Deep Semi-supervised Label Propagation for SAR Image Classification*

Joshua Enwright[a], Harris Hardiman-Mostow[a], Jeff Calder[b], and Andrea Bertozzi[a]

[a]University of California, Los Angeles, Department of Mathematics, 520 Portola Plaza, Los Angeles, CA 90095, USA
[b]University of Minnesota, School of Mathematics, 538 Vincent Hall, 206 Church Street SE, Minneapolis, MN 55455, USA

1. ABSTRACT

Automatic target recognition with synthetic aperture radar (SAR) data is a challenging problem due to the complexity of the images and the difficulty in acquiring labels. Recent work[1] used a convolutional variational autoencoder to extract relevant features prior to constructing a similarity graph in a graph-based active learning framework for SAR data. In this work we present two novel methods for classifying SAR data that use convolutional neural network (CNN) feature extraction together with techniques from graph-based semi-supervised learning in an end-to-end manner that can provide improved classification performance in the small labeled dataset regimes that are common in SAR ATR. First, we introduce Laplace Output Activation Neural Networks (LOAN Networks) as a way of directly optimizing feature embeddings for use with graph-based semi-supervised learning techniques. Next, we introduce Pseudo Label Propagation Neural Networks (PsLaPN Networks) as a inexpensive way to both boost the training signal as well as combat overconfidence and poor model calibration in neural networks. We present a novel derivation of simple formulas for the direct and efficient computation of derivatives of the outputs of graph-based algorithms like label propagation[2] for use in the training of our networks. We test the proposed end-to-end networks for active learning on OpenSARShip, a SAR dataset, where both methods surpass the previous state-of-the-art.

Keywords: Graph-based Learning, Deep Learning, Pseudolabeling, Synthetic Aperture Radar

2. INTRODUCTION

Synthetic Aperture Radar (SAR) is a useful remote sensing technique for constructing two dimensional images of three dimensional objects. Using the motion of the radar antenna over a target area, SAR can provide finer spatial resolution than ordinary radar. In addition, its day-and-night, all-weather abilities make it a promising and exciting technology. Automatic Target Recognition (ATR) seeks to classify objects in SAR images. However, hand-labeling large SAR datasets by a human annotator is prohibitively time-consuming; this makes SAR data a frequent area of research in machine learning.[1,3–8]

Supervised machine learning methods typically require large amounts of high quality, labeled training data. However, in many applications (including SAR ATR), labeling data can be expensive, time consuming, or impractical, while the passive collection of unlabeled data is only becoming easier and more common. This motivates the use of semi-supervised learning (SSL) techniques, which aim to mine data sets consisting of both labeled and unlabeled instances for additional information to supplement the labels during training. One popular collection of such techniques, which we will refer to as graph-based SSL, encode relational information among instances in the form of similarity graphs in order to supplement labels.

Graph-based SSL[2,9,10] is a class of methods for propagating labels from labeled points to unlabeled points based on the topology of a similarity graph. By supplementing label information with relational information encoded in the similarity graph, graph-based semi-supervised learning can achieve high accuracy with a fraction of the labels. Hence, constructing a high-quality similarity graph is paramount for effective

* **Source code:** https://github.com/jwcalder/SAR/tree/main/end_to_end
JE and HHM contributed equally to this work.
Send correspondence to JE: jlenwright1@math.ucla.edu

graph learning. A good similarity graph will capture similarities that are important for classification, while ignoring features that are not. In imaging, pixel values are sensitive to noise and other small corruptions. Even in the absence of corruption, naive metrics (like Euclidean or those coming from the other p-norms) on the image space don't provide meaningful measures of similarity; the classification of an image is preserved under transformations, such as translation and rotation, that can significantly alter the image pixel-wise and hence lead to similar images which are very far from one another in a given metric. Thus, since we cannot rely on constructing the similarity graph directly from pixel values, many standard preprocessing techniques have been developed for use prior to constructing the similarity graph: Scale Invariant Feature Transformation (SIFT),[11] the scattering transform,[12] pre-trained neural networks,[13] or variational autoencoders (VAEs).[7,8,14] Some research has focused on using VAEs to embed data prior to constructing similarity graphs,[10,15] while others have utilized a convolutional neural network VAE[1] (CNNVAE), producing state-of-the-art results on MSTAR,[16] another SAR dataset. However, these feature extraction techniques take place entirely before the construction of the similarity graph. While, empirically, this leads to strong results after applying graph learning, the separation of the feature extraction process and graph learning motivates a framework in which the two processes are optimized in a more synergistic way.

In this work we present three novel contributions. The first is a derivation of a formula for backpropagating Laplace learning, presented in Section 4. The second is novel framework combining CNN feature extraction with Laplace learning we call the Laplace Output Activation Neural Network (LOAN Network). The LOAN Network improves on previous work by incorporating outputs from graph learning into the training process, so that the CNN can more directly learn the best possible features for graph learning. By using Laplace learning as our output activation and tracking the necessary gradients for backpropagation (see Section 4), we can directly train the CNN to learn better features for graph learning than in previous works. Third, we use autodifferentiable label propagation earlier in a CNN in order to generate pseudo-labels for semi-supervised training in a way that is differentiable but does not make use of the potentially poorly calibrated predictions of the network itself.[17] We do this by introducing a lightweight framework called Pseudo Label Propagation Neural Network (PsLaPN Network, pronounced "Slappin' Network"). We validate our contributions by demonstrating state-of-the-art image classification results on the OpenSARShip dataset.[5]

2.1 Related Work

2.1.1 Graph-Based Semi-Supervised Learning

Many graph-based SSL models have been proposed, which compute a graph function that yields inferences of unlabeled nodes using labeled nodes and the graph topology. *Laplace learning*,[9] also known as *label propagation*, is a widely used method which computes a graph harmonic function to extend the given labels to the remainder of the graph. For large datasets with very few labels, Laplace learning degenerates and suffers from a "spiking" phenomenon[10] around labeled points, resulting in poor performance. Several methods have been proposed for these very low label rate regimes, including reweighted Laplace learning,[18,19] p-Laplace methods (for $p > 2$),[20] and Poisson learning.[10] Recent works have also focused on graph total variation problems which utilize the notion of a graph cut.[21–23] This paper will focus integrating Laplace learning with CNN feature extractions, and we leave other graph learning methodologies as interesting avenues for future work.

2.1.2 Graph Learning and Neural Networks

The use of graph-based classifiers within neural networks has been investigated previously[24] with a focus on adversarial robustness and semi-supervised deep learning, particularly in low-label rate regimes. Their proposed method is similar to our LOAN Network; they replace of the final softmax output activation function in a deep neural network with a graph-based, data-dependent activation function. However, this work only used the graph-based activation function to compute loss, and backpropagated only through linear layers in the network. In contrast, we derive formulas that allow for direct and efficient computation of the derivatives of a wide class of graph-based classification algorithms and use this to directly train our LOAN Network.

The development of LOAN Network was motivated by prior research[1] where the authors proposed an active learning pipeline for SAR data that performed classification with the use of of similarity graphs built on latent features learned from a variational autoencoder (VAE). This method for latent feature learning

has no explicit goal of learning features that are useful for graph learning, however, and so we were inspired to develop LOAN as a way of learning visual features for similarity graph-based classification in an end-to-end manner. Similarities can also be drawn between LOAN and previous work[25] on constructing similarity graphs with potentially more application-appropriate, non-Euclidean metrics. LOAN can be seen as a way of learning effective similarity kernels from data rather than hand-crafting them.

The other architecture that we propose, PsLaPN Network, was inspired by the works of Iscen *et al*[26] and Elezi *et al.*[27] Their work extends prior work on pseudo-labelling,[28] which refers to supplementing labeled training data with unlabeled instances whose labels have been predicted by some (possibly different) machine learning model. As in previous work, we use a graph-based classifier to generate pseudo-labels for deep network training.[26,27,29] Unlike these works, however, we keep soft pseudo-labels throughout, we train our model end-to-end, and we generate fresh psuedo-labels each epoch in a mini-batched fashion.

The PsLaPN Network also shares similarities with regularization techniques such as label smoothing,[30,31] which aims to combat poor model calibration in deep neural networks by penalizing overconfident predictions. This is done by averaging the usual one-hot target labels with a uniform distribution, and it has shown evidence of aiding with generalization as well as mitigating the effects of noisy labels.[31,32] The use of a uniform distribution for the smoothing might be suboptimal, however, as semantic similarity between classes is itself nonuniform. Recent work has explored smoothing with different, potentially more meaningful distributions,[33] and the unsupervised loss term in PsLaPN Network can similarly be seen as label smoothing using the predictions of graph-based classifiers.

Finally, we note that, similarly to us, prior research investigated the use of label-propagation to improve the predictions of a neural network in an end-to-end manner.[34] They only perform an approximation to label propagation, however, backpropagating through finitely many iterates of an iterative method for the computation of the solution to label propagation. They also do not construct their graphs from data, assuming instead that it is given to them as is typical of graph neural networks.

2.2 Previous Work on SAR

Owing to their broad success in many machine learning tasks, CNNs have been the focus of significant recent work on SAR ATR. CNNs are neural networks with several convolutional layers, which extract features from the data, followed by fully connected layers, which map the features to a predicted classification. The convolutional layers are translation invariant and hence well-suited to image classification problems such as ATR. The main challenge in applying deep learning to SAR is the limited amount of data. The OpenSARShip dataset, for example, has under 2300 images total and about 1000 training images, but deep learning approaches typically require much larger amounts of training data to avoid overfitting to the training set. One way to avoid this issue is to reduce the model's complexity. Because convolutional layers have significantly fewer parameters relative to the fully connected layers, researchers have proposed many alternatives to the fully connected layers to reduce the complexity of the overall architecture. Our approach replaces the fully connected layers with graph learning in testing.

Previous work used Support Vector Machines (SVMs) in place of fully connected layers to improve generalization and explainability.[35] Another work proposed the All-Convolutional Networks (A-ConvNets),[36,37] which replaced the fully connected layers with more convolutional layers to reduce the number of parameters and the risk of overfitting. Another approach used regularization to speed up convergence and prevent overfitting.[38] Other approaches have focused on feature extraction methods which are unsupervised or minimally supervised. Some authors have proposed a semi-supervised CNN method that integrated linear discriminant analysis into the training process.[39] Convolutional variational autoencoders were recently employed to extract features from SAR images in a completely unsupervised manner.[1] An autoencoder was also utilized in the so-called Euclidean Distance Restricted Autoencoder.[7] This method added a Euclidean distance-based penalty term to the reconstruction error to encourage training data from the same class to have similar feature representations. Sparse autoencoders have also been used as a preprocessing step before a single-layer CNN extracted features.[8] Additional research has focused transfer learning, which utilized pre-trained models to lessen the amount of training data needed.[40,41] Some recently proposed works have incorporated hand-crafted features as part of a deep learning framework,[3,42] leading to improved accuracy on some benchmark SAR classification tasks.

2.3 Organization of Paper

We provide mathematical background on graph-based semi-supervised learning in Section 3. We introduce and derive our method for backpropagating through Laplace learning in Section 4. We present our novel network architectures and demonstrate state-of-the-art results on OpenSARShip in Section 5. We conclude and offer future directions of work in Section 6.

3. TECHNICAL BACKGROUND

3.1 Graph-Based Semi-Supervised Learning

We consider *transductive, semi-supervised classification problems*. We restrict attention here to *binary classification* for simplicity (one can use a one-vs-all scheme for a straightforward extension to *multi-class classification*).

We suppose that we are given a data set $\mathcal{X} \coloneqq \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ consisting of points embedded in Euclidean space, with each data point belonging to exactly one of two possible classes. For $1 \leq i \leq n$, we let $y_i \in \{0, 1\}$ denote the class to which the instance $\mathbf{x}_i$ belongs and we refer to these as the *class labels* for the data set. We suppose that we are able to observe the *entire* data set $\mathcal{X}$ but that we are only able to observe a *subset* of the labels $y_1, \ldots, y_k$ for some $1 \leq k < n$. The goal of transductive semi-supervised learning is to use these observations to predict the remaining class labels $y_{k+1}, \ldots, y_n$.

Laplace Learning[9] is a classical *graph-based* method for solving such a semi-supervised learning problem. Denote by $\mathcal{L} \coloneqq \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\}$ the *labeled subset* and by $\mathcal{U} \coloneqq \{\mathbf{x}_{k+1}, \mathbf{x}_2, ..., \mathbf{x}_n\}$ the *unlabeled subset*. We begin by constructing an *affinity matrix* $W \in \mathbb{R}^{n \times n}$, which will be a nonnegative, symmetric matrix whose entry W_{ij} records the *affinity* (or *similarity*) between the instances $\mathbf{x}_i$ and $\mathbf{x}_j$. We remark that, given its nonnegativity and symmetry, such an affinity matrix can be interpreted as the adjacency matrix of a *weighted, undirected graph* whose nodes are in correspondence with the elements of the point cloud $\mathcal{X}$. A popular way to construct affinity matrices, which we will employ, is by using a Gaussian kernel

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma}\right) \tag{3.1}$$

for some *bandwidth* hyperparameter $\sigma > 0$. We then form the *degree matrix* $D \in \mathbb{R}^{n \times n}$, a diagonal matrix with diagonal entries given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$, and use this to form the the *(unnormalized) Laplacian matrix* $L = D - W$.

Laplace Learning makes predictions for the labels $y_{k+1}, \ldots, y_n$ by first obtaining the solution $\mathbf{z} \in \mathbb{R}^n$ to the problem

$$\begin{cases} (L\mathbf{z})_i = 0, & \text{if } \mathbf{x}_i \in \mathcal{U} \\ \mathbf{z}_i = y_i, & \text{if } \mathbf{x}_i \in \mathcal{L}. \end{cases} \tag{3.2}$$

We remark that the solution to (3.2) has a natural interpretation as the solution to a discrete Dirichlet problem on the graph determined by W, with the labeled instances $\mathbf{x}_i \in \mathcal{L}$ serving as boundary points. We then define

$$\hat{y}_i = \underset{y \in \{0,1\}}{\operatorname{argmin}} |y - \mathbf{z}_i| \tag{3.3}$$

for $\mathbf{x}_i \in \mathcal{U}$ (breaking ties arbitrarily) and use these for our label predictions.

4. BACKPROPAGATING THROUGH LAPLACE LEARNING

Consider, once again, binary classification with Laplace Learning with notation as above. The affinity matrix W can be partitioned as

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix},$$

i.e., with the first k rows and columns corresponding to the labeled instances and the last $n-k$ rows and columns corresponding to the unlabeled instances. Denote by

$$L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}$$

the unnormalized Laplacian obtained from W, partitioned in the same way as W. Denote by

$$\mathbf{y} \in \{0,1\}^k$$

the binary vector encoding class membership among the labeled instances. Then the solution to the Laplace Learning problem can be expressed as

$$\mathbf{z} = \begin{bmatrix} \mathbf{y} \\ -L_{uu}^{-1} L_{ul} \mathbf{y} \end{bmatrix}. \tag{4.1}$$

Consider now the use of Laplace Learning within a neural network or some other gradient-based learning context. Note that only the component

$$\mathbf{z}_u = -L_{uu}^{-1} L_{ul} \mathbf{y}$$

is fed into the loss $\mathcal{J}$, and hence the gradient of the loss with respect to $\mathbf{z}$ satisfies

$$\nabla_{\mathbf{z}} \mathcal{J} = \begin{bmatrix} \mathbf{0} \\ \nabla_{\mathbf{z}_u} \mathcal{J} \end{bmatrix}. \tag{4.2}$$

Assuming that this gradient $\nabla_{\mathbf{z}_u} \mathcal{J}$ has been obtained, we are interested in computing the gradient $\nabla_W \mathcal{J}$ of the loss with respect to the entries of the affinity matrix W. The main conceptual contribution of this work is a particularly simple formula for the exact computation of this gradient. We state this result in Lemma 4.1 and prove it in the remainder of this section.

LEMMA 4.1 (FORMULA FOR $\nabla_W \mathcal{J}$).

$$\frac{\partial \mathcal{J}}{\partial W_{ij}} = (z_j - z_i) v_i \tag{4.3}$$

To prove this, we start by viewing $\mathbf{z}_u$ as the solution to the linear system

$$L_{uu} \mathbf{z}_u = -L_{ul} \mathbf{y}. \tag{4.4}$$

Noting that the entries of the matrices L_{uu} and L_{ul} are smooth functions (in fact, they are linear) of the entries of W and that the vector $\mathbf{y}$ is constant with respect to the entries of W, we can differentiate both sides of Equation (4.4) to obtain

$$\frac{\partial L_{uu}}{\partial W_{ij}} \mathbf{z}_u + L_{uu} \frac{\partial \mathbf{z}_u}{\partial W_{ij}} = -\frac{\partial L_{ul}}{\partial W_{ij}} \mathbf{y} \tag{4.5}$$

and hence

$$\frac{\partial \mathbf{z}_u}{\partial W_{ij}} = -L_{uu}^{-1} \left(\frac{\partial L_{ul}}{\partial W_{ij}} \mathbf{y} + \frac{\partial L_{uu}}{\partial W_{ij}} \mathbf{z}_u \right) \tag{4.6}$$

$$= -L_{uu}^{-1} \frac{\partial L_u}{\partial W_{ij}} \mathbf{z}, \tag{4.7}$$

where

$$L_u = \begin{bmatrix} L_{uu} & L_{ul} \end{bmatrix} \tag{4.8}$$

denotes the last $n-k$ rows of L.

While Equation (4.7) gives an exact formula for the derivatives of $\mathbf{z}_u$ with respect to the entries of W, computing them directly in this way would, a priori, require us to solve n^2 linear systems. We can cut this down to $(n-k)^2$ solves by noting that the entries of both L_{uu} and L_{ul} are constant with respect to W_{ij} whenever $i \leq k$, but this might still be expensive. Taking our cue from the recent trend of *implicit layers* in deep learning, which include examples like deep equilibrium models[43] and Neural ODEs[44] as special cases, we find that if we are only interested in the gradient $\nabla_W \mathcal{J}$ then we can actually replace these $(n-k)^2$ solves with a single solve. Using the chain rule, we obtain

$$\frac{\partial \mathcal{J}}{\partial W_{ij}} = \frac{\partial \mathcal{J}}{\partial \mathbf{z}_u} \frac{\partial \mathbf{z}_u}{\partial W_{ij}} \tag{4.9}$$

$$= (\nabla_{\mathbf{z}_u} \mathcal{J})^T \frac{\partial \mathbf{z}_u}{\partial W_{ij}} \tag{4.10}$$

$$= (\nabla_{\mathbf{z}_u} \mathcal{J})^T \left(- L_{uu}^{-1} \frac{\partial L_u}{\partial W_{ij}} \mathbf{z} \right) \tag{4.11}$$

$$= -(L_{uu}^{-1} \cdot \nabla_{\mathbf{z}_u} \mathcal{J})^T \frac{\partial L_u}{\partial W_{ij}} \mathbf{z}, \tag{4.12}$$

with a single linear solve in the final line (4.12) that is independent of the indices (i, j).

We can simplify this formula even further by noting that

$$\frac{\partial L_u}{\partial W_{ij}} = E_{ii} - E_{ij} \tag{4.13}$$

whenever $i > k$, where $E_{st} \in \mathbb{R}^{(n-k)\times n}$ denotes the matrix with its (s,t)-entry equal to unity and all others zero. Thus,

$$\frac{\partial \mathcal{J}}{\partial W_{ij}} = \begin{cases} 0, & \text{if } i \leq k \\ (z_j - z_i)(L_{uu}^{-1} \cdot \nabla_{\mathbf{z}_u} \mathcal{J})_i, & \text{if } i > k. \end{cases} \tag{4.14}$$

Writing

$$\mathbf{v} = \begin{bmatrix} \mathbf{0} \\ L_{uu}^{-1} \cdot \nabla_{\mathbf{z}_u} \mathcal{J} \end{bmatrix}, \tag{4.15}$$

we finally obtain

$$\frac{\partial \mathcal{J}}{\partial W_{ij}} = (z_j - z_i) v_i \tag{4.16}$$

for all $1 \leq i, j \leq n$, as desired.

5. EXPERIMENTAL RESULTS

5.1 Dataset

We present state-of-the-art results on the publicly available OpenSARShip dataset[5] using our novel end-to-end framework. OpenSARShip is a popular dataset for analysis,[3,4,6,42,45,46] and hence is a good choice to measure the capability of our methodology. The dataset consists of three classes of ships (Bulk Carrier, Container Ship, and Tanker) imaged using SAR (see Table 1), and has an imbalance in its testing set, posing a challenge to many ML methods. The images are 128×128 pixels.

OpenSARShip Dataset			
Category	#Train	#Test	#Total
Bulk Carrier	338	328	666
Container Ship	338	808	1146
Tanker	338	146	484

Table 1: Number of ships in each category of the OpenSARShip dataset.

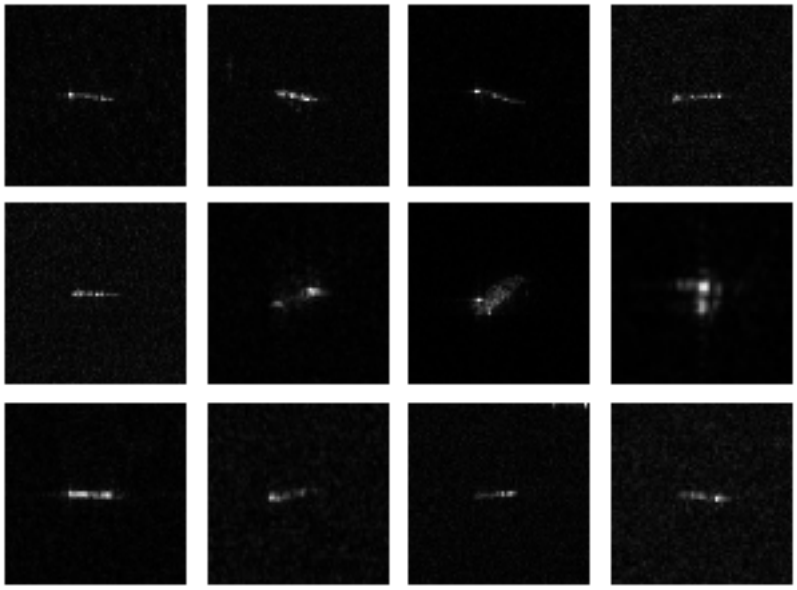

Figure 1: Example OpenSARShip images in grayscale. The top, middle, and bottom rows are Bulk Carriers, Container Ships, and Tankers, respectively.

5.2 LOAN Network Architecture

The CNN phase of our architecture is simple and standard, identical to the network presented in Miller et al.[1] It consists of two convolutional layers followed by two fully connected layers. The convolutional layers have 32 and 64 channels, respectively, with a 2×2 max pool after the first layer and 4×4 max pool after the second. To help avoid overfitting, there is a dropout[47] with $p = 0.25$ before the first fully connected layer, followed by a batch normalization,[48] and then a dropout with $p = 0.5$ in between the fully connected layers. ReLU activations are used throughout the network.

Each training minibatch consists of 26 "held in" labeled points per class, while the remaining training points are "held out." Once we embed the datapoints into the feature space using the CNN and run Laplace learning, we use the predictions based on the held in points to compute the classification loss on the held out points. In a given epoch, each minibatch uses a non-overlapping set of held in labeled points, meaning that on OpenSARShip, there are 13 batches per epoch. We also embed the unlabeled test points in order to produce a more robust graph. Note that the labels of the test points are never used in training; they are only included to ensure a more accurate approximation of the manifold structure on which graph learning operates. After the features are extracted from the CNN, we construct a fully connected graph in training, and a sparse graph using an approximate k-Nearest Neighbors search[†] with $k = 20$ in testing. Using a fully connected graph in training maintains differentiability that would be lost in the sparse case. We use angular similarity as our metric, which tends to be a beneficial choice compared to Euclidean similarity in high dimensions. From there, we construct the weight matrix as in Equation (3.1). Additionally, during testing we use self-tuning bandwidths $\sigma_i = d_k(\mathbf{x}_i)^2$, where $d_k(\mathbf{x}_i)$ is the distance from $\mathbf{x}_i$ to its k^{th} nearest neighbor. Since the k-nearest neighbor relation is not symmetric, we also symmetrize the weight matrix during testing by replacing W with $\frac{W+W^T}{2}$.

After the feature extraction and graph construction process, Laplace learning runs on the resulting graph. Laplace learning assigns a probability of each node belonging to each class. Among the labeled points that

[†]We use the python package Annoy https://github.com/spotify/annoy.

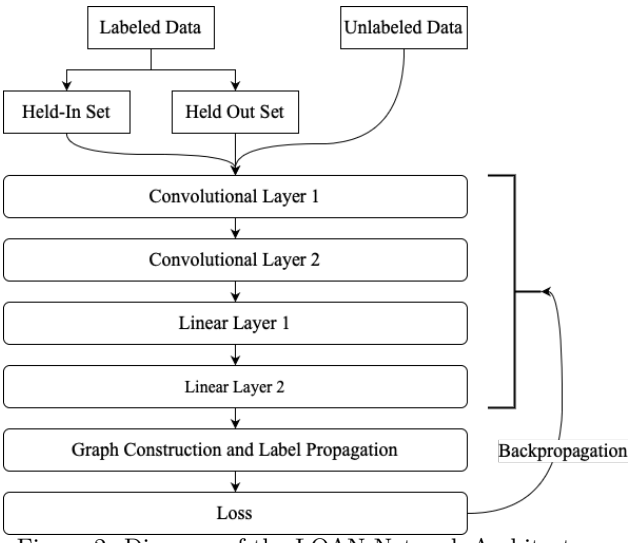

Figure 2: Diagram of the LOAN Network Architecture.

were held out, we use their known labels along with these probabilities to compute a loss. We use the standard negative log likelihood loss function. We can then use this to backpropagate (using the innovations discussed in Section 4) to update the CNN. We trained for 100 epochs, as in the work of Zhang et al,[3] which we present comparisons to shortly. We use a learning rate of 1 and the Adadelta optimizer.[49] We train on the full CNN, and perform tests by running Laplace learning on *convolutional* features, a beneficial strategy to boost performance used in a seminal work on contrastive learning.[50]

The CNN phase of LOAN was implemented using PyTorch, and the graph learning algorithms were implemented using the Graph Learning package[‡]. See Figure 2 for a visual overview of LOAN's architecture.

5.3 LOAN Results and Comparison to Prior Work

LOAN is a simple, lightweight, and computationally inexpensive learning framework compared to much of the prior art in SAR ATR, which can perform on par or exceeding the state-of-the-art depending on the use of *class priors* - a priori knowledge of the fraction of the data belonging to each class. Many graph learning methods, including Laplace learning, can be modified to automatically balance the label predictions based on prior knowledge of class sizes. Depending on the context, it may be reasonable to assume a priori knowledge of (possibly imbalanced) class sizes, so we present results with and without class priors in this section. Following previous work on graph learning,[10] we enforce the class priors by a weighting the label decision *after* solving the Laplace learning equation. To be more precise, the output of Laplace learning is a $n \times k$ matrix Z whose (i, j) entry z_{ij} can be interpreted as the probability that the i^{th} datapoint belongs to class j. The weighted label decision over k classes is given by

$$\ell_i = \underset{1 \le j \le k}{\operatorname{argmax}} \{s_j z_{ij}\},$$

where the positive weights $s_1, s_2, \ldots, s_k$ are selected with an iterative gradient descent-type approach to ensure the class priors hold (i.e., the correct number of data points are predicted in each class). There is one weight s_j for each class, and in the weighted label decision above we can see that increasing (resp. decreasing) s_j effectively increases (resp. decreases) the number of nodes predicted in class j. We refer the reader to Calder et al.[10,15] for more details on the class priors algorithm, and additional applications.

Since the test set of OpenSARShip is imbalanced, including class priors naturally improves the performance of LOAN. In addition to its high classification accuracy, LOAN enjoys significantly faster run times than previous work; the previous state of the art in OpenSARShip classification[3] reported 25.75 seconds per epoch on average, whereas LOAN only takes 4.55 seconds per epoch on average. These experiments were

[‡]See https://github.com/jwcalder/GraphLearning.

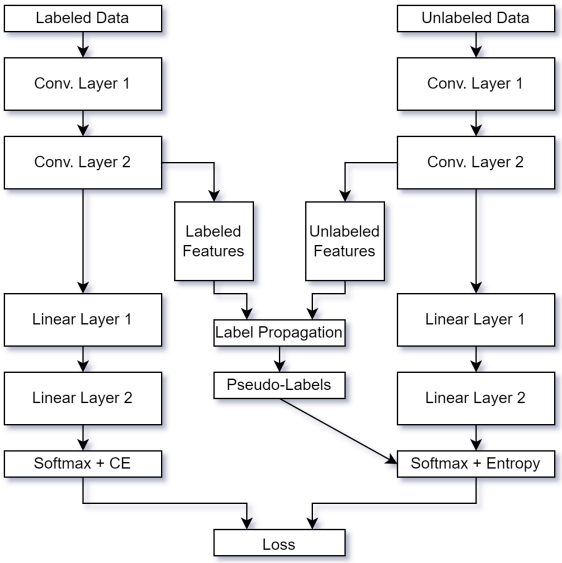

Figure 3: Illustration of the PsLaPN Network Architecture

run on a computer with AMD Ryzen Threadripper 3960X processors and an 11GB GeForce RTX 2080Ti with 4352 CUDA Cores. In particular, this GPU is very similar to the one used in the previous work.[3]

Zhang et al.[3] assessed many popular deep learning architectures, along with their own novel approach, on classifying OpenSARShip. Their framework achieved an accuracy of 78.15%±0.57%. The mean and standard deviation comes from 10 "optimal" trials, which followed the procedure of Wang et al.[45] This accuracy outperformed all previous works. For a fair comparison, we followed the same 10-trial procedure with *the same train and test split* in order to compare directly to Zhang et al.[3] and the approaches tested therein. LOAN with class priors can achieve 80.26%±0.54% accuracy (see Table 2), an over 2% absolute improvement over the previous state-of-the-art. LOAN without class priors achieves an accuracy of 77.26 ± 0.97.

OpenSARShip Classification Results (10 Trials)	
Method	Accuracy (%)
State of the Art[3]	78.15 ± 0.57
LOAN Network	77.26 ± 0.97
LOAN Network (Best Trial)	78.24
LOAN Network, CP	80.26 ± 0.54
LOAN Network, CP (Best Trial)	81.36

Table 2: Classification results on OpenSARShip, using 10 trials to calculate mean and standard deviation. Experiments run with class priors are indicated by CP.

5.4 PsLapN Network Architecture

While the PsLaPN network can be built on top of any architecture for supervised feature extraction, in this work we exclusively consider a CNN backbone consisting of two convolutional layers with max-pooling and two fully-connected layers, with ReLU activations between hidden layers and a softmax output activation. We also make use of batch normalization[48] and dropout[47] between layers, imitating the specifications of the network used in prior art.[1] In particular, we do not perform any hyperparameter tuning here.

The network is trained in mini-batches, with each mini-batch containing both labeled and unlabeled data. Each forward pass returns both the convolutional features as well as the final network predictions. Label propagation is run on the convolutional features using the given labels in order to generate pseudo labels for the unlabeled data. The loss is formed by adding two cross-entropy terms, one consisting of the network prediction distributions for the labeled data together with the one-hot label distributions and the

OpenSARShip Classification Results (10 Trials)	
Method	Accuracy (%)
HOG-ShipCLSNet[3] (Cropping)	78.15 ± 0.57
HOG-ShipCLSNet[3] (No Augs.)	76.16 ± 0.74
PsLAPN Network	78.74 ± 0.67
PsLAPN Network (Best Trial)	79.49
Backbone Network	74.53 ± 1.03

Table 3: Classification results on OpenSARShip, using 10 trials to calculate mean and standard deviation. "Backbone Network" refers to trials run using the basic CNN without use of pseudo labels. Results from Zhang et al.[3] included for comparison.

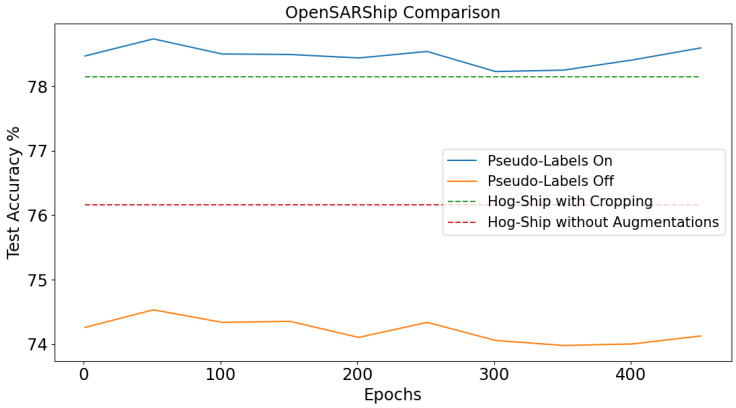

Figure 4: Comparison of PsLaPN Net to Hog-Ship on OpenSarShip.

other consisting of the network prediction distributions for the unlabeled data together with the prediction distributions obtained from label propagation. As with LOAN, we construct a dense affinity matrix with entries as in Equation (3.1), using a bandwidth parameter of $\sigma = \frac{1}{2}$, for use in generating pseudo-labels for training. An illustration of this is given in Figure 3.

5.5 PsLaPN Results and Comparison to Prior Art

As above, we compare the performance of PsLaPN Network on OpenSARShip to the previous state of the art.[3] As with LOAN, we follow the 10-trial procedure of Zhang et al[3] with the same train and test split. For these experiments, we train PsLaPN Network with a learning rate of 2 and the Adadelta optimizer. Mini-batches consist of 78 labeled instances (26 from each of the three classes) and 100 unlabeled instances. As with LOAN, we train PsLaPN Network for 100 epochs for each trial. Our results with PsLaPN Network and a comparison to those of Zhang et al[3] are presented in Table 3. For further comparison, we also report the performance of the CNN backbone of PsLaPN Network when trained with no use of psuedo-labels. To provide a comprehensive view of the training behavior of PsLaPN Network and its insensitivity to choice of training duration, we ran each of our 10-trials for 500 epochs, reporting test results every 50 epochs. Figure 4 shows these results, still providing a comparison between our results and the results reported in Zhang et al.[3]

Note that, in contrast to LOAN Network, the tests for PsLaPN are run by feeding the test data through the entire network and using the softmax predictions. In particular, we only perform label propagation during training, and hence do not use class priors while testing. Finally, we remark that PsLaPN Network is extremely lightweight, requiring only 1.4 seconds per epoch when trained on Google Colaboratory compared to 25.75 seconds per epoch in Zhang et al.[3]

6. CONCLUSION AND FUTURE WORK

In this paper we presented two novel architectures, the LOAN Network and the PsLaPN Network, both of which integrate graph learning into the training of a CNN and improve upon previous state-of-the-art SAR

image classification results. To train these networks efficiently, we derived equations for backpropagation of the loss gradients through Laplace learning.

There are many exciting future directions for this work. Our CNN was a very simple architecture: two convolutional layers and two fully connected layers. While more computationally expensive, a deeper model may improve results further. In this paper, we only considered the use of Laplace learning, but other graph learning techniques exist, such as Poisson learning,[10] which may perform better than Laplace learning at very low label rates. LOAN used a negative log likelihood loss function, which is standard in CNNs for classification problems, but a loss which enforces a smoothness constraint on the labels across the graph may be more desireable as we further explore the interplay of the CNN and graph learning. The unsupervised loss for PsLaPN network is asymmetric in the prediction distributions generated by label propagation and by the network, but this does not seem well-justified. It might be useful to consider a "symmetric cross-entropy" loss[51] and/or other methods for training neural networks on noisy labels for the unsupervised loss. It would also be worthwhile to explore ways to take into account the confidence or uncertainty of the pseudo labels that are generated.

Active learning has been shown to greatly improve results of graph-based SSL methods, especially at lower label rates,[1] and would likely be a positive addition to our methods. Since both of our methods involve training a classifier from scratch, we would need to address the "cold start" problem that we would likely encounter. For this it might be worthwhile to investigate fully unsupervised feature extraction procedures like contrastive learning[50] for use in the querying process.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Geospatial-Intelligence Agency under Award No. HM0476-21-1-0003. Approved for public release, NGA-U-2023-00758. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Geospatial-Intelligence Agency. JC was also supported by NSF-CCF MoDL+ grant 2212318.

REFERENCES

[1] Miller, K., Mauro, J., Setiadi, J., Baca, X., Shi, Z., Calder, J., and Bertozzi, A. L., "Graph-based active learning for semi-supervised classification of sar data," in [*Algorithms for Synthetic Aperture Radar Imagery XXIX*], **12095**, 126–139, SPIE (2022).

[2] Zhu, X., Ghahramani, Z., and Lafferty, J., "Semi-supervised learning using Gaussian fields and harmonic functions," in [*Proceedings of the 20th International Conference on International Conference on Machine Learning*], 912–919, AAAI Press, Washington, DC, USA (Aug. 2003).

[3] Zhang, T., Zhang, X., Ke, X., Liu, C., Xu, X., Zhan, X., Wang, C., Ahmad, I., Zhou, Y., Pan, D., et al., "Hog-shipclsnet: A novel deep learning network with hog feature fusion for sar ship classification," *IEEE Transactions on Geoscience and Remote Sensing* **60**, 1–22 (2021).

[4] Li, J., Qu, C., and Peng, S., "Ship classification for unbalanced sar dataset based on convolutional neural network," *Journal of Applied Remote Sensing* **12**(3), 035010 (2018).

[5] Huang, L., Liu, B., Li, B., Guo, W., Yu, W., Zhang, Z., and Yu, W., "Opensarship: A dataset dedicated to sentinel-1 ship interpretation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **11**(1), 195–208 (2017).

[6] Huang, Z., Datcu, M., Pan, Z., and Lei, B., "Deep sar-net: Learning objects from signals," *ISPRS Journal of Photogrammetry and Remote Sensing* **161**, 179–193 (2020).

[7] Deng, S., Du, L., Li, C., Ding, J., and Liu, H., "SAR automatic target recognition based on euclidean distance restricted autoencoder," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **10**(7), 3323–3333 (2017).

[8] Chen, S. and Wang, H., "SAR target recognition based on deep learning," in [*2014 International Conference on Data Science and Advanced Analytics (DSAA)*], 541–547, IEEE (2014).

[9] Zhu, X., Lafferty, J., and Ghahramani, Z., "Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions," in [*International Conference on Machine Learning (ICML) 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*], 58–65 (2003).

[10] Calder, J., Cook, B., Thorpe, M., and Slepčev, D., "Poisson learning: Graph-based semi-supervised learning at very low label rates," in [*Proceedings of the 37th International Conference on Machine Learning*], 1306–1316, Proceedings of Machine Learning Research (Nov. 2020). ISSN: 2640-3498.

[11] Lowe, D. G., "Object recognition from local scale-invariant features," in [*The Proceedings of the Seventh IEEE International Conference on Computer Vision*], **2**, 1150–1157, IEEE (1999).

[12] Bruna, J. and Mallat, S., "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(8), 1872–1886 (2013).

[13] Simonyan, K. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," in [*Proceedings of the International Conference on Learning Representations (ICLR)*], (2015).

[14] Mei, S., Ji, J., Geng, Y., Zhang, Z., Li, X., and Du, Q., "Unsupervised spatial–spectral feature learning by 3d convolutional autoencoder for hyperspectral classification," *IEEE Transactions on Geoscience and Remote Sensing* **57**(9), 6808–6820 (2019).

[15] Calder, J. and Ettehad, M., "Hamilton-Jacobi equations on graphs with applications to semi-supervised learning and data depth," *Journal of Machine Learning Research* **23**(318), 1–62 (2022).

[16] AFRL and DARPA, "Moving and stationary target acquisition and recognition (MSTAR) dataset." https://www.sdms.afrl.af.mil/index.php?collection=mstar. Accessed: 2021-07-10.

[17] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q., "On calibration of modern neural networks," in [*International conference on machine learning*], 1321–1330, PMLR (2017).

[18] Calder, J. and Slepčev, D., "Properly-weighted graph Laplacian for semi-supervised learning," *Applied Mathematics & Optimization* **82**, 1111–1159 (Dec. 2020).

[19] Shi, Z., Osher, S., and Zhu, W., "Weighted nonlocal laplacian on interpolation from sparse data," *Journal of Scientific Computing* **73**(2), 1164–1177 (2017).

[20] Flores, M., Calder, J., and Lerman, G., "Analysis and algorithms for Lp-based semi-supervised learning on graphs," *Applied and Computational Harmonic Analysis* **60**, 77–122 (2022).

[21] Bertozzi, A. L. and Flenner, A., "Diffuse Interface Models on Graphs for Classification of High Dimensional Data," *SIAM Review* (2016).

[22] Merkurjev, E., Bertozzi, A. L., and Chung, F., "A semi-supervised heat kernel pagerank mbo algorithm for data classification," *Communications in Mathematical Sciences* **16**(5), 1241–1265 (2018).

[23] Merkurjev, E., Bertozzi, A., Yan, X., and Lerman, K., "Modified cheeger and ratio cut methods using the ginzburg–landau functional for classification of high-dimensional data," *Inverse Problems* **33**(7), 074003 (2017).

[24] Wang, B. and Osher, S. J., "Graph interpolating activation improves both natural and robust accuracies in data-efficient deep learning," *European Journal of Applied Mathematics* **32**(3), 540–569 (2021).

[25] Kileel, J., Moscovich, A., Zelesko, N., and Singer, A., "Manifold learning with arbitrary norms," *Journal of Fourier Analysis and Applications* **27**(5), 1–56 (2021).

[26] Iscen, A., Tolias, G., Avrithis, Y., and Chum, O., "Label propagation for deep semi-supervised learning," *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* , 5065–5074 (2019).

[27] Elezi, I., Torcinovich, A., Vascon, S., and Pelillo, M., "Transductive label augmentation for improved deep network learning," in [*2018 24th International Conference on Pattern Recognition (ICPR)*], 1432–1437, IEEE (2018).

[28] Lee, D.-H. et al., "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in [*Workshop on challenges in representation learning, ICML*], **3**(2), 896 (2013).

[29] Sellars, P., Aviles-Rivero, A. I., and Schönlieb, C.-B., "Laplacenet: A hybrid energy-neural model for deep semi-supervised classification," *arXiv preprint arXiv:2106.04527* (2021).

[30] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z., "Rethinking the inception architecture for computer vision," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 2818–2826 (2016).

[31] Müller, R., Kornblith, S., and Hinton, G. E., "When does label smoothing help?," *Advances in neural information processing systems* **32** (2019).

[32] Lukasik, M., Bhojanapalli, S., Menon, A., and Kumar, S., "Does label smoothing mitigate label noise?," in [*International Conference on Machine Learning*], 6448–6458, PMLR (2020).

[33] Zhang, C.-B., Jiang, P.-T., Hou, Q., Wei, Y., Han, Q., Li, Z., and Cheng, M.-M., "Delving deep into label smoothing," *IEEE Transactions on Image Processing* **30**, 5984–5996 (2021).

[34] Klicpera, J., Bojchevski, A., and Günnemann, S., "Predict then propagate: graph neural networks meet personalized pagerank," in [*ICLR*], (2019).

[35] Wagner, S. A., "Sar atr by a combination of convolutional neural network and support vector machines," *IEEE Transactions on Aerospace and Electronic Systems* **52**(6), 2861–2872 (2016).

[36] Wang, H., Chen, S., Xu, F., and Jin, Y.-Q., "Application of deep-learning algorithms to MSTAR data," in [*2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*], 3743–3745, IEEE (2015).

[37] Chen, S., Wang, H., Xu, F., and Jin, Y.-Q., "Target classification using the deep convolutional networks for SAR images," *IEEE transactions on geoscience and remote sensing* **54**(8), 4806–4817 (2016).

[38] Wagner, S., Barth, K., and Brüggenwirth, S., "A deep learning SAR ATR system using regularization and prioritized classes," in [*2017 IEEE Radar Conference (RadarConf)*], 0772–0777, IEEE (2017).

[39] Yue, Z., Gao, F., Xiong, Q., Wang, J., Huang, T., Yang, E., and Zhou, H., "A novel semi-supervised convolutional neural network method for synthetic aperture radar image recognition," *Cognitive Computation* **13**(4), 795–806 (2021).

[40] Huang, Z., Pan, Z., and Lei, B., "Transfer learning with deep convolutional neural network for sar target classification with limited labeled data," *Remote Sensing* **9**(9), 907 (2017).

[41] Wang, Y., Wang, C., and Zhang, H., "Ship classification in high-resolution sar images using deep learning of small datasets," *Sensors* **18**(9), 2929 (2018).

[42] Zhang, T. and Zhang, X., "A polarization fusion network with geometric feature embedding for sar ship classification," *Pattern Recognition* **123**, 108365 (2022).

[43] Bai, S., Kolter, J. Z., and Koltun, V., "Deep equilibrium models," *Advances in Neural Information Processing Systems* **32** (2019).

[44] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K., "Neural ordinary differential equations," *Advances in neural information processing systems* **31** (2018).

[45] Wang, C., Shi, J., Zhou, Y., Yang, X., Zhou, Z., Wei, S., and Zhang, X., "Semisupervised learning-based sar atr via self-consistent augmentation," *IEEE Transactions on Geoscience and Remote Sensing* **59**(6), 4862–4873 (2020).

[46] Huang, Z., Pan, Z., and Lei, B., "What, where, and how to transfer in sar target recognition based on deep cnns," *IEEE Transactions on Geoscience and Remote Sensing* **58**(4), 2324–2336 (2019).

[47] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research* **15**(1), 1929–1958 (2014).

[48] Ioffe, S. and Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in [*International conference on machine learning*], 448–456, PMLR (2015).

[49] Zeiler, M. D., "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701* (2012).

[50] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G., "A simple framework for contrastive learning of visual representations," in [*Proceedings of the 37th International Conference on Machine Learning*], III, H. D. and Singh, A., eds., *Proceedings of Machine Learning Research* **119**, 1597–1607, PMLR (13–18 Jul 2020).

[51] Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J., "Symmetric cross entropy for robust learning with noisy labels," *CoRR* **abs/1908.06112** (2019).