Matrix Completion for Graph-Based Deep Semi-Supervised Learning

Fariborz Taherkhani, Hadi Kazemi, Nasser M. Nasrabadi
Lane Department of Computer Science and Electrical Engineering
West Virginia University
fariborztaherkhani@gmail.com, hakazemi@mix.wvu.edu, nasser.nasrabadi@mail.wvu.edu

Abstract

Convolutional Neural Networks (CNNs) have provided promising achievements for image classification problems. However, training a CNN model relies on a large number of labeled data. Considering the vast amount of unlabeled data available on the web, it is important to make use of these data in conjunction with a small set of labeled data to train a deep learning model. In this paper, we introduce a new iterative Graph-based Semi-Supervised Learning (GSSL) method to train a CNN-based classifier using a large amount of unlabeled data and a small amount of labeled data. In this method, we first construct a similarity graph in which the nodes represent the CNN features corresponding to data points (labeled and unlabeled) while the edges tend to connect the data points with the same class label. In this graph, the missing label of unsupervised nodes is predicted by using a matrix completion method based on rank minimization criterion. In the next step, we use the constructed graph to calculate triplet regularization loss which is added to the supervised loss obtained by initially labeled data to update the CNN network parameters.

Introduction

CNN models require vast amounts of labeled data to be trained properly; however, providing reliable annotated data to train the CNN models tends to be expensive. There are essentially two principal solutions that are usually used to deal with this challenge: 1) Transfer Learning (TL) and 2) Semi-Supervised Learning (SSL). In TL methods (Weiss, Khoshgoftaar, and Wang 2016), we enhance new task learning via transfer of knowledge from a related task which has already been learned. In SSL methods (Zhu 2005), however, we are motivated by the fact that in a lot of applications, there are a vast amount of unlabeled data but only a small amount of labeled data and we essentially aim to learn discriminative learning methods that can make use of the information about the input distribution that is given by a large amount of unlabeled data. The SSL is a broad research field which has been used in variety of applications such as image search (Fergus, Weiss, and Torralba 2009) and natural language processing (Liang 2005). Among the recent SSL approaches, the GSSL methods have received a lot of attention and have become popular due to their flexibility in practical applications and low computational complexity. In GSSL methods, one assumes that the data points (both labeled and unlabeled) are embedded in a low-dimensional manifold which might be reasonably represented by a graph. In GSSL methods, each data point is expressed as a node in a graph and weights between nodes provide a measure of similarity between them. In GSSL, we inject seed labels on a subset of the nodes and then we infer labels on the unlabeled nodes in the graph. The intuition behind the similarity graph is that it captures the information from the labeled samples which is then propagated through to the unlabeled samples within the graph.

In this paper, we propose a novel iterative GSSL algorithm to train a CNN-based classifier. Our GSSL algorithm uses a new method to construct a similarity graph by leveraging matrix completion method based on rank minimization criteria. Once the similarity graph is constructed, we use it to regularize the fully supervised loss (i.e., given by initially labeled data points) to force that connected data points in the graph (i.e., data points which belong to the same class) share similar feature representations while disconnected ones have different representations.

The entire framework is trained end to end such that in each training iteration, the feature representations computed from the CNN are used to construct the similarity graph, then the graph is used to calculate triplet regularization loss which is added to the supervised loss to update the parameters of the network which provides new feature representations for the next iteration.

Related Work

Deep Semi-Supervised Learning. Deep learning models with SSL algorithms have been attempted by several groups (Laine and Aila 2016; Chongxuan et al. 2017; Rasmus et al. 2015; Donahue, Krähenbühl, and Darrell 2016). Most of deep SSL approaches leverage the idea of adversarial training; however, these approaches suffer from a range of disadvantages including training instability, lack of topology generalization, and computational complexity (Arjovsky, Chintala, and Bottou 2017). Salimans et al. (Salimans et al. 2016) use a model to improve the effectiveness of Generative Adversarial Networks (GAN) for SSL applications. The model provides a new technique in which the performance of supervised task is improved by learning on additional unlabeled samples. The model consists of two deep networks which are trained jointly as in typical GAN framework. The first network is a generative model that generates new samples while the other network is the discriminative network. The main problem of this method is training instability, and extra time and memory cost spent to train the two deep networks. Rasmus et al. (Rasmus et al. 2015) merge supervised with unsupervised learning methods using a deep learning model. The model is trained to minimize the sum of supervised and unsupervised cost functions by using back propagation, and at the same time prevents the need for layer-wise pre-training. The main problem of this method is the lack of a clear path to generalize it to other network topologies, such as recurrent or residual networks. The probabilistic formulation of CNN models proposed in (Patel, Nguyen, and Baraniuk 2016) natively supports SSL introduced by using a new family of hierarchical generative models. However, the main concern of these methods is that the activation function requires to be ReLU and that the overall network topology follow a CNN. There are some other methods which extend the generative models for the SSL; for example, Maaløe et al. (Maaløe et al. 2016a) extend GANs with auxiliary variables to learn better variational approximations and more expressive variational distribution. Tobias et al. (Springenberg 2015) propose categorical GAN (CatGAN) which replaces the binary discriminator in the standard GAN with a multi-class classifier, and trains the generator and the discriminator using mutal information on unlabeled samples. Kingma et al. (Kingma et al. 2014) use conditional Variational Autoencoders (VAEs) to treat labels as conditions of generative models to describe the input data; they make posterior inference of labels given unlabeled samples to generate a particular class of samples.

Graph-Based Semi-Supervised Learning. The GSSL approaches generally contains two main steps. In the first step, the graph is constructed from all the data points (both labeled and unlabeled data) to represent the relationship between them while in the second step, the information from the labeled data is propagated to the unlabeled data over the graph. Among the different GSSL methods which formulate the information propagation step by using different objective functions such as low-rank minimization (Zheng et al. 2013), k-nearest neighbor methods (Anastasiu and Karypis 2015), structured sparsity (Zhou, Lu, and Peng 2013) mincut (Blum and Chawla 2001), energy minimization (Blum et al. 2004) and Laplacian spectral method (Fergus, Weiss, and Torralba 2009), there is one common assumption which states that the data points on the same structure (i.e., manifold, cluster or subspace) more likely have the same label. In fact, GSSL methods tend to model the structural density among the data points by measuring the proximity (similarity) between data points in the graph and then propagate information of labeled data to the unlabeled data in a way that the missing labels in the graph are predicted based on the closest labeled data points (e.g, k-nearest neighbors classifier). Since normally there is no explicit solution to model the underlying structures of the data in the feature space, a graph created from the data usually serves as an approximation of the real structure. As a result, constructing a proper graph that best captures the main structure of the data point is important to all GSSL approaches (Berton and de Andrade Lopes 2015).

Matrix Completion Based on Rank Minimization. The problem of completing a low-rank matrix from a few sampled entries has been successfully applied in a variety of applications such as the Netflix challenge. A major breakthrough by Candes et al. (Candès and Recht 2009) states that minimizing a matrix rank subject to some constrains can be recast as minimizing the nuclear norm (sum of singular values) of the matrix. Since nuclear norm minimization of a matrix has characteristic of a Semidefinite Programming (SDP), many approaches have been proposed to solve this minimization problem effectively (Fazel 2002). In the field of computer vision and machine learning, nuclear norm minimization has been applied to many problems such as robust PCA (Wright et al. 2009) and subspace segmentation (Liu, Lin, and Yu 2010).

Similarity Graph Construction

In this section, we describe our method to construct the similarity graph via the matrix completion method. Let $g(x_1), ..., g(x_n) \in \mathbb{R}^d$ be the features captured by the CNN model corresponding to *n* samples; each of these samples is represented by a node in the similarity graph. Let $\mathbf{X} = [g(x_1), g(x_2), ..., g(x_n)]$ be a $d \times n$ feature matrix constructed by stacking samples column wise. Suppose that c is the number of classes; $y_1, ..., y_n$ are one hot encoding label of samples, and $\mathbf{Y} = [y_1, ..., y_n]$ indicates a $c \times n$ label matrix which is obtained by a linear model from $\mathbf{X}$ (i.e., $y_i = \mathbf{W}g(x_i) + b$, where $\mathbf{W}$ is a $c \times n$ weight matrix). In our problem, all the entries in feature matrix $\mathbf{X}$ are known and observable; however, entries corresponding to unlabeled samples in the label matrix $\mathbf{Y}$ are missed and we essentially aim to predict them. We note that by assuming $\mathbf{X}$ as a low rank matrix, the combined $(c+d) \times n$ matrix $\mathbf{Z} = [\mathbf{X}; \mathbf{Y}]$ produces a low rank matrix too (i.e., rank $([\mathbf{X}; \mathbf{Y}]) \leq$ rank $(\mathbf{X})+c$); and predicting the missing entries in this matrix can be cast as a matrix completion problem (Cai, Candès, and Shen 2010; Cabral et al. 2011). Here, we take advantage of the matrix completion method to predict missing labels in the graph.

Similar to other GSSL methods, we assume that the data points are embedded within a low-dimensional manifold; indeed we make a structural assumption about the data points. This assumption causes the feature matrix $\mathbf{X}$ to have a low rank or approximately have a low rank, and consequently it provides the fact that the matrix $\mathbf{Z}$ should also have a low rank. Therefore, we set up an optimization problem to predict missing labels in the graph where the decision variable $\mathbf{Z}$ is the data matrix with the missing labels. We aim to find the matrix $\mathbf{Z}$ which matches the observed entries (i.e., labels of supervised data and matrix $\mathbf{X}$) and has the minimum rank.

However, rank minimization is an NP hard problem. To the best of our knowledge, the best algorithm we know to minimize the rank of a matrix under linear constraint is doubly exponential in the size of the matrix (Chistov and Grigor'Ev 1984), which means that when the size of the matrix is more than seven, it is not practical to solve it on the computer. To relax the rank term in our optimization problem, we use the heuristic proposed in Candes et al. (Candès and Recht 2009); the heuristic uses an alternative method which minimizes the nuclear norm $||\mathbf{Z}||_* = \sum_{k=1}^{n} \sigma_k(\mathbf{Z})$ (the sum of the singular values) under the constraint set over the observed entries; where $\sigma_k(\mathbf{Z})$ denotes the k^{th} largest singular value of $\mathbf{Z}$. The nuclear norm of matrix $\mathbf{Z}$ is the dual norm of the spectral norm of the matrix $\mathbf{Z}$ which is convex, and it can be solved by variety of convex optimization algorithms (Fazel 2002). The relationship between rank and nuclear norm is similar to that of ℓ_0 norm and ℓ_1 norm for vectors. Since we can not minimize the rank of the matrix, we choose the nuclear norm as an alternative optimization problem which is the tightest convex relaxation to the rank.

Predicting Missing Labels via Matrix Completion

In this section, we provide an optimization problem to predict missing entries of the data matrix $\mathbf{Z} = [\mathbf{X}; \mathbf{Y}]$ such that the nuclear norm of $\mathbf{Z}$ is minimized and entries in $\mathbf{Z}$ match the observed entries (i.e., labels of supervised data and feature matrix $\mathbf{X}$). Let $\Omega_{\mathbf{X}}$ be the index set of observed entries in the feature matrix $\mathbf{X}$, where $(i, j) \in \Omega_{\mathbf{X}}$ if and only if $\mathbf{X}_{ij}$ is observed. Likewise, let $\Omega_{\mathbf{Y}}$ be the index set of observed entries in the label matrix $\mathbf{Y}$ and $(i, j) \in \Omega_{\mathbf{Y}}$ if and only if y_j is a labeled sample. In this problem $|\Omega_{\mathbf{X}}| = d \times n$ (no missing entries in $\mathbf{X}$), and $1 < |\Omega_{\mathbf{Y}}| < c \times n$ (some missing entries in $\mathbf{Y}$). The optimization problem for predicting missing entries in $\mathbf{Z}$ is defined as follow:

$$\begin{aligned} \underset{\mathbf{Z}}{\text{argmin}} \;\; & \mu||\mathbf{Z}||_* + \frac{1}{|\Omega_{\mathbf{X}}|} \sum_{i,j \in \Omega_{\mathbf{X}}} c_x(z_{ij}, x_{ij}) + \\ & \frac{\lambda}{|\Omega_{\mathbf{Y}}|} \sum_{i,j \in \Omega_{\mathbf{Y}}} c_y(z_{(i+d)j}, y_{ij}). \end{aligned} \tag{1}$$

We shift row index of the stacked matrix $\mathbf{Z}$ in the $c_y(z_{(i+d),j}, y_{ij})$ because we want to skip $\mathbf{X}$ part in the $\mathbf{Z}$. Apart from minimizing the nuclear norm of $\mathbf{Z}$, we penalize the cost function in (1) by adding $c_y(.)$ and $c_x(.)$ losses to avoid trivial solutions and large distortions of $\mathbf{Z}$ from the observed entries in $\mathbf{X}$ and $\mathbf{Y}$ matrices. The observed label data type is of a different type than the observed feature data; thus we define two different losses. The $c_x(z_{ij}, x_{ij}) = \frac{1}{2}(z_{ij} - x_{ij})^2$ is defined as the squared loss, while the $c_y(z_{(i+d)j}, y_{ij}) = \log(1 + \exp(-z_{(i+d)j}.y_{ij}))$ is the logistic loss which accentuates the error on entries switching labels as is different from their absolute numerical deviation. The parameters μ, λ are the weights which create a balance between errors for better label error correction and feature adaptation.

Optimization Method The loss defined in (1) is a convex optimization problem. We use the soft-impute algorithm (Mazumder, Hastie, and Tibshirani 2010) which is a simple and effective algorithm for nuclear norm regularized matrix completion. This algorithm iteratively restore the missing entries with those attained from a soft-thresholded SVD. We first define a projection operator P_Ω on the observed set Ω as follows:

$$[P_\Omega(\mathbf{Z})]_{ij} = \begin{cases} z_{ij} & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega \end{cases} \tag{2}$$

thus, the optimization problem in (3) can be rewritten as follows:

$$\begin{aligned} f(\mathbf{Z}) = & \underbrace{\mu||\mathbf{Z}||_*}_{G(\mathbf{Z})} + \underbrace{\frac{1}{2|\Omega_{\mathbf{X}}|}||P_\Omega(\mathbf{X}) - P_\Omega(\mathbf{Z}_x)||_F^2}_{H(\mathbf{Z}_x)} + \\ & \underbrace{\frac{\lambda}{|\Omega_{\mathbf{Y}}|} log(1 + exp(-P_\Omega(\mathbf{Z}_y) \circ P_\Omega(\mathbf{Y}))),}_{U(\mathbf{Z}_y)} \end{aligned} \tag{3}$$

where, $\mathbf{Z}_x$ and $\mathbf{Z}_y$ are sub-matrices of $\mathbf{Z}$ for parts $\mathbf{X}$ and $\mathbf{Y}$, respectively (i.e., $\mathbf{Z} = [\mathbf{X}; \mathbf{Y}]$), and symbol $\circ$ indicates Hadamard or element wise product between $P_\Omega(\mathbf{Z}_y)$ and $P_\Omega(\mathbf{Y})$. In this set of formulation, we call the second and third parts $H(\mathbf{Z}_x)$, $U(\mathbf{Z}_y)$, respectively which are convex and smooth, and the first part $G(\mathbf{Z})$ which is also convex but not smooth. Therefore, we can think about three ingredients needed for proximal gradient descent:

• The first is $\nabla H(\mathbf{Z}_x)$; here subgradient is just the gradient:

$$\nabla H(\mathbf{Z}_x) = -\frac{1}{|\Omega_{\mathbf{X}}|}(P_\Omega(\mathbf{X}) - P_\Omega(\mathbf{Z}_x)), \tag{4}$$

where, the gradient of $H(z_{ij})$ is $-\frac{1}{|\Omega_{\mathbf{X}}|}(x_{ij} - z_{ij})$.

• The second is $\nabla U(\mathbf{Z}_y)$; here, subgradient is also just the gradient:

$$\nabla U(\mathbf{Z}_y) = \frac{\lambda}{|\Omega_{\mathbf{Y}}|} \frac{-P_\Omega(\mathbf{Y})}{1 + exp(P_\Omega(\mathbf{Z}_y) \circ P_\Omega(\mathbf{Y}))}, \tag{5}$$

where, the gradient of $U(z_{(i+d)j})$ is $\frac{\lambda}{|\Omega_{\mathbf{Y}}|} \frac{-y_{ij}}{1+exp(z_{(i+d)j} y_{ij})}$.

• The third is the *prox* operator:

$$prox_t(\mathbf{Z}) = \arg\min_{\mathbf{C}} \frac{1}{2t}||\mathbf{Z} - \mathbf{C}||_F^2 + \mu||\mathbf{C}||_*. \tag{6}$$

It can be proved that $prox_t(\mathbf{Z}) = S_{\lambda t}(\mathbf{Z})$ (Mazumder, Hastie, and Tibshirani 2010) which is the *matrix soft thresholding at the level* λ; where $S_\lambda(\mathbf{Z})$ is defined by

$$S_\lambda(\mathbf{Z}) = U \Sigma_\lambda V^\top, \tag{7}$$

where, $\mathbf{Z} = U \Sigma_\lambda V^\top$ is an SVD, and Σ_λ is diagonal with $(\Sigma_\lambda)_{ii} = max\{\Sigma_{ii} - \lambda, 0\}$. This matrix soft-thresholding is an element-wise soft-thresholding of the matrix. By using a soft-threaded singular matrix Σ_λ , this returns a low-rank *prox* result.

Therefor, the Proximal Gradient update step is written as follows:

$$\begin{aligned} \mathbf{Z}^+ = & prox_t(\mathbf{Z} - t(\nabla H(\mathbf{Z}_x) + \nabla U(\mathbf{Z}_y))) \\ = & S_{\lambda t}(\mathbf{Z} - t(\nabla H(\mathbf{Z}_x) + \nabla U(\mathbf{Z}_y)). \end{aligned} \tag{8}$$

Note that $(\nabla H(\mathbf{Z}_x) + \nabla U(\mathbf{Z}_y))$ is Lipschitz continuous with $L = 1$, thus we can choose fixed step size $t = 1$ and then the update step in this case is expressed as follows:

$$\mathbf{Z}^+ = S_\lambda(\mathbf{Z} - (\nabla H(\mathbf{Z}_x) + \nabla U(\mathbf{Z}_y)). \tag{9}$$

Once the label of unlabeled data are predicted by matrix completion; in the next step, the nodes in the graph are connected to each other if they belong to the same class and they are disconnected otherwise.

Supervised Task Regularization via the Constructed Graph

In this section, we provide a new approach based on the triplet loss function to leverage the constructed graph for training the CNN model. In this approach, we aim to regularize the supervised task by adding a semi-supervised loss term as an auxiliary task to the CNN. In other words, we concentrate mostly on the classifier regularization learned in a supervised fashion with few labeled data. To regularize the supervised task using unsupervised data, we apply the triplet loss function using a triplet of data on the graph as follows;

$$\mathcal{L}_{trip}(g(x_a), g(x_p), g(x_n), A) = \\ max(||g(x_a) - g(x_p)||^2 - ||g(x_a) - g(x_n)||^2 + \alpha, 0), \tag{10}$$

where A is the adjacency matrix of the graph in which entries in the matrix indicate if pairs of the nodes are adjacent or not in the graph; the α parameter is the margin in the triplet loss and $g(x_a), g(x_p), g(x_n)$ are the output of CNN for the images x_a, x_p and x_n, respectively. In triplet loss, we look at the three data point on the graph at the same time; in our case we choose labeled samples in the graph as the anchor (i.e., sample x_a in (10)) and sample x_p as a positive sample if $A(x_a, x_p) = 1$ and sample x_n as a negative sample if $A(x_a, x_n) = 0$; we want to bring positive and anchor pairs (i.e., two images which are connected in the graph) close to each other while push away the negative and anchor pairs (i.e., two images which are not connected in the graph) simultaneously. The triplet loss is added to the total network loss function to regularize the supervised classification loss with an auxiliary semi-supervised loss term.

Deep Semi Supervised Loss Function

Now, we have all the loss terms including the supervised loss and semi-supervised loss to set up the total CNN loss function. Our SSL loss function for updating the CNN parameters is defined as follows:

$$\mathcal{L}(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \underbrace{\sum_{x_i \in \mathbf{x}_s} \mathcal{L}_c(\mathbf{w}, g(x_i), y_i) +}_{\text{supervised}} \\ \underbrace{\gamma\left(\sum_{x_a \in \mathbf{x}_s; \{x_p, x_n\} \in \mathbf{x}} \mathcal{L}_{trip}(\mathbf{w}, g(x_a), g(x_p), g(x_n), A) \right)}_{\text{semi-supervised}}, \tag{11}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the training batch and samples labels in the training batch (i.e., real labels and predicted labels); $\mathcal{L}_c$ is supervised classification loss (softmax loss is used, but can be chosen other type of losses such as center loss, contrastive center loss); $\mathbf{x}_s$ is the supervised samples in training batch $\mathbf{x}$. The $\mathbf{w}$ is the parameters of the CNN and γ is the balancing term between two supervised and semi-supervised losses. The batch set is created such that number of similar (i.e., anchor and positive) and dissimilar (i.e., anchor and negative) pairs to be roughly balanced. In each triplet, the labeled data (not predicted labels) are chosen as the anchor and predicted labeled data are considered only as positive and negative samples in the batch.

Experiments

Datasets and Pre-Processing. We conducted our experiments on the widely used MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), small NORB (LeCun, Huang, and Bottou 2004) and CIFAR 10 (Krizhevsky and Hinton 2009) datasets. For each of these datasets, we split the training set to two different sets of labeled and unlabeled samples. We ensure that all the classes are balanced such that each class should have the same number of labeled samples. We ran our model for 10 times with different random splits of the labeled and unlabeled data for each dataset, and we report the mean and standard deviation of the error rate.

MNIST is handwritten digits of 10 different classes dataset which contains a training set of 60,000 samples, and a test set of 10,000 samples. The digits have been size-normalized and centered in a fixed-size (28×28) images. We select 100 samples in the training set as labeled and remaining of it as unlabeled.

SVHN is another digit dataset similar to MNIST with 32×32 color images centered around a single character. The task is to classify the center digits in the images; we follow (Sermanet, Chintala, and LeCun 2012; Goodfellow et al. 2013) methods to split the dataset to 598,388 training data and 26,032 testing data. For this dataset, we choose randomly 1000 samples as labeled and rest of it, is used as unlabeled.

CIFAR-10 is a collection of 32×32 RGB images of 10 classes including airplane, automobile, bird, cat, deer, dog, frog, horse, ship and trucks. This dataset contains 50,000 number of images for training and 10,000 for testing; we select 4,000 samples in the training set as labeled and rest as unlabeled.

NORB contains gray scale images of 5 general classes including animal, human, airplane, truck and car. The objects were imaged by two cameras under different lighting conditions, elevations and azimuths. This dataset contains 24,300 images for both training and testing sets. In our experiments, we resize the images to 32×32 as it is in (Maaløe et al. 2016b); we select 1,000 samples in the training set as labeled and remaining is considered as unlabeled.

Experimental Setup. Our CNN architecture has been shown in Fig.1; it is composed of three convolutional layers , three max pooling layers and one fully connected layer. There are 64 number of filters in the first convolutional layer and 128 number of filters in the second and third convolutional layers, respectively. The size filters in this architecture are 3×3 and the convolution stride is set to 1 pixel. To preserve spatial resolution after convolution, spatial padding of the convolutional layer is fixed to 1 pixel for all 3×3 convolutional layers. The max-pooling layers are placed after each convolutional layers, respectively; the max-pooling is carried out on a 2×2 pixel window with a stride of 2. We apply batch normalization (i.e., shifting inputs to zero-mean and unit variance) after each convolutional and fully connected layer, and before performing the Rectified Linear Units (ReLU) activation function. The batch normalization

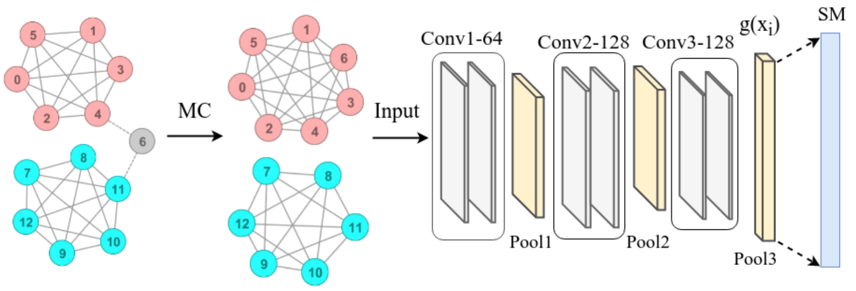


Figure 1: The labels of unsupervised nodes in the graph are predicted and then the graph is used to train the network.

Methods	MNIST (100)	SVHN (1000)	NORB (1000)	CIFAR-10 (4000)
Matrix Completion	1.98(±0.03)	10.06(±0.08)	9.11(±0.11)	19.91(±0.23)
GSCNN+No Reg	1.14(±0.09)	8.41(±0.22)	8.71(±0.18)	18.03(±0.31)
GSCNN	0.84(±0.12)	5.13(±0.39)	7.01(±0.53)	15.49(±0.64)

Table 1: Comparing GSCNN error rate in different scenarios on MNIST, SVHN, NORB and CIFAR-10 datasets.

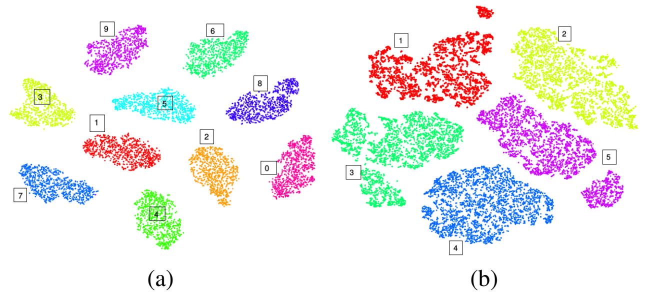


Figure 2: Visualizations of training data which are partially labeled for (a) MNIST and (b) NORB datasets using our GSSL method.

potentially helps to achieve faster learning as well as higher overall accuracy. Furthermore, batch normalization allows us to use a higher learning rate, which potentially provides another boost in speed. The parameters of the network are initialized by sampling randomly from $\mathcal{N}(0, 0.001)$ except for the bias parameters which are initialized as zero. We implemented our framework in TensorFlow and performed our experiments on two GeForce GTX TITAN X 12GB GPUs. We use Adam optimizer (Kingma and Ba 2014) with the default hyper-parameters values ($\epsilon = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) in our experiments. The batch size in all experiments is fixed to 128, and we set γ to 0.1 experimentally to create a balance between supervised loss and unsupervised loss in the total network loss function.

Hyper-Parameter Tuning. We used 10-fold cross validation in each experiment to tune hyper-parameters in our model. For λ in (1), we randomly divide the labeled data into ten disjoint subsets; next we run the matrix completion over $\frac{9}{10}$ and we calculate the performance on the remaining $\frac{1}{10}$; then we average the results over the 10 folds (We note that we used label error as performance criterion to select parameters because our goal in MC is to predict label of unlabeled data points). The range of λ values are $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. For μ in (1), we initialize it to be $0.25\sigma_1$, where σ_1 is the largest singular value of the matrix $[\mathbf{X}; \mathbf{Y}]$ and decrease it gradually until 10^{-5} as it is suggested in (Mazumder, Hastie, and Tibshirani 2010).

Matrix Completion. We use SoftImput algorithm implemented in fancyimput package in python to predict missing labels in the graph. We set shrinkage value which is the value by which we shrink singular values on each iteration to the maximum singular value of the initialized matrix (zeros for missing values) divided by 100; the maximum number of SVD iteration is set to 1000. In matrix completion, we stop the algorithm by defining a convergence threshold (0.001 in our experiments) which is the minimum ration difference between iterations (as a fraction of the Frobenius norm of the current solution). In SoftImput algorithm, a sequence of solutions are produced for which the criterion decreases to the optimal solution with every iteration. Convergence threshold can be given to the SoftImput algorithm implemented in fancyimput package.

Triplet Mining on the Graph. In our experiments, we concentrate on the online triplet mining strategy to generate useful triplets for data on the graph. Considering a batch of b samples, we extract CNN features for each sample, and we then can create a maximum of b^3 triplets even though most of these triplets are not valid (i.e., triplet except two positive and one negative). This technique provides us more triplets for a single batch of samples during the network training. We use all the valid triplets in the training and we average the loss on the hard triplet (i.e., $d(g(x_n), g(x_a) < d(g(x_p), g(x_a)))$ and semi-hard triplet (i.e., $d(g(x_a), g(x_p)) < d(g(x_a), g(x_p)) + \alpha$); we disregard the easy triplets (triplets with zero loss) because averaging on them makes the overall loss very small.

Effectiveness of Regularization based on Triplet. To show the effectiveness of our semi-supervised loss based on the triplet in training of our CNN model, we conducted experiments in two different scenarios, a) GSCNN+No Reg

Samples Per Class	100	200	400	800
NORB	$9.88(\pm 0.54)$	$7.01(\pm 0.53)$	$6.12(\pm 0.41)$	$5.07(\pm 0.19)$
CIFAR-10	$18.98(\pm 0.62)$	$16.82(\pm 0.47)$	$15.49(\pm 0.64)$	$14.51(\pm 0.34)$

Table 2: GSCNN error rate by given number of initially labeled data per class. Results are averaged on 10 times randomly split.

Methods	MNIST (100)	SVHN (1000)	NORB (1000)	CIFAR-10 (4000)
M_1 + M_2 (Kingma et al. 2014)	$3.33(\pm 0.14)$	$36.02(\pm 0.1)$	$18.79(\pm 0.05)$	-
VAT (Miyato et al. 2015)	2.33	24.63	9.88	-
Ladder (Rasmus et al. 2015)	$1.06(\pm 0.37)$	-	-	$20.40(\pm 0.47)$
CatGAN (Springenberg 2015)	$1.39(\pm 0.28)$	-	-	$19.58(\pm 0.46)$
ADGM (Maaløe et al. 2016a)	$0.96(\pm 0.02)$	22.86	$10.06(\pm 0.05)$	-
SDGM (Maaløe et al. 2016a)	$1.32(\pm 0.07)$	$16.61(\pm 0.24)$	$9.4(\pm 0.04)$	-
FM (Salimans et al. 2016)	$0.93(\pm 0.07)$	$8.11(\pm 1.3)$	-	$18.63(\pm 2.32)$
Triple GAN (Chongxuan et al. 2017)	$0.91(\pm 0.58)$	$5.77(\pm 0.17)$	-	$16.99(\pm 0.36)$
$\prod$ model (Laine and Aila 2016)	-	$5.43(\pm 0.25)$	-	$16.55(\pm 0.29)$
ALI (Donahue, Krähenbühl, and Darrell 2016)	-	$7.42(\pm 0.65)$	-	$17.99(\pm 1.62)$
GSCNN	**0.84**$(\pm$**0.12**$)$	**5.13**$(\pm$**0.39**$)$	**7.01**$(\pm$**0.53**$)$	**15.49**$(\pm$**0.64**$)$

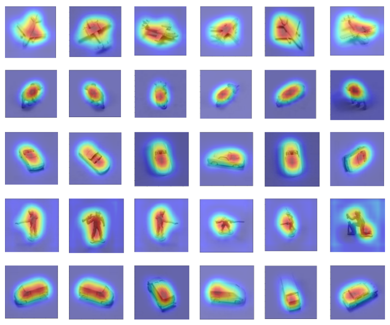
Table 3: Comparing SSL models on MNIST, SVHN, NORB and CIFAR-10 datasets

which indicates the case where we remove the triplet regularization loss in (11) and use all the labeled and unlabeled data with true and completed labels directly in the softmax classifier loss, b) GSCNN which indicates the case where we use the triplet regularization loss in (11) in training of our CNN model. Table.1 shows the performance of our model in two different scenarios (i.e., a) GSCNN+No Reg and b) GSCNN); the results show that our regularization loss based on the triplet can improve the model performance by 0.3%, 3.28%, 1.7% and 2.54% on the MNIST, SVHN, NORB and CIFAR-10 datasets, respectively. This improvement is because the softmax classifier loss only forces the CNN features of different classes to stay apart, while the triplet loss not only does this, but also efficiently brings the CNN features of the same class close to each other. Therefore, by considering triplet regularization in the training, not only the inter-class features differences are enlarged, but also the intra-class features variations are reduced. Moreover, since our method is an iterative process of two steps (i.e., the first step uses matrix completion to predict the labels, and the next step uses the predicted results to train a CNN), we reported in Table.1 the matrix completion error rates to provide an empirical analysis showing that matrix completion has significant influence on training the CNN model. Accuracy of the matrix completion indicates those unlabeled data predicted correctly by matrix completion, are then used in the regularizer to improve the CNN performance.

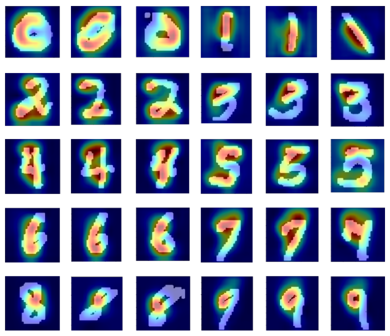
Constructed Graph Properties. The graph in our model is constructed dynamically while the CNN network is trained; this is because the graph in our model needs the network output to be constructed. In our SSL method, the graph is created online in a local scope (over a few samples of training set) which is virtually similar to the concept of training batch in the CNN models. In each training batch, the labels of the unsupervised data are predicted based on all the labeled data using our matrix completion method. Indeed, we take a batch of training data and then we predict the class label of unsupervised data to construct the graph for the batch. Our graph construction method is an Expectation Maximization (EM) like algorithm in a sense that in forward pass, the graph is constructed for a batch (including labeled and unlabeled data), and later on it is used as a regularizer in the network loss calculation to update the network parameters by back propagation in the backward pass. This property enables us to create a robust graph through the training step; because the graph is constructed by better set of CNN features as we train the CNN through several training epochs. This factor makes the graph construction method more robust in comparison to the offline based graph construction methods with static data embedding.

Most of graph construction algorithms are usually expensive in terms of time complexity. For example, graph construction using offline k-NN method in brute-force fashion is $\mathcal{O}(n^2)$ where n is number of training data. Even though there are other efficient methods (Zhang et al. 2013) to improve k-NN method in terms of time complexity, in most of the offline construction methods the time complexity usually makes graph construction step unpractical specially for the large scale datasets. However, splitting the data to small chunks of data with equal size which in our case is the batch makes the graph construction step more efficient, and also feasible for online computation and simultaneous with CNN training; because in this case, only a small portion of the training data is used to construct the graph. Our method is online and use small part of the whole data. The computationally demanding part of our graph construction algorithm is the equation (9) where we take SVD from a low-rank matrix $\mathbf{Z}$ to predict missing label of unsupervised nodes in the graph. For example, it takes around 6 (secs) to complete a 100×100 matrix in each iteration of our algorithm.

Effect of Number of Initially Labeled Data in the Model. Since supervised data points are taken as anchors while forming the triplets in our model, we conducted experiment showing the influence of the number of initially



(a)



(b)

Figure 3: Example of Grad-CAM generated for (a) NORB and (b) MNIST datasets from our GSSL; it is shown that highlighted regions are activated by Grad-CAM algorithm for different classes.

labeled data points on the final performance. In this experiment, we selected 100 samples per each class and gradually increased the number of samples to 200, 400, and 800, respectively; the results in Table.2 indicate that the model performance increases as the number of initially labeled data points in the regularizer increases. The trend of improvement is reported in Table.2. The results on the CIFAR-10 dataset show the model is improved by around 2.16%, 1.33% and 0.98% when we increase the number of labeled samples from $100 \rightarrow 200$, $200 \rightarrow 400$ and $400 \rightarrow 800$, respectively, while on the NORB dataset, the model performance is improved by around 2.87%, 0.89% and 1.05% when we increase the number of labeled samples from $100 \rightarrow 200$, $200 \rightarrow 400$ and $400 \rightarrow 800$, respectively.

Evaluation and Discussion. We compare our GSSL model with a large body of previous models on MNIST, SVHN, NORB and CIFAR-10 datasets using 100, to , 4000 labeled samples, respectively. Experimental results in Table.3 show that our method is competitive to the state-of-the-art results for all these datasets; given 100 labeled samples on the MNIST dataset, our method still is comparable to the outstanding generative models including FM (Salimans et al. 2016) and Triplet GAN (Chongxuan et al. 2017); Table.3 also shows Semi-Supervised results on the more challenging datasets including SVHN, NORB and CIFAR-10 datasets. Following previous models (Maaløe et al. 2016a; Kingma et al. 2014; Miyato et al. 2015), we use 1,000 labeled samples on SVHN and NORB datasets to compare our method with other methods. The results show that our method outperforms the previous state-of-the-art.

Inspired by the Grad-CAM (Selvaraju et al. 2016) on class activation map, we can interpret the classification decision made by our method. We can see that our model is triggered by different semantic regions of the image for different classes of classification. Fig. 3 shows that our GSSL method by using Grad-CAM method provides "visual explanations" for decisions from the all classes of the CNN models. The Fig. 3 indicate the class activation of the model for MNIST and NORB dataset where we use 100 and $1,000$ labeled samples to train the model and use the remaining as test; the result shows the outstanding result of the model in object localization using Grad- CAM technique. We also used T-SNE (Maaten and Hinton 2008) to visualize the CNN features for training data which are partially supervised. Fig. 2 indicates that the model has acceptable discriminative ability; we applied this method on MNIST and NORB datasets using 100 and 1,000 labeled samples in the training step. The figure shows that our model can discriminate the training data in the embedded space using partially supervised samples.

Conclusion

In this paper, we proposed a new Graph-based Semi-Supervised Learning method to train a CNN model using a vast amount of unsupervised data in conjunction with a small amounts of supervised data. In this model, we make structural assumption about the data point to predict the missing labels on the graph and then we leverage the constructed graph as regularizer to train the CNN model. Experimental results show that our model is comparable to the state of the art for Semi-Supervised image classification.

References

Anastasiu, D. C., and Karypis, G. 2015. L2knng: Fast exact k-nearest neighbor graph construction with l2-norm pruning. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 791–800. ACM.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Berton, L., and de Andrade Lopes, A. 2015. Graph construction for semi-supervised learning. In *IJCAI*, 4343–4344.

Blum, A., and Chawla, S. 2001. Learning from labeled and unlabeled data using graph mincuts.

Blum, A.; Lafferty, J.; Rwebangira, M. R.; and Reddy, R. 2004. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, 13. ACM.

Cabral, R. S.; Torre, F.; Costeira, J. P.; and Bernardino, A. 2011. Matrix completion for multi-label image classification. In *Advances in Neural Information Processing Systems*, 190–198.

Cai, J.-F.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20(4):1956–1982.

Candès, E. J., and Recht, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* 9(6):717.

Chistov, A. L., and Grigor'Ev, D. Y. 1984. Complexity of quantifier elimination in the theory of algebraically closed fields. In *International Symposium on Mathematical Foundations of Computer Science*, 17–31. Springer.

Chongxuan, L.; Xu, T.; Zhu, J.; and Zhang, B. 2017. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems*, 4091–4101.

Donahue, J.; Krähenbühl, P.; and Darrell, T. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Fazel, M. 2002. *Matrix rank minimization with applications*. Ph.D. Dissertation, PhD thesis, Stanford University.

Fergus, R.; Weiss, Y.; and Torralba, A. 2009. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, 522–530.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 3581–3589.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Laine, S., and Aila, T. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Y.; Huang, F. J.; and Bottou, L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, II–104. IEEE.

Liang, P. 2005. *Semi-supervised learning for natural language*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Liu, G.; Lin, Z.; and Yu, Y. 2010. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 663–670.

Maaløe, L.; Sønderby, C. K.; Sønderby, S. K.; and Winther, O. 2016a. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.

Maaløe, L.; Sønderby, C. K.; Sønderby, S. K.; and Winther, O. 2016b. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Mazumder, R.; Hastie, T.; and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research* 11(Aug):2287–2322.

Miyato, T.; Maeda, S.-i.; Koyama, M.; Nakae, K.; and Ishii, S. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

Patel, A. B.; Nguyen, M. T.; and Baraniuk, R. 2016. A probabilistic framework for deep learning. In *Advances in neural information processing systems*, 2558–2566.

Rasmus, A.; Berglund, M.; Honkala, M.; Valpola, H.; and Raiko, T. 2015. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 3546–3554.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2234–2242.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2016. Grad-cam: Visual explanations from deep networks via gradient-based localization. *See https://arxiv. org/abs/1610.02391 v3* 7(8).

Sermanet, P.; Chintala, S.; and LeCun, Y. 2012. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, 3288–3291. IEEE.

Springenberg, J. T. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*.

Weiss, K.; Khoshgoftaar, T. M.; and Wang, D. 2016. A survey of transfer learning. *Journal of Big Data* 3(1):9.

Wright, J.; Ganesh, A.; Rao, S.; Peng, Y.; and Ma, Y. 2009. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, 2080–2088.

Zhang, Y.-M.; Huang, K.; Geng, G.; and Liu, C.-L. 2013. Fast knn graph construction with locality sensitive hashing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 660–674. Springer.

Zheng, Y.; Zhang, X.; Yang, S.; and Jiao, L. 2013. Low-rank representation with local constraint for graph construction. *Neurocomputing* 122:398–405.

Zhou, G.; Lu, Z.; and Peng, Y. 2013. L1-graph construction using structured sparsity. *Neurocomputing* 120:441–452.

Zhu, X. 2005. Semi-supervised learning literature survey.