbor node set of the node v and U is the prior distribution over all labels, $w_{u,v}$ is the edge weight between nodes u and v and μ_1, μ_2 and μ_3 are hyper parameters that balance the contribution of individual terms in the objective. The terms in the objective function above encourage that: (a) the label distribution of seed nodes should be close to the ground truth, (b) the label distribution of neighboring nodes should be similar, and, (c) if relevant, the label distribution should stay close to our prior belief. This objective function can be solved efficiently using iterative methods such as the Jacobi procedure. That is, in each step, each node aggregates the label distributions from its neighbors and adjusts its own distribution, which is then repeated until convergence.

There are many variants of label propagation that could be viewed as optimizing modified versions of eq. (2) and balancing the smoothness constraint and the fitting constraint [6]. For example, manifold regularization [7] replaces the label distribution $\hat{Y}$ by a Reproducing Kernel Hilbert Space mapping from input features. Similarly, [8] also employs such mapping but uses a feed-forward neural network instead. Both methods can be classified as inductive learning algorithms; whereas the original label propagation algorithm is transductive [9].

Neural Structured Learning generalizes above mentioned frameworks for graph-augmented training of neural networks and extends it to new settings where there is only graph input and no features are available. In our experiments, we aim to learn better predictive models directly from the graphs and we compare our method to two stage (embedding + classifier) techniques.

The work with Neural Structure Learning is different and orthogonal to recent works with using neural networks on graphs. [10,11] employ spectral graph convolution to create neural network like classifier. However, these approaches require many approximations to arrive at a practical implementation. However, in Neural Structured Learning, a training objective is advocated in such a way that uses graphs to augment neural network learning and works with many forms of graphs and with any type of Neural Network.

2.3 Neural Graph Machines

Neural Graph Machines [12] is proposed by Ravi et al., in 2018 by devising a discriminative training objective for neural networks that is inspired by the label propagation objective and uses both labeled and unlabeled data and can be trained by stochastic gradient descent. For example: if a cat image and a dog image are strongly connected in a graph, and if the cat node is labeled as animal, the predicted probability of the dog node being an animal is also high. This can be done by encouraging neighboring data points to have similar hidden representations Ravi et al., proposed following objective function which is a weighted sum of neural network cost and label propagation cost as follows,

$$C_{NGM}(\theta) = \alpha_1 \sum_{(u,v)\in\varepsilon_{LL}} w_{uv} d\big(h_\theta(x_u), h_\theta(x_v)\big) +$$

$$\alpha_2 \sum_{(u,v)\in\varepsilon_{LU}} w_{uv} d\big(h_\theta(x_u), h_\theta(x_v)\big) +$$

$$\alpha_3 \sum_{(u,v)\in\varepsilon_{UU}} w_{uv} d\big(h_\theta(x_u), h_\theta(x_v)\big) +$$

$$\sum_{n=1}^{V_l} c(g_\theta(x_n), y_n), \qquad (3)$$

where $\varepsilon_{LL}, \varepsilon_{LU}$ and ε_{UU} are sets of labeled-labeled, labeled-unlabeled and unlabeled-unlabeled edges correspondingly, *h(.)* represents hidden representations of the inputs produced by the neural network and *d(.)* is a distance metric, and $\{\alpha_1, \alpha_2, \alpha_3\}$ are hyperparameters. This framework is general, so one can plug in either the hidden representations at any intermediate layer or the estimated soft label vector. In practice, *l-1* or *l-2* distance metric is chosen for *d(.)* and *h(x)* to be the last hidden of the neural network, or a cross entropy cost for the predicted label vector. Neural Graph Machines concept forms the core of neural structured learning.

2.4 Adversarial Learning

Goodfellow et al., in 2015 proved that neural networks can be tricked by applying small but intentionally worst-case perturbations to examples from the dataset [13]. Goodfellow et al., in 2015 have further showed that perturbations can be generated by using following equation,

$$n = \epsilon sign(\nabla_x J(\theta, x, y)) \qquad (4)$$

where θ be the parameters of the model, *x* is the input to the model, *y* is the targets associated with *x* (for machine learning tasks that have targets) and $J(\theta, x, y)$ is the cost used to train the neural network. Further to this training with an adversarial objective function based on fast gradient sign method was found to be effective regularizer as per below equation

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + sign(\nabla_x J(\theta, x, y)) \quad (5)$$

Neural Structured Learning also generalizes adversarial training if augmented neighbors generated by graph augmentation are adversarial perturbation which will be further explained in next section.

3. NEURAL STRUCTURED LEARNING

Neural Structured Learning (NSL) is a new learning paradigm to train neural networks by leveraging structured signals in addition to featured inputs. Structure can be explicit as represented by a graph or implicit as induced by adversarial perturbation.

Structured signals are commonly used to represent relations or similarity among samples that may be labeled or unlabeled. Therefore, leveraging these structured signals during neural network training harnesses both labeled and unlabeled data, which can improve model accuracy, particularly when amount of labeled data is relatively small. Additionally, models that are trained by adding adversarial perturbation have been shown to be robust against malicious attacks, which are designed to mislead a model's prediction or classification. NSL therefore, generalizes to Neural Graph Learning and Adversarial Learning.

As introduced by Bui et al., in 2018, structured signals are used to regularize the training of neural network, forcing the model to learn accurate predictions by minimizing the supervised loss, while at the same time maintaining the input structure similarity by minimizing the neighbor loss as per Figure1 below. This technique can be applied to any arbitrary neural architectures such as feed forward neural network, convolution neural network, recurrent neural network.

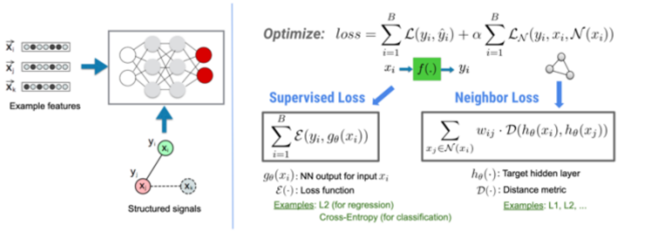

$\vec{x}_i$
$\vec{x}_j$
$\vec{x}_k$
Example features
y_i
x_i
y_j
x_j
x_k
Structured signals
Optimize: $loss = \sum_{i=1}^{B} \mathcal{L}(y_i, \hat{y}_i) + \alpha \sum_{i=1}^{B} \mathcal{L}_{\mathcal{N}}(y_i, x_i, \mathcal{N}(x_i))$
$x_i \rightarrow f(.) \rightarrow y_i$
Supervised Loss
Neighbor Loss
$\sum_{i=1}^{B} \mathcal{E}(y_i, g_\theta(x_i))$
$\sum_{x_j \in \mathcal{N}(x_i)} w_{ij} \cdot \mathcal{D}(h_\theta(x_i), h_\theta(x_j))$
$g_\theta(x_i)$: NN output for input x_i
$\mathcal{E}(\cdot)$: Loss function
Examples: L2 (for regression)
Cross-Entropy (for classification)
$h_\theta(\cdot)$: Target hidden layer
$\mathcal{D}(\cdot)$: Distance metric
Examples: L1, L2, ...

Figure1: Input features and structured signals as inputs to network and loss function as combination of supervised and neighbor loss.

This generalized neighbor loss equation is flexible and can have other forms besides the loss function illustrated in Figure1. Neighbor loss can also be selected as $\sum_{x_j \epsilon N(x_i)} \varepsilon(y_i, g_\theta(x_j))$ which calculates distance between ground truth(y_i) and prediction from the neighbor($g_\theta(x_j)$). This is commonly used in adversarial learning as per section 2.4.

The overall workflow for NSL is shown in Figure2. Black arrows represent the conventional training workflow and red arrows represent the new workflow as introduced by NSL to leverage structured signals. First, the training samples are augmented to include structured signals. When structured signals are not explicitly provided, they can be either constructed or induced (the latter applies to adversarial learning). Next, the augmented training samples (including both original samples and their corresponding neighbors) are fed to the neural network for calculating their embeddings. The distance between a sample's embedding and its neighbor's embedding is calculated and used as the neighbor loss, which is treated as a regularization term and added to the final loss. For explicit neighbor-based regularization, we typically compute the neighbor loss as the distance between the sample's embedding and the neighbor's embedding. However, any layer of the neural network may be used to compute the neighbor loss. On the other hand, for induced neighbor-based regularization (adversarial), we compute the neighbor loss as the distance between the output prediction of the induced adversarial neighbor and the ground truth label.

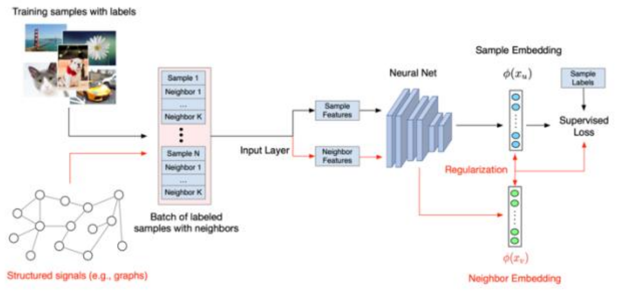


Figure2: NSL Workflow

NSL brings following advantages, 1. Higher accuracy: Joint training of structured signals and input features has shown to outperform many existing methods (that rely on training with features only) on wide range of tasks, such as document classification and semantic intent classification.

2. Robustness: Models trained with adversarial examples have shown to be robust against adversarial perturbations.

3. Works even with less labeled data: NSL enables neural networks to harness both labeled and unlabeled data, which extends the learning paradigm to semi-supervised learning.

4. EXPERIMENTS AND RESULTS

In this section, we test the effectiveness of Graph based Learning in situations where structured signals may be explicitly given or constructed and also test the robustness of Neural Structured Framework on adversarial learning.

Case 1: When the graph is explicitly given with the dataset

Here we test the use of graph regularization to classify documents that form a natural graph.We use Cora dataset which is a citation graph where nodes represent machine learning papers and edges represent citations between pairs of papers. The task involved is a multi-class document classification where the goal is to categorize each paper into one of 7 classes.

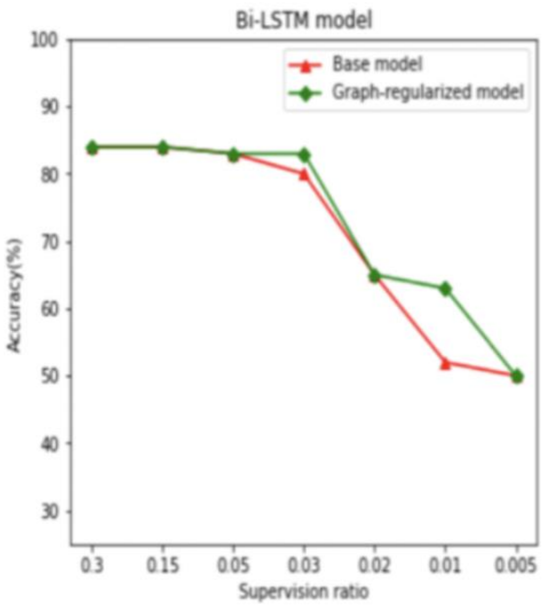
The dataset consists of 2708 scientific publications and a directed citation network. However, we consider the undirected version of this graph. So, if paper A cites paper B, we also consider paper B to have cited A. Although this is not necessarily true, but for this, we consider citations as a proxy for similarity, which is usually a commutative property.

First, we tested the model using a base model which is a simple Feed Forward Neural Net with 2 hidden layers with dropout between them and achieved an accuracy of 78%. We then performed graph regularization on this model by tweaking the loss term and achieved an accuracy of 80%. Graph regularization increased the accuracy of the base model by approximately 2%.

Case 2: When the graph is not explicitly given but constructed.

Here we test the effectiveness of the proposed graph regularization for sentiment classification using synthesized graphs. We used the IMDB and Yelp Polarity datasets.Both IMDB and Yelp Polarity datasets contains a total of 50,000 reviews which are equally split into testing and training samples and are balanced with equal number of positive and negative reviews. The Original Yelp Polarity dataset consists of 5,60,000 reviews but we have only used a part of it since it is computationally very expensive to create and traverse a graph with huge number of nodes. The task in both the datasets is to classify a review into positive and negative sentiment.

Graph construction involves creating embeddings for text samples and then using a similarity function to compare the embeddings. We used pretrained Swivel embeddings to create embeddings foreach sample in the input. These sample embeddings are used to build a similarity graph. We build the edges between nodes by comparing the embeddings using cosine similarity with a similarity threshold of 0.99 that is used to discard dissimilar edges from the final graph.We used bi-directional LSTM as our base model and we take 60% of the initial training samples (60% of 25000) as labeled data for training and the remaining as validation data. Since the initial train/test split was 50/50 (25000 samples each), the effective train/validation/test split we now have is 30/20/50. We achieved an accuracy of 85% with IMDB dataset and 88.8% with Yelp Polarity reviews. We then performed graph regularization on this model and achieved an accuracy of 84.24% and 88.9% respectively.



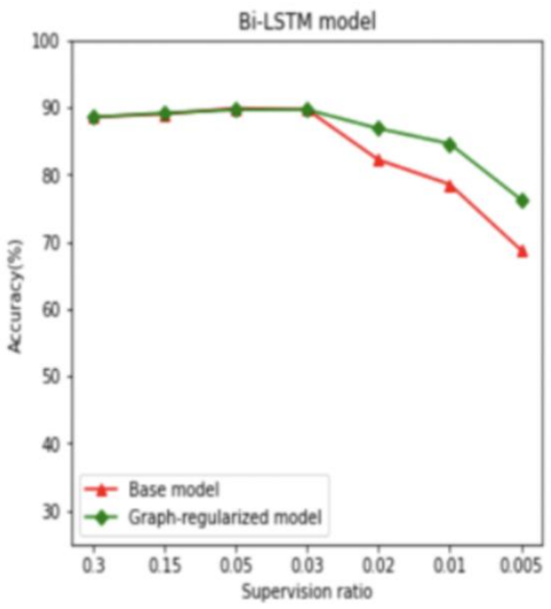
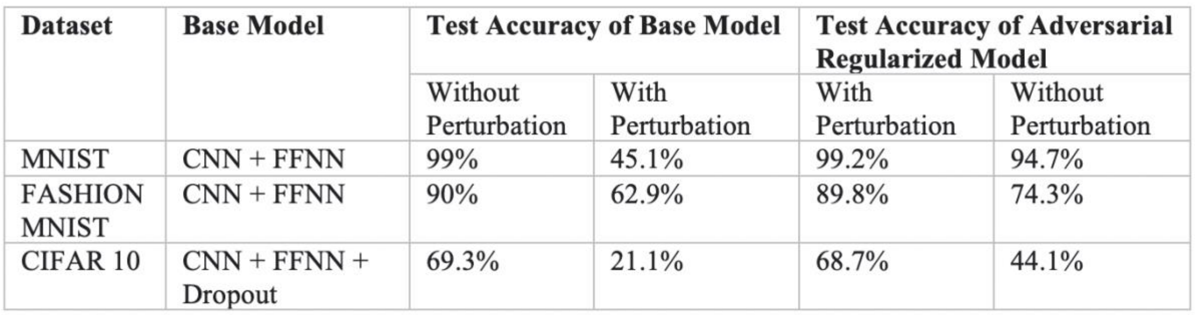


Figure 3: Graph depicting the effect of supervision ratio on model accuracy for both IMDB (left) and Yelp Polarity Reviews (right)

Semi-supervised learning and more specifically, graph regularization can be really powerful when the amount of training data is small. The lack of training data is compensated by leveraging similarity among the training samples, which is not possible in traditional supervised learning. We have used a supervision ratio of 0.3 (30% of the labeled data) for training both the base model as well as the graph-regularized model. We illustrate the impact of the supervision ratio on model accuracy in the figure 3.

It can be observed that as the supervision ratio decreases, model accuracy also decreases. This is true for both the base model and for the graph-regularized model, regardless of the model architecture used. However, notice that the graph-regularized model performs better than the base model for both the architectures. In particular, for IMDB dataset, when the supervision ratio is 0.01, the accuracy of the graph-regularized model is 20% higher than that of the base model. This is primarily because of semi-supervised learning for the graph-regularized model, where structural similarity among training samples is used in addition to the training samples themselves.



Dataset	Base Model	Test Accuracy of Base Model		Test Accuracy of Adversarial Regularized Model	
		Without Perturbation	With Perturbation	With Perturbation	Without Perturbation
MNIST	CNN + FFNN	99%	45.1%	99.2%	94.7%
FASHION MNIST	CNN + FFNN	90%	62.9%	89.8%	74.3%
CIFAR 10	CNN + FFNN + Dropout	69.3%	21.1%	68.7%	44.1%

Table 1: Test Accuracies

Case 3: Effect of Neural Structured Framework on Adversarial Training

The core idea of adversarial learning is to train a model with adversarially perturbed data (called adversarial examples) in addition to the organic training data. The adversarial examples are constructed to intentionally mislead the model into making wrong predictions or classifications. By training with such examples, the model learns to be robust against adversarial perturbation when making predictions.

The datasets we used are MNIST, Fashion MNIST and CIFAR 10. The MNIST dataset contains grayscale images of handwritten digits (from ’0’ to ’9’). Each image shows one digit at low resolution (28-by-28 pixels). There are 60,000 training examples and 10,000 test examples. The task involved is to classify images into 10 categories, one per digit. Fashion-MNIST is a dataset of Zalando’s article images—consisting of a training set of 60,000 examples and a test set of 10,000examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Our base model was a neural network consisting of 3 convolutional layers followed by 2 fully connected layers. We then performed adversarial regularization on the base model. The accuracies are shown in Table 1.

5. CONCLUSION AND FUTURE WORK

We have shown the efficacy of Neural Structured Learning on various tasks including document classification, sentiment classification and image classification problems, using various neural network architectures like FFNN, Bi-directional LSTM and CNN. Our experimental results showed that Neural Structured Learning almost always improves the accuracies of the base models. The graph regularized models were able to handle limited supervision better than the base models. As the supervision ratio was reduced the accuracy drop of these models was less than that of the base models. When adversarial perturbations were introduced to the classification problem the adversarial regularized model’s accuracy reduced by a far lower percentage than that of the base model. The results show that training models with structure signals help the models to become robust and accurate.

While we have shown the advantages of using Neural Structured Learning in various problems, thereare still improvements that can be made. We leave this as future work. Computation efficiency to generate structure (graph) can be improved. We computed Graph using a simple similarity metric. Better metrics such as SSIM(Structural Similarity Index) and LPIPS(Learned Perceptual Image Patch Similarity) can be used to produce the graph. The model was trained against adversarial inputs using FGSM. Recently many other defense mechanisms have been proposed such as defensive distillation can used to further test the robustness of NSL.

6. REFERENCES

[1] Dong-Hyun Lee. 2013. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In ICML 2013 Workshop: Challenges
in Representation Learning (WREPL).

[2] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label propagation and quadratic criterion. In Semi-supervised learning, O Chapelle, B Scholkopf, and A Zien (Eds.). MIT Press, 193–216

[3] Xiaojin Zhu and Zoubin Ghahramani. [n. d.]. Learning from labeled and unlabeled data with label propagation.

[4] Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In International Conference on Machine Learning.

[5] Sujith Ravi and Qiming Diao. 2016. Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation. In Proceedings of the 19th International
Conference on Artificial Intelligence and Statistics. 519–528.

[6] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In Advances in Neural Information Processing Systems. 321–328

[7] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples.
Journal of machine learning research 7, Nov (2006), 2399–2434

[8] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep learning via semi-supervised embedding. In Neural Networks: Tricks of the Trade.

Springer, 639–655.

[9] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting SemiSupervised Learning with Graph Embeddings. In Proceedings of The 33rd International Conference on Machine Learning. 40–48

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems. 3837–3845.

[11] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv preprint arXiv:1609.02907 (2016).

[12] Thang D Bui, Sujith Ravi, Vivek Ramavajjala, Neural graph learning: Training neural networks using graphs, Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 64-71

[13] Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy, Explaining and Harnessing Adversarial Examples, Published as a conference paper at ICLR 2015

[14] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[15] Chun-Sung Ferng, Arjun Gopalan, Allan Heydon, Yicheng Fan, Chun-Ta Lu, Philip Pham and Andrew Tomkins. NSL Framework.

[16] Noam Shazeer, Ryan Doherty, Colin Evans, Chris Waterson. Swivel: Improving Embeddings by Noticing What's Missing. CoRR abs/1602.02215 (2016)