Semi-supervised Hyperspectral Image Classification via Neighborhood Graph Learning

Daniel Jiwoong Im, *Student Member, IEEE,* Graham W. Taylor

***Abstract*—In problems where labeled data is scarce, semi-supervised learning (SSL) techniques are an attractive framework that can exploit both labeled and unlabeled data. These approaches typically rely on a smoothness assumption such that examples that are similar in input space should also be similar in label space. In many domains, such as remotely sensed hyperspectral image (HSI) classification, the data violates this assumption. In response, we propose a general method by which a neighborhood graph used in SSL is learned using binary classifiers that are trained to predict whether a pair of pixels shares the same label. Working within the framework of semi-supervised neural networks (SSNN), we show that our approach improves on the performance of the SSNN on two HSI datasets.**

***Index Terms*—semi-supervised learning, neural networks, hyperspectral image classification, aerial image analysis**

I. Introduction

SCENE understanding from remotely sensed images is a challenging task in machine learning. One key concern faced is the diversity of data collected from optical sensors such as multispectral and hyperspectral imagery (HSI). These open up new areas of application but pose new methodological challenges in data analysis [1]. Another key challenge is the relative scarcity of labeled data for training and evaluation.

To this end, semi-supervised learning (SSL) techniques are an attractive framework that can exploit both labeled and unlabeled information [2]. A number of SSL techniques have been applied to multispectral and hyperspectral classification, including Transductive SVMs (TSVM) [3], Laplacian SVMs (LapSVM) [4], and Graph-based methods [5]. Due to computational complexity (at least quadratic in the number of examples), they do not scale to millions of unlabeled pixels and are therefore not suitable for large-scale remote sensing applications. [6] proposed semi-supervised neural networks (SSNN) to label scenes from hyperspectral data. Trained by gradient-based methods, these computationally efficient networks were shown to outperform existing SSL approaches.

In practice, all of the above techniques build a neighborhood graph based on Euclidean distance in input space. They are based on a smoothness assumption that examples that are similar in input space (e.g., spectra) should also be similar in label space (e.g., ground cover). The Euclidean distance cannot capture complex, nonlinear effects such as class mixing within pixels, or the fact that certain regions of the spectra may be more important than others in distinguishing among classes. The distance is also fixed rather than adapted to the data. This problem is well-recognized and motivates the many approaches to similarity or distance metric learning [7]. On the other hand, these methods tend to overfit when labeled data is limited [8], as is often the case in remote sensing.

D.J. Im and G.W. Taylor are with the School of Engineering, University of Guelph, Guelph, ON, Canada.

The main contribution of this paper is a simple and novel alternative for learning the neighborhood graph for a wide class of SSL algorithms. The intuition behind our approach is that labels should be exploited for graph learning, but when they are scarce, we should use them in a simpler task that can be solved by a model that is less complex and therefore less prone to overfitting. The simple models we propose are binary classifiers trained to predict whether two inputs are from the same or different class. We experiment with two neural network-based architectures: a classical and multiplicative interaction network. By adopting the SSNN framework, our approach markedly improves the performance on two HSI classification datasets.

II. Background

The remote sensing community was an early adopter of semi-supervised variants of support vector machines (SVMs), and these models were shown to work reasonably well on HSI classification [1], [4], [9]. The downside of SVMs is that the methods they rely on to solve convex programs are at least quadratic in the number of examples, so they do not scale to large datasets. This is a problem when one wants to exploit many unlabeled examples. To address this concern, [6] proposed a neural network framework, to which the ideas of the Laplacian Eigenmap and the regularizer in the transductive SVM could be naturally extended. This methodology was named the semi-supervised neural network (SSNN). One advantage of using neural networks is that they can be trained by stochastic gradient descent (SGD) which is linear in the number of examples and naturally extends to on-line learning. Furthermore, they can extract multiple layers of learned, nonlinear representations rather than relying on engineering kernels. For these reasons, we adopt the SSNN as a general framework for building computationally efficient semi-supervised methods, and briefly review its formulation. We refer the reader to [6] for a more detailed presentation.

The general formulation for the SSNN is to minimize the following cost function:

$$\mathcal{L} = \frac{1}{l}\sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \lambda_u \frac{1}{l+u}\sum_{i,j=1}^{l+u} L(f_i, f_j, W_{ij})$$

which is composed of two terms. $V(\cdot)$ is the hinge loss reflecting the errors on labeled data. It sums over l labeled data points $\{\mathbf{x}_i, y_i\}$. $L(\cdot)$ is the regularizer whose form is discussed

Algorithm 1 Training scheme of the SSNN (adapted from [6])

Require: labeled pixels $\{\mathbf{x}_i, y_i\}$ and unlabeled pixels $\{\mathbf{x}_j^*\}$
repeat
Pick a random labeled example $\{\mathbf{x}_i, y_i\}$
Make a gradient step to optimize $V(\mathbf{x}_i, y_i, f)$
Pick a random unlabeled pair of neighbors $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$
Make a gradient step to optimize $\lambda_u L\left(f\left(\mathbf{x}_i^*\right), f\left(\mathbf{x}_j^*\right), 1\right)$
Pick a random unlabeled pair of non-neighbors $\mathbf{x}_k^*$ and $\mathbf{x}_l^*$
Make a gradient step to optimize $\lambda_u L\left(f\left(\mathbf{x}_k^*\right), f\left(\mathbf{x}_l^*\right), 0\right)$
until stopping criterion

below. It sums over the l labeled points and u unlabeled points. The coefficient λ_u is an empirically determined constant which controls the strength of the regularizer. The function f is a neural network classifier predicting labels $\hat{y} = f(\mathbf{x})$ using multiple layers of nonlinear transformations of the data. We use the shorthand $f_i = f(\mathbf{x}_i)$. W_{ij} are edge weights which define pairwise similarity between unlabeled examples.

A key consideration is that in the framework proposed by [6], W is restricted to be a hard neighborhood graph defined according to a k-nearest-neighbor (kNN) criterion on inputs. The neighborhood graph represents the relationship between data points where each vertex denotes a data point and edges represent the relationship between a pair of data points. That is, $W_{ij} = 1$ if j is one of i's k *nearest neighbors* and $W_{ij} = 0$ otherwise. In the following, we will revisit this restriction and later relax it such that W is a *learned* similarity graph.

The general training scheme for the SSNN is given in Algorithm 1. It makes three gradient updates per iteration. First, it attempts to minimize the labeled loss V (1 update), then the regularizer L (2 updates). The algorithm generalizes both supervised and unsupervised learning. If λ_u is set arbitrarily high, the V term is effectively ignored and the model performs unsupervised learning. If λ_u is set arbitrarily low, then the model performs supervised learning only on labeled data.

We will follow [6] and set V to be the hinge loss used in SVMs as well as its variants TSVM and LapSVM. The hinge loss has recently been shown to be more effective than negative log likelihood (cross-entropy) for supervised training of neural network classifiers [10]. We now discuss two alternatives to the regularizer, analogous to the TSVM and LapSVM.

A. LapSSNN unsupervised objective

The Laplacian SVM learns a function $f(\mathbf{x})$ which preserves neighborhood relationships among the inputs. This is equivalent to maintaining a smoothness property in feature space. Although the LapSVM is a *functional* version of the Laplacian Eigenmaps method, optimizing the Laplacian Eigenmaps loss term $W_{ij}||f(x_i) - f(x_j)||^2$ is difficult because of the hard constraints that must be imposed on the output. Therefore Ratle *et al*. adopted the Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) loss from [11], which permits an unconstrained optimization:

$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j||^2, & \text{if } W_{ij} = 1 \\ \max(0, m - ||f_i - f_j||)^2, & \text{if } W_{ij} = 0. \end{cases} \quad (1)$$

The first term in Eq. 1 ensures the smoothness property that similar points in the input space have similar outputs. The second term ensures that dissimilar points that lie within a user-specified margin m are repelled. This prevents the embedding from collapsing to a single point.

B. TSSNN unsupervised objective

Instead of enforcing smoothness on the outputs of the network with respect to the inputs, we can force neighbors to share the same class assignment. For the case of binary classification, this can be achieved through the following loss

$$L(f_i, f_j, W_{ij}) = \begin{cases} \eta^{(+)} V(\mathbf{x}_i, f_i, c), & \text{with } c = \text{sign}(f_i + f_j) \\ & \text{if } W_{ij} = 1 \\ -\eta^{(-)} V(\mathbf{x}_i, f_i, c), & \text{with } c = \text{sign}(f_j) \\ & \text{if } W_{ij} = 0 \end{cases}$$

where $\eta^{(+)}$ and $\eta^{(-)}$ are separate learning rates assigned to the neighboring pairs and non-neighboring pairs. As before, V is a hinge loss. Intuitively, we can think of c as a pseudo label which i is trained to predict. If i and j are neighbors, and they predict the same class, then i is trained to predict the same class as j. If they do not have the same class, i is trained to predict the most confident predicted label from the pair. If i and j are not neighbors, then i is trained to predict the opposite label of j. For the multi-class case, we change the hinge loss to sum over the classes

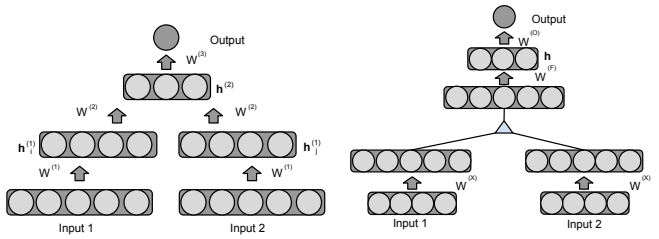
$$V(\mathbf{x}, f(\mathbf{x}), y) = \sum_{c=1}^{C} \max(0, 1 - y(c) f_c(\mathbf{x}))$$

where $y(c) = 1$ if $y = c$ and -1 otherwise.

III. Learning a Neighborhood Graph with Classifiers

The neighborhood graph represents similarity relationships between data points. Data points are represented as vertices, and the edges represent affinity between them. Selecting binary edges via kNN is restricted by the appropriateness of the distance metric (typically Euclidean). Moreover, using heuristics to set k is difficult in settings with very few labeled points. An alternative to kNN-based similarity is to use a kernel, for example radial basis functions (RBF). The RBF kernel has a single tunable parameter per dimension, the Gaussian width (σ_d), which is a simple way to control the smoothness of samples (spectra) in feature space [12]. One may try alternative kernels, but these typically must be selected and evaluated for each problem and/or dataset. Even in the case of a few labeled training points, it is possible to use this information to *learn* a more appropriate neighborhood graph.

An alternative class of methods is distance metric learning [7], [8], [11], [13], [14]. Such methods use labeled data to learn an improved distance metric which could be used to formulate the neighborhood graph for all points, including the unlabeled data. However, it has been observed that these methods tend to overfit [8], and this was our experience in preliminary experiments with both Neighborhood Components Analysis (NCA) [14] and DrLIM [11].



(a) Classical net for neighborhood graph learning (NGL)

(b) Multiplicative net for neighborhood graph learning (MNGL)

Fig. 1: Two dual-input neural net architectures for predicting whether two pixels have the same class label.

Instead, we propose to construct the neighborhood graph by using a binary classifier. The idea is simple: using only the labeled examples, we first train a classifier on pairs of pixels and attempt to predict whether they are from the same class or not. We then use the confidence output by the classifier to define the weight W_{ij} between points i and j. One advantage of our approach is that we can make better use of our sparsely labeled examples. By pairing examples, we generate $(N^2 - N)/2$ training examples to implicitly learn similarity.

We introduce two neural network architectures for constructing the dual-input classifier (Figure 1). As the networks nonlinearly transform the inputs through multiple layers of learned representation, inputs do not need to be Euclidean-similar to yield a high affinity.

The first architecture consists of two representational pathways joined by a common hidden layer. The common layer is connected to a single sigmoid output unit. Let $f^{(i)}$ be the propagation function from the i^{th} layer to the $i+1^{th}$ layer, and let $W^{(i)}$ and $b^{(i)}$ denote the weights and bias for the i^{th} hidden layer. The feed-forward process can be expressed as:

$$\begin{aligned} \hat{y}_{ij} &= f^{(3)}(\mathbf{h}^{(2)}) = \sigma(W^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}) \\ \mathbf{h}^{(2)} &= f^{(2)}(\mathbf{h}_i^{(1)}, \mathbf{h}_j^{(1)}) = \sigma(W^{(2)}\mathbf{h}_i^{(1)} + W^{(2)}\mathbf{h}_j^{(1)} + \mathbf{b}^{(2)}) \\ \mathbf{h}_i^{(1)} &= f^{(1)}(\mathbf{x}_i) = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}) \end{aligned} \quad (2)$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$, $\mathbf{x}$ is the input, and $\mathbf{h}^{(i)}$ is the hidden units of the i^{th} layer. Note that the weights $W^{(1)}$ are tied between pathways in the first and second hidden layer, and that Eq. 2 is applied for both $\mathbf{x}_i$ and $\mathbf{x}_j$.

An alternative architecture is to use a third-order model which permits multiplicative interactions between the two representational pathways. The third-order model learns relations between the transformed pixel intensities [15], whereas in the classical model the two pathways contribute additively to the penultimate layer.

The prediction function for this architecture involves a linear mapping of inputs to latent factors, a multiplicative interaction among factors, and then mapping to a set of hidden units:

$$\begin{aligned} \hat{y}_{ij} =& f^{(2)}(\mathbf{x}_i, \mathbf{x}_j) = \sigma(W^{(O)}\mathbf{h} + \mathbf{b}^{(O)}) \\ \mathbf{h} =& f^{(1)}(\mathbf{x}_i, \mathbf{x}_j) = \sigma(W^{(F)}(W^{(X)}\mathbf{x}_i) \odot (W^{(X)}\mathbf{x}_j) + \mathbf{b}_f^{(F)}) \end{aligned}$$

where $W^{(X)}$ are weights that map inputs to factors, $W^{(F)}$ and $\mathbf{b}^{(F)}$ are weights and biases that map the factors to hidden units, and $W^{(O)}$ and $\mathbf{b}^{(O)}$ are the weights and bias for the output. The symbol $\odot$ denotes element-wise multiplication.

Regardless of architecture, we use a cross-entropy loss:

$$L(\mathbf{x}_i, \mathbf{x}_j, y_{ij}) = -\frac{1}{N}\sum_{i,j} y_{ij}\log\hat{y}_{ij} + (1-y_{ij})\log(1-\hat{y}_{ij})$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the inputs, y_{ij} is the true "same class" label, and $\hat{y}_{ij}$ is the predicted label.

A. Extension to the Regularizer Term

In the original SSNN formulation described in Section II, the alternative regularization loss functions L are each piecewise, where one function is applied for $W_{ij} = 1$ and another function is applied for $W_{ij} = 0$. For each pair of pixels input, i and j, our classifier outputs a real-valued probability of "same class", which we adopt as W_{ij}. We threshold the output and proceed with a binary neighborhood graph. The Laplacian regularizer then has the form:

$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j||^2, & \text{if } W_{ij} \geq \theta \\ \max(0, m - ||f_i - f_j||)^2, & \text{if } W_{ij} < 1-\theta \end{cases}$$

and the transductive regularizer would then have the form:

$$L(f_i, f_j, W_{ij}) = \begin{cases} \eta^{(+)}V(x_i, f_i, c), & \text{with } c = \text{sign}(f_i + f_j) \\ & \text{if } W_{ij} \geq \theta \\ -\eta^{(-)}V(x_i, f_i, c), & \text{with } c = \text{sign}(f_j) \\ & \text{if } W_{ij} < 1-\theta \end{cases}$$

In our experiments, we set $\theta = 0.5$ since it empirically led to the "same-class" and "different-class" predictions in the same relative proportions to that of the training set. We admit that optimizing this parameter may lead to better results.

IV. Experiments

Our evaluation is organized into two parts. First, we assess the two proposed architectures for pairwise classification in order to verify that the constructed neighborhood graph makes sensible predictions. Second, we conduct experiments on semi-supervised neural networks using the revised regularization terms. In the latter case, we compare our model to SSNN proposed by Ratle *et al*. and demonstrate that we improve on their HSI classification performance. We also compare a composition of the SSNN with a supervised similarity learning-based method of forming the neighborhood graph, DrLIM [11]. We denote the two variants of the SSNN algorithm presented in [6] by LapSSNN and TSSNN, and denote the composition of our method of neighborhood construction and their SSNN as (Lap/T)SSNN+(NGL/MNGL). Here, NGL/MNGL stands for neighborhood graph learning using the classical and multiplicative architectures, respectively. Moreover, we include as a baseline a purely supervised neural network with two hidden layers that minimizes the hinge loss function (i.e., no regularization term) denoted SNN.

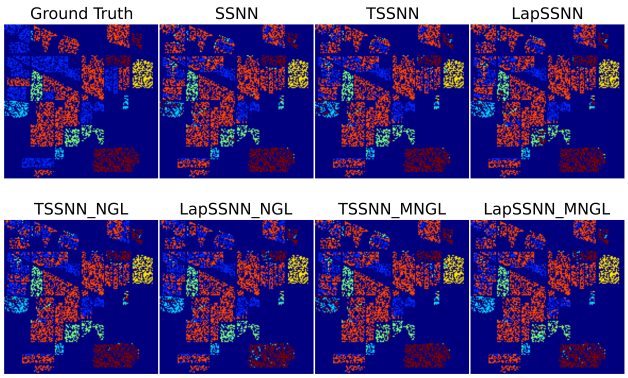


Fig. 2: The top left scene mask is the ground truth of the IP6 dataset (only showing test data points), and the next seven images are the predicted scene masks using various types of SSNNs trained with 30 labeled examples. Best viewed in color.

TABLE I: AUROC on the IP6 training and test set with respect to the number of labeled examples per class.

		5	10	15	30
Euclidean Dist	Test		50.1 ± 2.69		
DrLIM Dist	Train	97.6 ± 1.30	90.6 ± 0.90	88.7 ± 1.60	86.1 ± 1.35
	Test	77.1 ± 5.21	78.5 ± 5.39	78.9 ± 3.04	79.0 ± 3.04
NGL	Train	87.4 ± 2.95	91.0 ± 3.19	93.6 ± 2.13	97.0 ± 3.62
	Test	74.2 ± 5.40	81.1 ± 3.56	82.3 ± 3.67	84.2 ± 3.99
MNGL	Train	84.3 ± 3.17	84.3 ± 3.40	85.7 ± 2.33	87.3 ± 3.81
	Test	79.2 ± 3.85	80.0 ± 7.47	80.2 ± 2.59	80.5 ± 2.93

TABLE II: AUROC on the Pavia University training and test set with respect to the number of labeled examples per class.

		5	10	15	30
Euclidean Dist	Test		50.4 ± 4.16		
DrLIM Dist	Train	91.2 ± 3.12	90.2 ± 1.87	87.9 ± 2.38	94.6 ± 0.83
	Test	81.3 ± 4.66	82.5 ± 4.81	82.4 ± 4.45	89.7 ± 3.37
NGL	Train	97.3 ± 0.92	97.8 ± 1.17	98.5 ± 0.80	98.7 ± 0.30
	Test	83.6 ± 3.55	88.2 ± 3.09	89.5 ± 2.85	96.4 ± 1.53
MNGL	Train	96.4 ± 1.93	96.0 ± 2.01	95.0 ± 0.72	95.7 ± 1.45
	Test	86.1 ± 4.45	86.3 ± 3.69	87.5 ± 2.66	88.0 ± 2.21

A. Datasets

To assess our models, we considered two datasets that are commonly used in analyzing HSI classification algorithms: the Indian Pine Hyperspectral dataset with six classes, and the Pavia University Hyperspectral dataset with nine classes. Indian Pines consists of a single 145 × 145 pixel hyperspectral image containing 220 bands. The image is labeled with 16 classes of ground cover. The approximate length of one side of a pixel on the ground, also known as ground sample distance (GSD), is 20 m. As in other works which have considered this dataset, classes are amalgamated, reducing the total number of classes to six due to the strong mixing effects within each pixel [6]. The classes with the number of labeled pixels in parentheses are as follows: Corn, consisting of Corn, Corn-notill, and Corn-mintill (2268), Grass/Pasture (497), Grass/Trees (747), Hay-windrowed (489), Soy-beans, consisting of Soybean-notill, Soybean-mintill, and soybean-clean (4050), and Woods (1294). Figure 2 displays the ground truth labels. Bands (104 – 108), (150 – 163), and 220 are removed since they were deemed noisy and represent water absorption. We refer to this dataset as IP6.

The second dataset considered is from the University of Pavia. The data is a single 610 × 340 pixel hyperspectral image with 115 bands. The GSD is 1.3 m. The nine labeled classes are: Asphalt, Meadow, Gravel, Trees, Painted Metal Sheets, Bare Soil, Bitumen, Self-Blocking Bricks, and Shadows. We refer to this dataset as PaviaU.

For both datasets, we partitioned the data into 40% training data, 10% validation data, and 50% test data. From the training data, we randomly selected 5, 10, 15, and 30 labeled data points per class and the rest were used as unlabeled data points.

B. Evaluation of Binary Classifiers

We evaluated our two classification architectures on the IP6 and PaviaU datasets over 20 folds using the held-out test set. All hyper-parameters, such as the learning rate, strength of the regularizer, size of mini-batch, and number of hidden units, were chosen based on performance on the validation set. The learning rate and strength of the regularizer were chosen from the range [0.001, 0.01], and the number of hidden units and batch size were selected from the range [15, 120]. The purpose of this experiment is to demonstrate that the binary classifier is sound, as one may suspect severe overfitting due to the scarcity in labeled training examples. We added 15% zero-mask noise to the input to regularize the model during training. The results are shown in Tables I and II. The evaluation was assessed based on area under the ROC curve (AUROC), illustrating the performance of a binary classifier as its discrimination threshold is varied. We compared our two neural network-based binary classifiers to two baselines. The first baseline classifies pairs of points as the same class by thresholding their Euclidean distance. This method was chosen because of its similarity to the commonly used Euclidean nearest neighbor approach to forming graphs. The second baseline is similar but it uses Euclidean distance in the learned DrLIM mapping instead of Euclidean distance in the input space. The results show that both of the proposed classifiers are more effective than similarity-based methods for determining whether or not two points share the same class. This is not surprising, as they are trained discriminatively for the task. Based on AUROC score, it also indicates that, even in the case of a limited number of labeled examples per class, NGL/MNGL do not catastrophically overfit. For PaviaU, the gap between train and test is about the same for DrLIM and NGL/MNGL. For IP6 our methods overfit less than DrLIM.

C. Evaluation of Semi-supervised Neural Networks

We evaluated the TSSNN and LapSSNN using our proposed classification-based neighborhood graphs. We conducted the experiment over 10 folds, each time using only a few labeled training examples. All hyper-parameters again were chosen based on performance on the validation set. Our training pipeline was to first train the classifiers, and then apply the SSNN with a regularizer based on the graph formed by the output of the classifiers. All models used mini-batch SGD.

A comparison of our proposed architectures to the baseline SSNNs and the SSNN + DrLIM approach on IP6 is shown in Table III. On this dataset, the multiplicative architecture slightly outperformed the classical architecture for 5 and 10 labeled examples per class. On the other hand, the classical

TABLE III: Classification accuracy for the IP6 test sets.

# Labeled Examples/Class	5	10	15	30
SNN	68.2 ± 17.1	71.2 ± 11.42	74.9 ± 0.48	77.7 ± 0.85
TSSNN [6]	69.5 ± 15.7	72.0 ± 6.63	75.5 ± 0.99	77.2 ± 1.99
LapSSNN [6]	73.9 ± 10.3	74.7 ± 1.85	75.1 ± 1.17	76.4 ± 0.49
TSSNN+DrLIM [6]+[11]	68.7 ± 24.8	71.8 ± 6.07	75.5 ± 1.03	77.4 ± 2.47
LapSSNN +DrLIM [6]+[11]	73.0 ± 4.21	74.5 ± 1.56	75.5 ± 0.90	76.0 ± 2.56
TSSNN+NGL	68.3 ± 18.2	73.1 ± 3.87	76.0 ± 1.49	79.2 ± 2.2
LapSSNN+NGL	73.6 ± 3.02	75.0 ± 2.82	**77.3 ± 1.01**	**79.4 ± 1.28**
TSSNN+MNGL	69.3 ± 19.6	72.3 ± 5.47	75.7 ± 0.75	78.8 ± 2.54
LapSSNN+MNGL	**73.7 ± 3.23**	**75.3 ± 1.16**	76.3 ± 0.67	78.0 ± 2.91

TABLE IV: Classification accuracy for the PaviaU test sets.

# Labeled Examples/Class	5	10	15	30
SNN	71.8 ± 14.0	74.1 ± 6.25	76.2 ± 3.99	78.6 ± 2.32
TSSNN [6]	73.2 ± 2.79	75.3 ± 2.09	76.3 ± 5.24	78.4 ± 1.46
LapSSNN [6]	72.7 ± 2.82	74.1 ± 1.35	74.9 ± 4.07	75.8 ± 1.28
TSSNN+DrLIM [6]+[11]	72.7 ± 5.01	75.3 ± 2.48	76.2 ± 2.40	78.6 ± 1.96
LapSSNN+DrLIM [6]+[11]	72.6 ± 3.26	74.4 ± 1.94	75.4 ± 2.90	76.4 ± 1.76
TSSNN+NGL	72.5 ± 3.84	75.3 ± 1.23	77.0 ± 3.08	79.2 ± 2.38
LapSSNN+NGL	**76.8 ± 1.45**	**77.8 ± 1.84**	**79.5 ± 2.48**	**80.3 ± 0.87**
TSSNN+MNGL	76.4 ± 4.65	77.8 ± 3.15	78.7 ± 6.62	78.7 ± 1.02
LapSSNN+MNGL	73.6 ± 5.01	74.9 ± 1.38	76.3 ± 3.33	79.0 ± 4.15

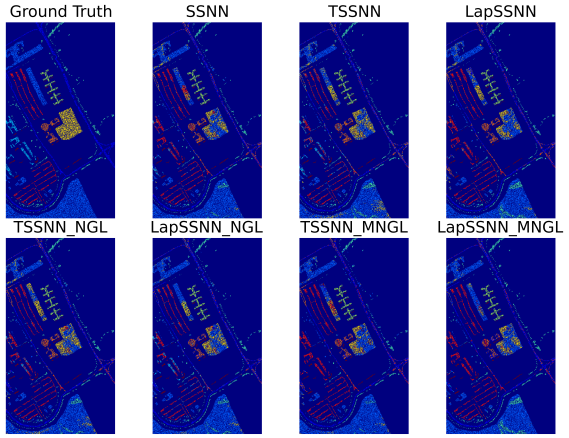


Fig. 3: The top left scene mask is the ground truth of PaviaU (only showing test data points), and the next seven images are the predicted scene masks using various types of SSNNs trained with 30 labeled examples. Best viewed in color.

architecture gave better performance when there were 15 or 30 labeled examples. Although performance of all algorithms were volatile in the extreme of only 5 labeled training examples per class, using the multiplicative architecture actually showed significantly more stable results compared to all others. Also, as we can observe from Table III, LapSSNN with NGL/MNGL outperformed TSSNN with NGL/MNGL. Figure 2 visualizes the scene mask on the IP6 dataset using each of the six models considered.

The experiments were repeated on the PaviaU dataset and results are shown in Table IV. Here, our method of LapSSNN + NGL outperformed all other methods across different numbers of labeled cases. TSSNN + MNGL also significantly outperformed the bases in the cases of 5, 10, and 15 labeled examples per class. Figure 3 visualizes the results.

On both datasets, the LapSSNN consistently outperformed the TSSNN regardless of neighborhood graph construction. This may have to do with the more explicit control of the output achieved by the Laplacian regularizer compared to the pseudo-label based transductive regularizer. Ultimately, we believe that LapSSNN with NGL is stable and offers the best performance among the other possible combinations.

NGL and MNGL are trained on a number of pairs which is quadratic in the number of labeled examples. Therefore we observed an increase of 4-6x (IP6) and 4-10x (PaviaU) in training time using these methods compared to the standard SSNN. However, none of the methods took more than 30 min to train to convergence on a GPU-equipped workstation. Using NGL and MNGL do not increase testing time.

V. Conclusion

Many semi-supervised learning algorithms assume that points that are close-by in input space share a label. This is not always true for high-dimensional, complex, noisy data, such as the HSI data we consider in this paper. Learning a representation may yield a data space suited to this assumption, and at first glance, distance metric learning techniques seem an attractive option. However, these methods fail in situations where the number of labeled training points is limited. We propose a method that can construct an improved neighborhood graph for SSL but maintains a small number of parameters. Based on its effectiveness and scalability we have chosen the semi-supervised neural network as a framework, but our technique can be applied to any SSL method that requires neighborhood graph construction.

Future work will consider learning a single rather than two-stage architecture for jointly learning the neighborhood graph and classification. We also intend to investigate unsupervised representation learning for building the neighborhood graph.

References

[1] G. Camps-Valls, D. Tuia, L. Bruzzone, and J. A. Benediktsson, "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Processing Magazine*, vol. 31, no. 10, pp. 45–54, 2014.

[2] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, p. 3, 2006.

[3] L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive SVM for semisupervised classification of remote-sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3363–3373, 2006.

[4] L. Gómez-Chova, G. Camps-Valls, J. Muñoz Marì, and J. Calpe, "Semisupervised image classification with Laplacian SVMs," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 3, pp. 336–340, 2008.

[5] G. Camps-Valls, T. V. Bandos, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 45, no. 10, pp. 3044–3054, 2007.

[6] F. Ratle, G. Camps-Valls, and J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, 2010.

[7] B. Kulis, "Metric learning: a survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.

[8] L. Yang, R. Jin, R. Sukthankar, and Y. Liu, "An efficient algorithm for local distance metric learning," in *AAAI*, 2006.

[9] G. Camps-Valls, L. Gómez-Chova, J. Muñoz Marì, J. Vila-Francs, and J. Calpe-Maravilla, "Composite kernels for hsi classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 1, pp. 93–97, 2006.

[10] Y. Tang, "Deep learning using support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[11] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reducton by learning an invariant mapping," in *CVPR*, 2006, pp. 1735–1742.

[12] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.

[13] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with app. to clustering with side-information," in *NIPS*, 2003.

[14] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood component analysis," in *NIPS*, 2004.

[15] R. Memisevic, "Gradient-based learning of higher-order image features," in *ICCV*, 2011.