UCLA CAM REU 2021 Active Learning Project

Xoaquin Baca[a], Jack Mauro[b], Jason Setiadi[c], Zhan Shi[d], Kevin Miller[d], Jeff Calder[e], and Andrea Bertozzi[d,f]

[a]Harvey Mudd College, Department of Computer Science, 201 Platt Blvd, Claremont, CA 91711, USA
[b]Loyola Marymount University, Department of Mathematics, 1 LMU Drive, Los Angeles, CA 90045, USA
[c]University of Minnesota, School of Statistics, 313 Ford Hall, 224 Church Street SE, Minneapolis, MN 55455, USA
[d]University of California, Los Angeles, Department of Mathematics, 520 Portola Plaza, Los Angeles, CA 90095, USA
[e]University of Minnesota, School of Mathematics, 538 Vincent Hall, 206 Church Street SE, Minneapolis, MN 55455, USA
[f]University of California, Los Angeles, Department of Mathematics and Mechanical and Aerospace Engineering, 420 Westwood Plaza, Los Angeles, CA 90095, USA

ABSTRACT

This paper encompasses the span of our research in the UCLA CAM REU Active Learning group, from our first musings with graph construction from a dataset to using graph-based learning in a novel approach to improve classification accuracy of the MSTAR dataset. We begin by going over necessary techniques and definitions for graph learning and active learning, using examples of simpler datasets. We also provide an overview of neural networks and discuss how they can improve the accuracy of graph learning when combined with it. We then introduce and explain the MSTAR dataset. Finally, we present our unique method for high classification accuracy of MSTAR data, through the use of neural networks for feature extraction which are then used to construct a graph by means of graph-based semi-supervised learning and active learning. With this method in mind, we utilize two types of neural networks, a convolutional neural network and a convolutional variational auto-encoder, that both provide competitive classification accuracies on the MSTAR dataset. We demonstrate that these methods, particularly the convolutional variational auto-encoder, are very useful in low-label rate SAR classification problems.

Keywords: Graph learning, active learning, semi-supervised learning, synthetic aperture radar (SAR), MSTAR, target recognition, deep learning, autoencoder

1. INTRODUCTION

Machine learning (ML) develops algorithms for computers to perform specific tasks without being explicitly programmed to do so. Some approaches are supervised learning, e.g. convolutional neural network (CNN) and support vector machine (SVM), where a large amount of training data is used; unsupervised learning, e.g. spectral clustering, k-means clustering and t-distributed stochastic neighbor embedding (t-SNE), where no training data is used; and semi-supervised learning (SSL), e.g. Laplace learning, Poisson learning and graph MBO scheme, where training data is used during the learning process but much less than in supervised learning. Section 2 provides an overview of common Graph Based Semi-Supervised learning (GSSL) methods.

Active learning is a sub field of machine learning that allows the learning algorithm to choose a subset of data to label. The key idea behind this is, rather than choosing the data at random, there are many sampling strategies that can be used to get the most informative data that yields greater accuracy. In other words, active learning seeks to find the most advantageous data in a dataset to label. This queried data is then sent to a human oracle to be labeled. One method is to label the data that the machine learning model is least certain of

how to label, which is called uncertainty sampling. It was shown in Ref. 1 that uncertainty sampling provides higher accuracy than random sampling in various amounts of data being sampled on a text classification problem. Uncertainty sampling, however, is known to be sensitive to outliers and redundancy, so section 3 provides an overview of other various active learning methods.

Section 4 explores applications of GSSL methods to image segmentation using the non-local means method. Section 5 provides an overview of other machine learning algorithms that can be used in conjunction with graph based learning. The synthetic aperture radar (SAR) MSTAR dataset is introduced in section 6. Experimental results with graph learning on MSTAR are presented in section 7 alongside a comparison with classification results from a Support Vector Machine. Section 8 provides an overview of the experimental results obtained by using the active learning methods from section 3 with Laplace Learning on MSTAR.

2. GRAPH-BASED LEARNING

2.1 Graph Definitions and Notation

Let $Z = \{z_1, z_2, \cdots, z_n\}$ be the dataset with each data point being a node in a graph. A weight function $w : Z \times Z \to \mathbb{R}$ is defined to characterize how similar each pair of data points are, with larger w means greater similarity. We assume the the graph is unidirectional, i.e. $w(z_i, z_j) = w(z_j, z_i)$. An example is

$$w(z_i, z_j) = \exp\left(\frac{d(z_i, z_j)^2}{\sqrt{\tau(z_i)\tau(z_j)}}\right), \tag{1}$$

where $d : Z \times Z \to [0, +\infty)$ is the metric and $\tau : Z \to (0, +\infty)$ is the local scaling. Here w has range $(0, 1]$. Possible choices of d include the Euclidean metric and, in the case of vector-valued data, the angular metric. For τ, one choice is to set it as a constant, which will make Eq. 1 the Gaussian function. Defining a constant τ does not add to the computational cost, but may affect the accuracy of the whole algorithm if there are multiple scales present in the dataset. In this case, we may define $\tau(z_i) = d(z_i, z_{M_i})$ where z_{i_k} is the k-th nearest neighbor of z_i. This idea was introduced in the Zelnik-Manor and Perona paper 2. We abbreviate this definition of the weight function as ZMP and let $k = 10$ unless otherwise specified. w induces a weight matrix W with entries $W_{ij} = w(z_i, z_j)$. W is symmetric due to the symmetry of w. Furthermore, we define the degree of each node z_i as $d_i = \sum_{j=1}^{n} w(z_i, z_j)$ and the degree matrix $D = \text{diag}(d_1, d_2, \cdots, d_n)$. The degree of a node can be interpreted as characterizing how strongly it is connected to the rest of the dataset.

Now we can define the unnormalized graph Laplacian, also know as the combinatorial Laplacian, as $L = D - W$. A common normalization is $L_s = D^{-1/2}LD^{-1/2}$, called the symmetric Laplacian. It can be shown that both the unnormalized and the symmetric Laplacians are positive semidefinite, with the smallest eigenvalue being zero. For this reason, the first eigenpair is called trivial. Conventionally, the eigenvalues of the graph Laplacian are listed in an increasing order, i.e. $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ with λ_i being the i-th eigenvalue of the graph Laplacian with corresponding eigenvector v_i.

2.2 Spectral Clustering

The graph Laplacian can be directly applied to binary classification using its second eigenvector, or the first non-trivial one, called the Fiedler vector. Figure 1 shows that spectral clustering using both the unnormalized and the symmetric Laplacians gives near-perfect classification of the two moons dataset with 1,000 data points and noise level of 0.12. As mentioned in Sec. 1, spectral clustering does not rely on any known labels for the classification, and thus belongs to unsupervised learning.

2.3 Accelerating the Diagonalization of the Graph Laplacian

Solving for the eigenpairs of the graph Laplacian can become extremely inefficient when we use the weight function as in Eq. (1) for a large dataset, because the resulting weight matrix and graph Laplacian is dense. 2.3.2 and 2.3.1 introduce two ways of accelerating the eigen-computation.

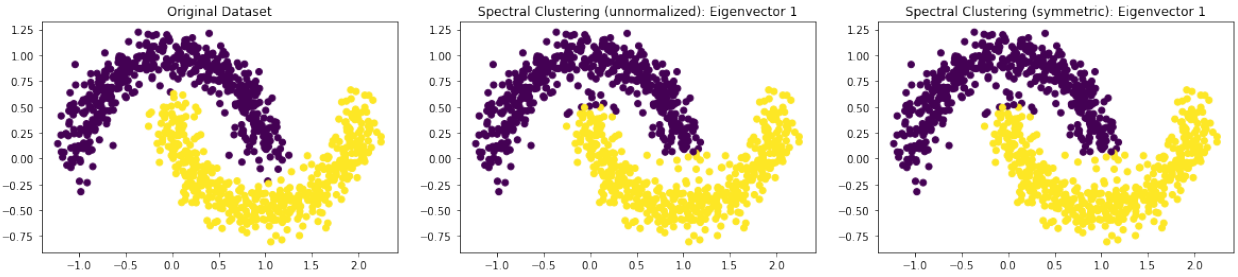

Figure 1. Two moons dataset with (left) true labels, (middle) labels assigned using the unnormalized Laplacian and (right) labels assigned using the symmetric Laplacian.

2.3.1 Nyström Extension

Nyström extension is a low rank approximation of a dense weight matrix. First we partition Z randomly into $X \subset Z$ and $Y = Z \setminus X$ with $k := |X| \ll |Z| =: n$ and relabel if necessary. We can then let W_{XY} denote the block of the weight matrix with entries $w(x, y)$ with $x \in X, y \in Y$ and similar for W_{XX}, WYX and W_{YY}, i.e.

$$W = \begin{bmatrix} W_{XX} & W_{XY} \\ W_{YX} & W_{YY} \end{bmatrix}.$$

By symmetry, we have $W_{XY} = W_{YX}^T$. It can be shown that W_{YY} may be approximated by $W_{YY} = W_{YX} W_{XX}^{-1} W_{XY}$, which means we only have to compute and store W_{XX} and W_{XY} (or W_{YX}), i.e. k rows (or columns) to approximate W. Ref. 3 has an algorithm for approximating the first k eigenpairs of L_s with complexity $O(k)$, as compared to $O(n^2)$ for a typical eigensolver for a dense matrix. This algorithm does not construct L_s explicitly, which further saves computational and storage costs.

The choice of k can have a significant impact on the running time and accuracy of the learning algorithms, and we can exemplify this with the MSTAR dataset (introduced in Sec. 6) using Nyström extension and multiclass MBO scheme (introduced in 2.4.3). Here k is also equal to the number of eigenpairs that Nyström extension outputs. Figure 2 shows how the accuracy of the learning algorithm changes with k. For $50 \leq k \leq 350$, the accuracy improves significantly with k, as more information is accounted for with more sampling points. The accuracy does not have a significant difference within $400 \leq k \leq 650$, and beyond that the accuracy begins to drop, probably due to the fact that very similar data points have very similar weights with other nodes and this causes singularity when inverting W_{XX}, and sampling more points has a higher likelihood of causing redundancy. The running time of both the Nyström extension and the MBO iterations are $O(k)$ as seen in Figure 3, but the former takes the majority of the time. Thus, $k = 400$ is the most reasonable balance of accuracy and running time, while we can also pick $k = 650$ for even higher accuracy with the expense of more running time.

2.3.2 k Nearest Neighbor Graph

Another idea for easing the computational burden is to set up a dense weight matrix in the first place. This can be achieved by only keeping the k largest weights associated with each node, which gives the resulting graph its name: k nearest neighbor (kNN) graph. The resulting weight matrix and graph Laplacian will only have k non-zero entries in each row, and typically $k \ll n$.

The kNN graph has the potential risk of being disconnected, i.e. there does not exist a path between some pair of node. Some algorithms may require the graph to be connected, consequently requiring a larger k. Also, kNN search can generate a very high computational cost, so fast, approximate search algorithms can be preferred. The high computational cost does not disadvantage the kNN graph because, even in a computation for a dense weight matrix using Eq. (1), kNN search is also often used to find better scale(s).

2.4 Graph-Based SSL and PDE-Based Methods

Graph-based SSL (GSSL) uses the graph structure in addition to a small amount of training data as is in SSL in general. Our goal is to find a function $u : Z \to \mathbb{R}^m$ for label assignment, where m is the number of classes

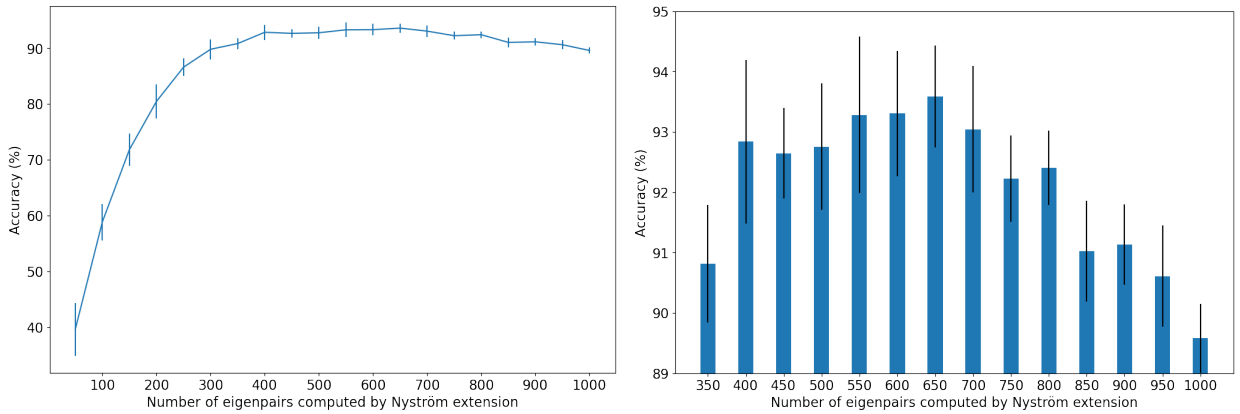

Figure 2. Accuracy of Nyström extension against the number of eigenpairs it outputs. The training labels are fixed with 100 per class, and the step size is 50. At each step, 10 trials are run to account for the randomness in the choice of X in Nyström extension.

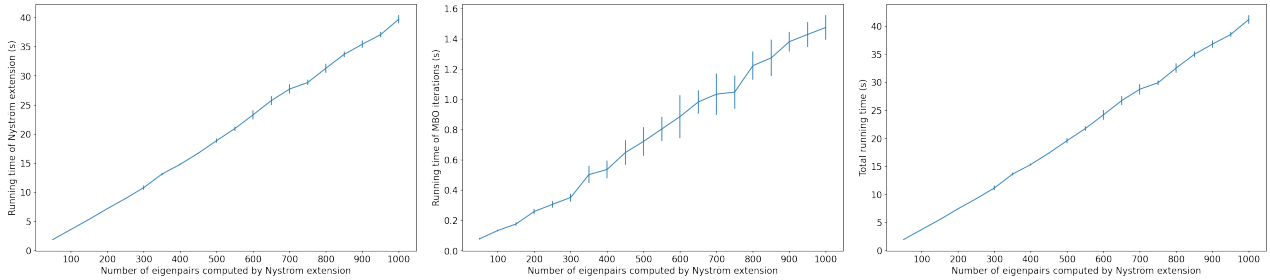

Figure 3. Running time of (left) Nyström extension, (middle) multiclass MBO iterations, and (right) total running time against the number of eigenpairs Nyström extension outputs.

we assign the data into, with the labels corresponding to the one-hot vectors $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m \in \mathbb{R}^m$. The training data $z_1, z_2, \cdots, z_l \in Z$ have labels $y_1, y_2, \cdots, y_l \in \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m\}$. u must thus satisfy

$$u(z_i) = y_i, \quad i = 1, 2, \cdots, l. \tag{2}$$

Intuitively, u should map similar data (i.e. data points with large among each other) to similar labels. Two optimization problems are the most popular: we minimize either the Dirichlet energy

$$E(u) := \frac{1}{2} \sum_{i,j} w_{ij} |u(z_i) - u(z_j)|^2 \tag{3}$$

or the total variation (TV) semi-norm

$$\|u\|_{TV} := \frac{1}{2} \sum_{i,j} w_{ij} |u(z_i) - u(z_j)|, \tag{4}$$

which lead to various PDE-based methods.

2.4.1 Laplace and Poisson Learning

A key observation for Eq. (3) is that $E(u) = u^T L u$, i.e. the unnormalized Laplacian is the first variation (i.e. first derivative) of $E(u)$. Thus, the minimizer of Eq. (3) satisfies $Lu = 0$ on the unlabeled data, i.e. the discrete version of the Laplace equation, which is the motivation of the name "Laplace learning". Some famous properties

of the solution of the Laplace equation in Euclidean space also extends to the discrete case: u is harmonic, and the mean value property gives

$$u(z_j) = \frac{1}{d_j} \sum_{i=1}^{n} w_{ij} u(z_i), \quad j = l+1, l+2, \cdots, n.$$

Also, the maximum principle guarantees the uniqueness of u. Further details of Laplace learning can be found in Ref. 4. A recent method that is related to Laplace learning is Poisson learning. Instead of setting Eq. (2) as a constraint, Poisson learning incorporates it into the source term of the Poisson equation. Further details of this method can be found in Ref. 5.

Compared to Laplace learning, Poisson learning has the advantage of achieving high accuracy at very low label rate. This can be exemplified by the experiments on the MNIST dataset. The MNIST dataset consists of 70,000 images of handwritten digits for classification. From Figure 4 we can see that even with only one label per class, Poisson learning produces an accuracy higher than 90%, while Laplace learning has only around 15% accuracy. The accuracy of both methods increase with the number of training data, and ultimately both methods can reach an accuracy of around 98%.

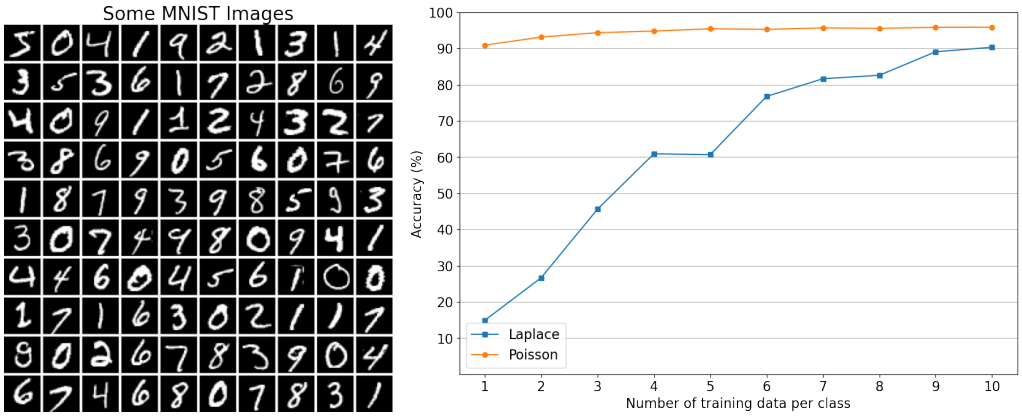

Figure 4. (Left) Some images in the MNIST dataset. (Right) Accuracy of Laplace and Poisson learning using kNN weight matrix against the number of training data per class. For each number of training data per class, 10 trials are run to account for the randomness in the choice of training labels.

2.4.2 Ginzburg-Landau Functional and Convex Splitting Scheme

In the case of binary classification (i.e. scalar-valued u), Ref. 6 has shown that the Ginzburg-Landau (GL) functional

$$GL(u) = \frac{\epsilon}{2} \langle u, L_s u \rangle + \frac{1}{4\epsilon} \sum_{i=1}^{n} \left((u(z_i))^2 - 1 \right)^2 \tag{5}$$

Γ-converges to Eq. 4. The GL functional is relatively easier to minimize, and Ref. 3 introduces a convex splitting scheme for that. Even though the *Gamma*-convergence does not easily extend to vector-valued u, a multiclass GL scheme extending that in Ref. 3 is formulated in Ref. 7.

2.4.3 Merriman-Bence-Osher Schemes

The Merriman-Bence-Osher (MBO) scheme was proposed in Ref. 8 to approximate motion by mean curvature by alternating between diffusion and thresholding steps. Ref. 7 formulates a multiclass MBO scheme for graph learning as follows:

1. *Diffusion.* Heat equation with forcing term:

$$\frac{u^{p+1/2} - u^n}{dt} = -L_s u^p - \mu(u^p - \hat{u}),$$

where $\mu(u^p - \hat{u})$ is the fidelity term to prevent labeled data from being pushed away from their true labels.

2. *Thresholding.* Projecting u to the vertices of the simplex:

$$u^{p+1}(z_i) = \mathbf{e}_{k_i}, \quad i = 1, 2, \cdots, n.$$

Ref. 9 introduces modularity MBO, which still has the diffusion and thresholding steps but functions more like a clustering algorithm, i.e. in theory it can produce reasonable accuracy without any training data, but using training data can boost the performance, as can be seen in Sec. 7.1.

3. ACTIVE LEARNING

Most active learning techniques can be categorized as either more explorative or more exploitative methods. Explorative methods seek to label points from a wide variety of clusters across the dataset whereas exploitative methods focus in on the ambiguous points in the dataset. We also want to introduce a synthetic dataset we used which is the 8 Gaussian Clusters Dataset.

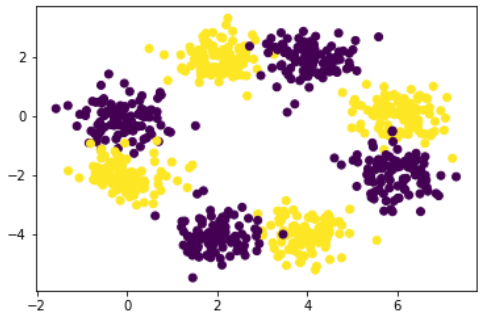

Figure 5. The 8 Gaussian Clusters Synthetic Dataset.

In this dataset, we are trying to visualize the active learning methods we want to perform before applying them to more complex datasets. This dataset has 800 points with 2 features and only has 2 distinct classes. The classifier we use for this experiment is Laplace Learning (Ref. 4). Below are the list of Active learning methods we performed in this dataset as well as the results we get.

3.1 Uncertainty Sampling

Uncertainty Sampling is one of the simplest exploitative methods in active learning. It seeks for the most ambiguous point defined by the classifier. As a result, it tends to choose outliers and repetitive points to label. $\hat{u}_k$ is defined as the output of our classifier in a form of a score matrix, which is a matrix of size number of points by number of classes with entries ranging from 0 to 1. The threshold we use in our experiments is simply the one hot vector of the predicted labels. The objective function in Uncertainty Sampling is given by

$$\hat{s}_k = softmax(\hat{u}_k)$$

$$k = \underset{k \in u}{\operatorname{argmax}} \|\hat{s}_k - thresh(\hat{s}_k)\|_2 \tag{6}$$

where k represents $k \in u$ that is the point queried to be labeled. Figure 6 shows the visualization of how Uncertainty Sampling selects points to label in our synthetic dataset.

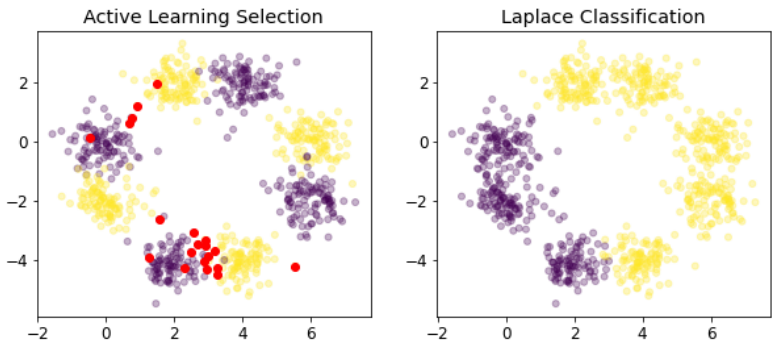

Figure 6. Uncertainty Sampling in the 8 Gaussian Clusters Dataset

3.2 Variance Optimality

Variance Optimality (V-Opt) is one of the most well-known explorative active learning methods. It tries to minimize the total variance of the distribution on the unlabeled data, as well as the expected prediction error (Ref. 10). C is defined as the covariance matrix of the unlabeled nodes where $C = L_{uu}^{-1}$. u is defined as the set of unlabeled indices. The objective function in Variance Optimality is given by

$$k = \underset{v \in u}{\operatorname{argmax}} \frac{\sum_{t \in u} (C_{vt})^2}{C_{vv}} \tag{7}$$

The key feature that we see from V-Opt in our experiments is that it tends to reach a high accuracy much quicker due to the exploration it performs. Figure 7 shows the visualization of how V-Opt selects points to label in our synthetic dataset.

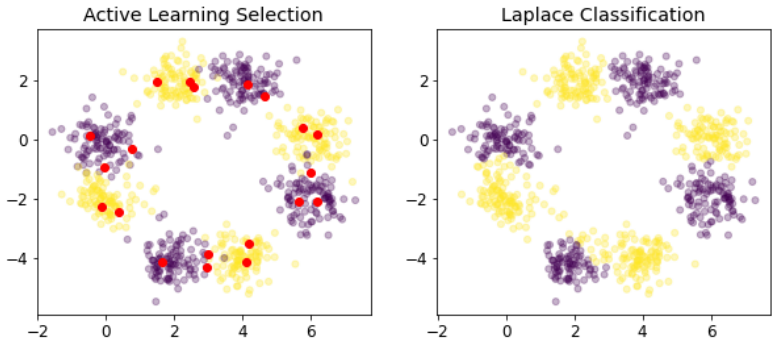

Figure 7. Variance Optimality in the 8 Gaussian Clusters Dataset

3.3 Σ Optimality

Σ Optimality (Σ-Opt) is an explorative active learning method introduced in Ref. 11. Σ Optimality tries to minimize the sum of the elements in the predictive covariance matrix C. The objective function in Σ Optimality is given by

$$k = \underset{v \in u}{\operatorname{argmax}} \frac{(\sum_{t \in u} C_{vt})^2}{C_{vv}} \tag{8}$$

Σ Optimality has a similar feature with V-Optimality in terms of exploring the structure of the dataset. The main difference is that V-Opt squares the entries in the C matrix first before summing whereas Σ-Opt does the opposite. Figure 8 shows the visualization of how Σ-Opt selects points to label in our synthetic dataset.

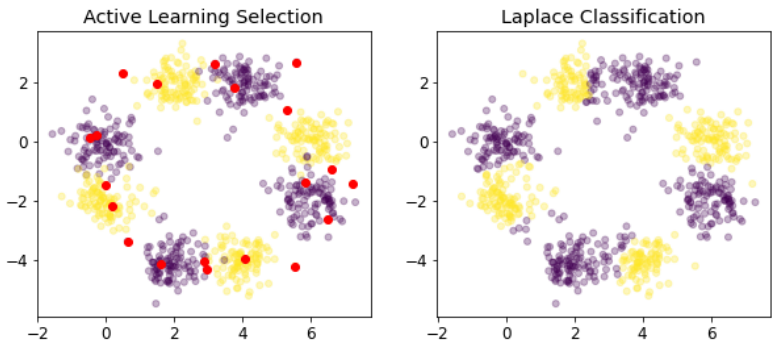

Figure 8. Σ Optimality in the 8 Gaussian Clusters Dataset

3.4 Model Change

Model Change is an active learning method introduced in Ref. 12 that contains both explorative and exploitative components. Model change queries the points that produce the greatest change in the model (see what this means). The model change objective function is given by

$$k = \underset{v \in u}{\operatorname{argmax}} \frac{\sqrt{\sum_{t \in u} (C_{vt})^2}}{C_{vv}} * \|\hat{s}_k - thresh(\hat{s}_k)\|_2 \tag{9}$$

The $\frac{\sqrt{\sum_{t \in u} (C_{vt})^2}}{C_{vv}}$ term is similar to 7, but the numerator is square rooted. This is the component of model change that explores various clusters in the dataset. The $|\hat{s}_k - thresh(\hat{s}_k)\|_2$ term is taken from 6. This causes model change to balance both the exploration of new clusters as well as find the more ambiguous points in the dataset. Unfortunately, we didn't get satisfactory results from performing Model Change on our synthetic dataset. We suspect that the cause of this problem is the presence of negative entries in our C matrix. Thus, we omit it from our results. Nevertheless, Model Change is a promising method as shown in Ref. 12.

3.5 Model Change Variance Optimiality

Model Change Variance Optimization (MC V-Opt) is a combination of variance optimization and model change. The objective function is given by:

$$k = \underset{v \in u}{\operatorname{argmax}} \frac{\sum_{t \in u} (C_{vt})^2}{C_{vv}} * \|\hat{s}_k - thresh(\hat{s}_k)\|_2 \tag{10}$$

The only difference between model change and MC V-opt is that the explorative term $\frac{\sum_{t \in u} (C_{vt})^2}{C_{vv}}$ is taken directly from 7 so the numerator is not square rooted. Figure 9 shows the visualization of how Model Change-VOpt selects points to label in our synthetic dataset.

3.6 Active Learning Results on Synthetic Dataset

Figure 10 shows average result we get from performing 10 trials of active learning on the synthetic dataset. The accuracy scores are obtained from performing Laplace Learning on every iteration of the active learning selection. We select 1 point per class randomly as our initial labeled set. Then we perform active learning to select the next points up to a total of 20 labeled points. We also perform random sampling as a baseline to compare our active learning methods to.

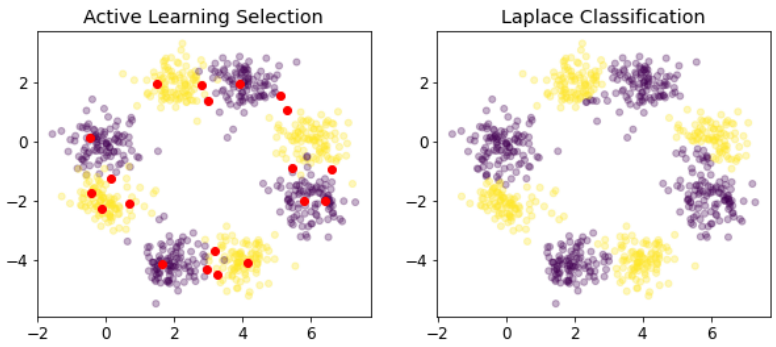

Figure 9. Model Change-Variance Optimality in the 8 Gaussian Clusters Dataset

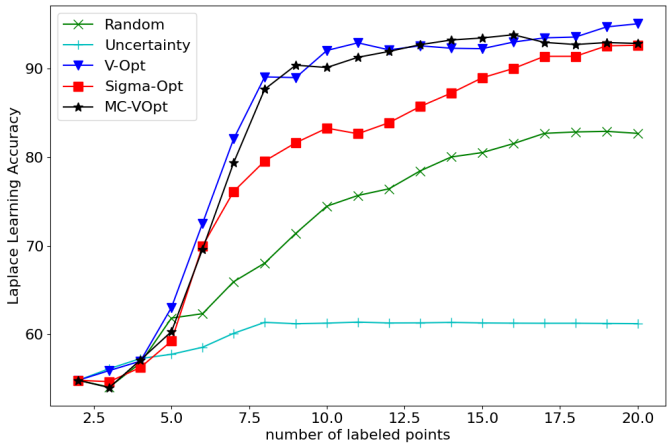

Figure 10. Average 10 Trials of Active Learning on Synthetic Dataset Results

4. NON-LOCAL MEANS METHOD FOR IMAGE SEGMENTATION

In this section, we want to introduce the Non-local Means method. It is an image pre-processing method commonly used for image segmentation problems. The idea behind this method is to form a square patch for each pixel in an image and transform each patch to a vector to help the classifier classify each pixel better. The algorithm is explained in Ref. 13 in the Algorithm 4 section. Figure 11 also shows how this method works.

As an introductory experiment to implement this method, we used the two cows RGB image shown in Figure 12. We pre-processed it using non-local means and tried to segment it using spectral clustering as in Ref. 14. For the non-local means, we used window size 3 which creates a 7x7 patch for each pixel in the image. Then we performed spectral clustering, which is an unsupervised learning method, to classify each pixel and perform the image segmentation. Figure 13 shows the segmentation result from performing the methods above.

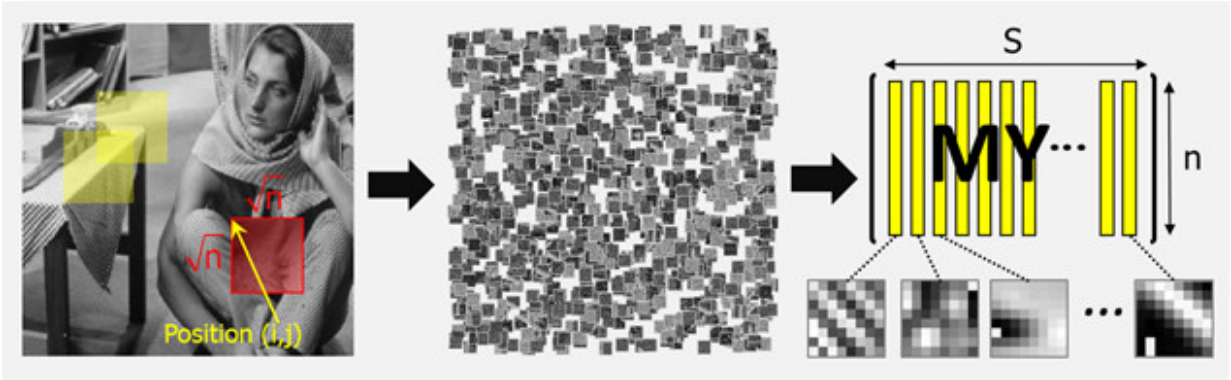

Figure 11. Non-local Means Method

Figure 12. Two Cows RGB image

Figure 13. Image Segmentation Result using Non-local Means and Spectral Clustering

5. COMBINING GRAPH LEARNING WITH NEURAL NETWORKS

Throughout our studies we were constantly searching for methods that could increase classification accuracy when paired with graph learning. One method in particular which we ended up spending a lot of time working with is the addition of neural networks. These models have led to incredible strides in the field of deep learning in recent years, in fields such as image recognition and segmentation, as well speech recognition. In less rigorous terms, they are essentially an ordered combination of functions that take in data and transform it some number of times, so that implicit patterns in the data are emphasized and the data is classified. Shown in Figure 14 is the most basic structure all neural networks have. Each network is made up of layers of nodes, or "neurons",

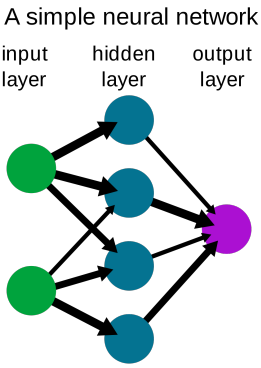

Figure 14. The fundamental structure behind any neural network.

as they are called, which perform calculations from an input and output a different number. The input in the first layer (input layer) is just the values from the raw data, and neurons in the input layer then output their results to neurons in the hidden layers. The output layer of a neural network produces a result that represents the classification of what the network predicts the input to be. These networks alone, without any training, can lead to improvements in graph learning accuracy, but the real gains occur after the network is trained on a specific dataset. Once this happens, networks adjust the weight and bias of the layers through gradient descent and backpropagation, and they are able to better classify the data.

5.1 Variational Autoencoder

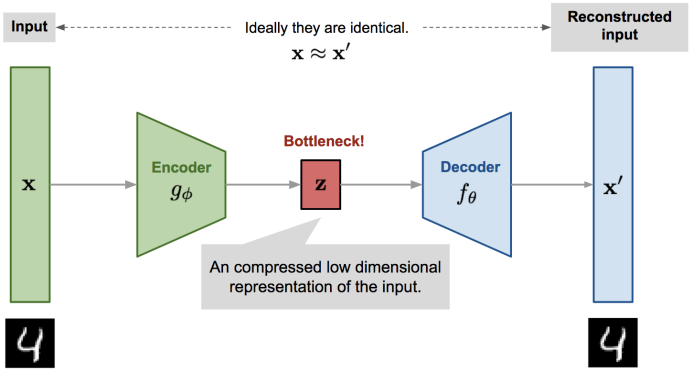

Figure 15. The architecture of a standard VAE.

The first type of network we explored is known as a Variational Autoencoder (VAE). An example is shown in Figure 15. These neural networks are very interesting primarily because they don't require labeled data to be trained. What a VAE does is take an input (usually an image), compress it into a small-dimensional form known as its latent-space, and from that try to reconstruct it back up to its original dimensionality. It then compares the original version and the reconstruction, and calculates the loss based on the difference between them. Thus when we combine it with graph learning we only need the small percentage of the labeled data necessary for semi-supervised Laplace learning.

5.2 Convolutional Neural Network

The other main type of network we worked with is the Convolutional Neural Network (CNN). A relatively simple structure is presented in Figure 16. These models contain the term "convolutional" because they include layers that employ a convolution operation as a way to transform data features. This is done by what is known usually

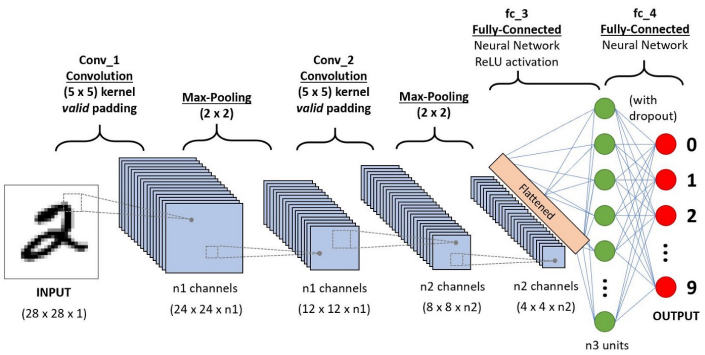

Figure 16. The architecture of a simple CNN.

as a convolution kernel, which acts as a window that slides over the image, creating multiple representations of the input. This process creates the convolutional layers, which then feed into the fully connected layers, which are called fully-connected because each node in the previous layer is connected to every node in the next. And finally, the last layer outputs a result that is one of the classes it determines the input to be.

5.3 Other Variations of Neural Nets

In the course of our research we also utilized variations based on these two types of neural networks to see if we could produce better results. These included a 3D CNN, ResNet-18, and a Convolutional Variational Autoencoder (CVAE). The 3D CNN is nearly identical to a normal CNN, only that it uses 3D convolutional layers instead of traditional 2D ones. ResNet-18 is a CNN model developed by He, et al.15 that was one of the first "deep" neural nets in that it contains many (18 in this case) layers. Finally, a CVAE is a net with essentially the same structure as a normal VAE, except its encoding and decoding layers are composed of convolutional layers.

5.4 Specifications of the Best-Performing Neural Nets

When applied to the MSTAR dataset (which we'll introduce in the next section) we found the only neural nets that produced desirable results were a traditional CNN and a CVAE. The CNN had 2 convolutional layers, 2 fully-connected layers, used dropout and batch-normalization to prevent over fitting, and used negative log-likelihood to calculate its loss. The CVAE contained 4 convolutional layers in its encoder, 2 fully-connected layers, a sampling (latent space) dimension of 32 channels, and 4 transpose-convolutional layers in its decoding. It also utilized dropout.

It is important to note that when we run graph-based learning on an image from a neural network, we are running it on the output of the last layer right before the first fully-connected layer. Reminiscent of the VAE's structure, we referred to this as the "encoding" of the neural network.

6. MSTAR

Before we introduce about the MSTAR dataset, let's understand how Synthetic Aperture Radar (SAR) images work. SAR is a type of radar that utilizes the movement of an antenna over a distance from the target to capture finer resolution images than standard radar. It is usually present in moving objects such as spacecrafts/air crafts and drones. SAR images are useful for Automatic Target Recognition (ATR) problems in military applications. The unique thing about SAR images is the presence of shadows of the target in the image.

Moving and Stationary Target Acquisition and Recognition (MSTAR) is a dataset with a collection of SAR images from 1995-1997. It contains images of 10 distinct types of ground vehicles such as tanks and trucks. Specifically, they are: 2s1 gun, zsu23-4 gun, bmp2 tank, t62 tank, t72 tank, brdm2 truck, zill31 truck, btr60

transport, btr70 transport, and bulldozer. The images are target chips containing magnitude data and phase data in the form of floating point numbers. There are 6874 images in this dataset and each image is 88x88 pixels. The 6874 images are partitioned into a training set of 3671 images and a test set of 3203 images.

The MSTAR dataset can be visualized with a t-SNE embedding in Figure 17. The t-SNE embedding reveals that MSTAR contains very natural clusters within the dataset, and there is minimal overlap between clusters. This insight demonstrates that MSTAR is a great candidate for graph learning.

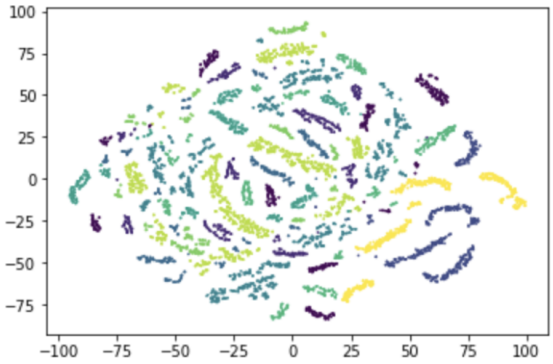

Figure 17. A t-SNE embedding is a way to two-dimensionally visualize a more complex dataset. Each point in the embedding above represents an image in the MSTAR dataset, and each color represents one of the 10 MSTAR classes.

6.1 Pre-Processing the MSTAR Dataset

Remembering that the raw MSTAR data contains a phase channel and a magnitude channel, let P represent the phase channels and let M represent the magnitude channels. The pre-processing introduced in Ref. 16 performs the following transformation:

1. Define $f : M \to [0, 1]$ by $f(m \in M) = \frac{m - min(M)}{max(M) - min(M)}$

2. Let $P_R = sin(P) * f(M)$ be the real component of the phase data and let $P_I = cos(P) * f(M)$ be the imaginary component of the phase data

3. Translate the P_R and P_I to $\mathbb{R}_{\geq 0}$ by:
   if $min(P_R) < 0$ **then**
   $P_R \leftarrow P_R - min(P_R)$
   if $min(P_I) < 0$ **then**
   $P_I \leftarrow P_I - min(P_I)$

4. Define $g_1 : P_R \to [0, 1]$ by $g_1(p \in P_R) = \frac{p - min(P_R)}{max(P_R) - min(P_R)}$ and $g_2 : P_I \to [0, 1]$ by $g_2(p \in P_I) = \frac{p - min(P_I)}{max(P_I) - min(P_I)}$

5. Make a 3 channel dataset where the channels are: $(M, g_1(P_R), g_2(P_I))$.

The version of the MSTAR dataset that undergoes this transformation is referred to as polar MSTAR throughout this paper. Experimentation with using polar MSTAR vs. raw MSTAR when feeding MSTAR through a CNN revealed that polar MSTAR yields a slightly higher classification accuracy.

7. EXPERIMENTAL RESULTS: GRAPH LEARNING ON MSTAR

In order to do graph learning on MSTAR, we first fed polar MSTAR through the CNN from Section 5.4 to obtain an encoded version of the MSTAR dataset. Using an encoded version of MSTAR rather than the raw

data yielded higher classification accuracies. Graph Learning was then ran using the encoded version of the data. All classification accuracies are calculated using the MSTAR test dataset.
In Figure 18, the CNN trained on 30% of the MSTAR dataset obtained a 92% classification accuracy. But, when graph learning was run on the encoded version of MSTAR obtained by feeding polar MSTAR through the CNN, Laplace learning was able to reach a near perfect classification accuracy when also trained on 30% of the data. In addition to this, Laplace learning only needed 5% of the MSTAR data as training data in order to match the classification accuracy of the CNN.

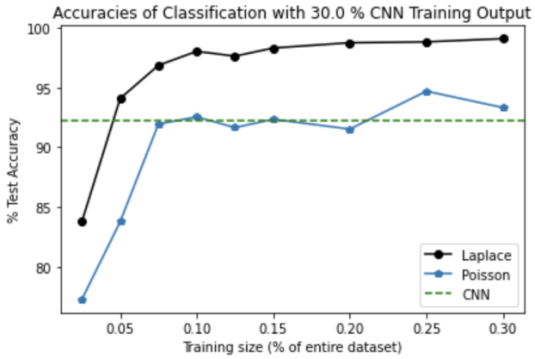

Figure 18. A CNN was trained using 30% of the MSTAR dataset as training data. The entire MSTAR dataset was then fed through the CNN to obtain an encoded version of the data.

In Figure 19, the CNN trained on 7.5% of the MSTAR dataset obtained an 82% classification accuracy. The drop in accuracy was expected as CNNs are known for requiring a lot of training data. But, when graph learning was run on the encoded version of MSTAR obtained by feeding the dataset through the CNN, Laplace learning was able to reach a 95% classification accuracy while similarly using 7.5% of the MSTAR dataset as a training dataset. These experiments with graph learning on the encoded version of MSTAR reveal that not only is graph

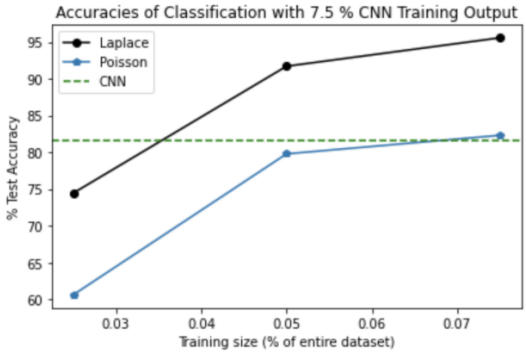

Figure 19. A CNN was trained using 7.5% of the MSTAR dataset as training data. The entire MSTAR dataset was then fed through the CNN to obtain an encoded version of the data.

learning able to achieve a higher classification accuracy than a CNN on encoded MSTAR images, but graph learning is also able to do so with less training data.

7.1 MBO Scheme Results

Using the encoded data from the CNN, we performed experiments using multiclass and modularity MBO schemes. Currently the only weight matrix processing method that works with modularity MBO is kNN. Both Nyström

extension and kNN work with multiclass MBO, and using a dense ZMP weight is also feasible since the MSTAR dataset is not forbiddingly large. Regardless of the exact choice of the weight function, multiclass MBO has poor performance at low label rate, but with more labels it can ultimately reach an accuracy as high as about 98% when coupled with kNN weights. Modularity MBO have much better performance at low label rate, but it currently has a bottleneck of around 88% at high label rate, which may be improved in the future by coupling modularity MBO with active learning.

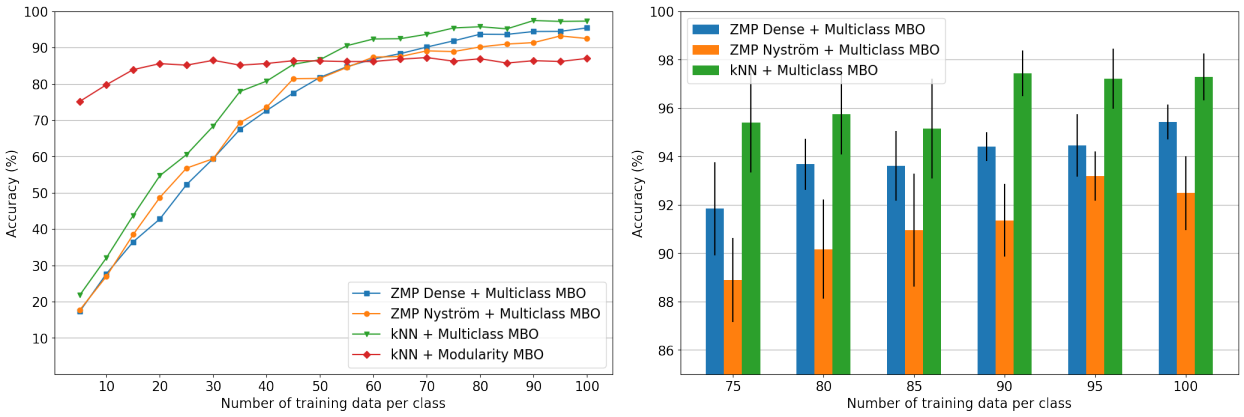

Figure 20. Accuracy of classification of the MSTAR dataset using multiclass and modularity MBO schemes against the number of training data per class. For each number of training data per class, 10 trials are run to account for the randomness in the choice of training labels.

7.2 CVAE Training and Results

In addition to developing a CNN for MSTAR, we also tailored a CVAE to pre-process the data for graph learning, in hopes of getting high accuracies with no labeled data required for training the network. Shown in Figures 19, 20, and 21, we see the CVAE's reconstruction of some of the 88x88 images after the 1st, 25th, and 50th epochs, and it is clearly visible that there is substantial improvement. After 50 epochs, we run the entire dataset

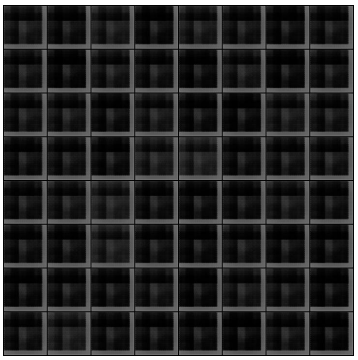

Figure 21. 1st Epoch.

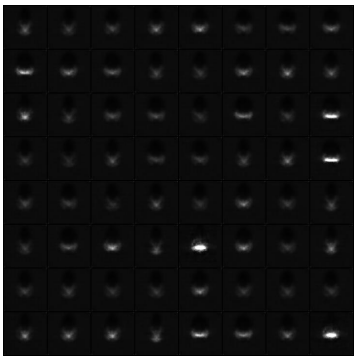

Figure 22. 25th Epoch.

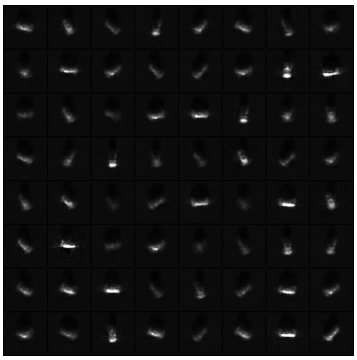

Figure 23. 50th Epoch.

through the trained encoding function, and conduct Laplace learning using varying amounts of labels per class. The results of this are shown in Figure 24.

7.3 Comparison with Support Vector Machine

Support vector machines (SVM) are often used for classification of linearly separable data. The natural clustering in the MSTAR dataset visible in Figure 17 prompts the use of a SVM on the MSTAR dataset. The training data for the SVM was obtained by feeding polar MSTAR through a CNN trained on the entire polar MSTAR training dataset, and the entire polar MSTAR training set was used to train the SVM. The SVM with a linear kernel

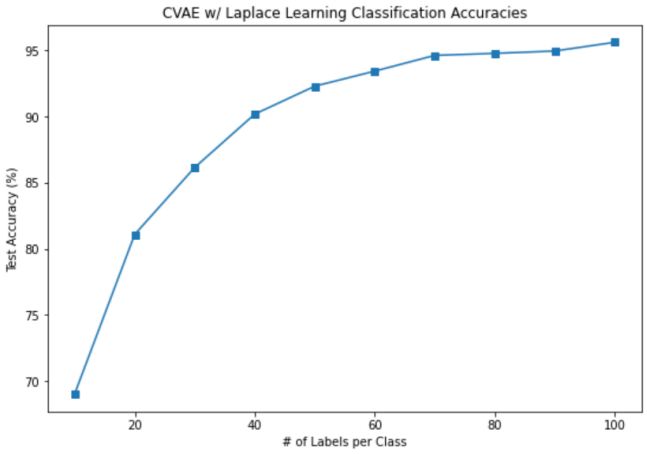

Figure 24. Accuracies of Laplace learning on MSTAR dataset after fed on a CVAE trained over 50 epochs. For reference, 50 labels per class is about 7.3% of the entire dataset, and it gives us an average accuracy of 92.28%.

trained on this training dataset achieved a 90.45% classification accuracy on the polar MSTAR test dataset. Because graph learning is able to reach a higher classification accuracy on the test set of MSTAR, this result further motivates the use of graph learning with MSTAR.

8. EXPERIMENTAL RESULTS: ACTIVE LEARNING ON MSTAR

The promising performance of GSSL on MSTAR motivates the exploration of active learning with MSTAR. When calculating classification accuracy, only the built-in test set of MSTAR images was used. We used Laplace Learning as the GSSL algorithm to perform the image classification. Figure 25 depicts the outcome of doing 50 iterations of V-Opt, Model Change, and MC V-Opt on an encoded version of the MSTAR dataset. Random sampling is also depicted in the plot as a baseline for classification accuracy.

8.1 Batch Active Learning on CNN Output

In this trial of active learning, rather than selecting one point to label at each iteration, one point in each predicted class was queried at each iteration of active learning. Because there are 10 classes in MSTAR, 10 points were selected at each active learning iteration, meaning 500 labeled points selected during active learning were added to the initial random 2 labeled points per class.

In Figure 25, Laplace learning is able to reach a slightly higher classification accuracy than the CNN after 50 iterations of active learning. V-Opt reaches a higher classification accuracy slightly quicker than other active learning techniques, and MC V-Opt does not perform as well on this version of the dataset, but overall the various active learning techniques behave similarly.

8.2 Sequential Active Learning on CNN Output

In this trial of active learning, we are selecting one point to label every iteration based on the respective objective values of each method. The methods we perform in this trial are Uncertainty Sampling, V-Optimality, Σ-Optimality, and Model Change-V Optimality. Initially, we randomly select 10 labels per class as our initial labeled set. Then, we perform 100 iterations of active learning selection to select a total of 200 labeled points.

Based on Figure 26, we can see that the active learning methods certainly beat random sampling. Furthermore, active learning easily beats the CNN test accuracy (83.55%) with just around 150 labeled points. One interesting result we find is how Uncertainty Sampling performs great and even better than the other active learning methods. We suspect the reason for this is due to the t-SNE embedding (Figure 17) where we can see clear separation between the clusters and thus most of the points in this dataset are considered to be ambiguous.

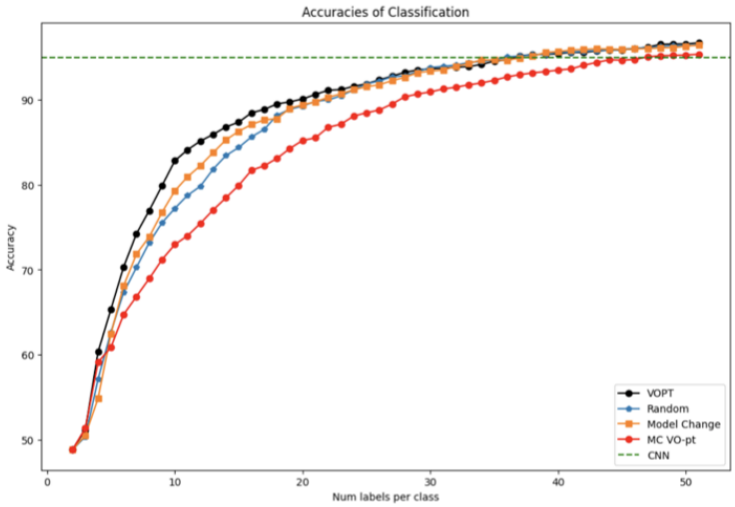

Figure 25. 50 iterations of active learning were run on the encoded version of the MSTAR dataset obtained by feeding polar MSTAR through a CNN trained on 40% of the MSTAR dataset. This plot is an average of 10 trials.

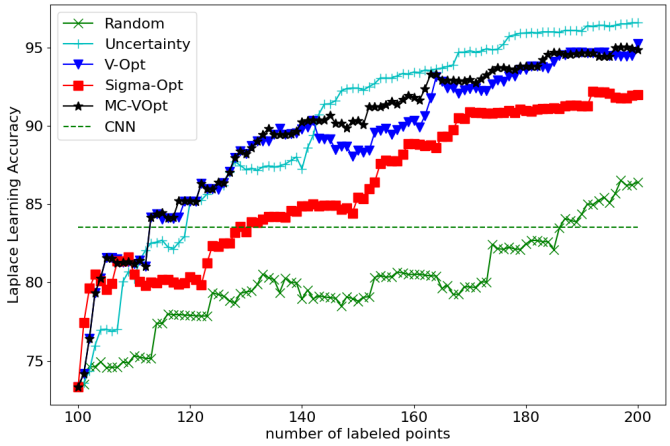

Figure 26. 100 iterations of active learning label selection on the encoded version of MSTAR through a CNN trained using 7.5% training data

8.3 Sequential Active Learning on CVAE Output

Lastly, we explored the use of active learning with CVAE to preprocess the MSTAR dataset. Once again, the motivation behind this is that CVAE doesn't require any training data while CNN requires a certain amount of training data to perform well. Since CVAE didn't use any training data, we generally need more labeled data when performing graph learning to achieve high accuracies.

In this trial, we randomly select 20 labels per class as our initial labeled set. Then, we perform 100 iterations of active learning selection to select a total of 300 labeled points. Figure 27 shows how we can still achieve high accuracies without using any training data to get the encoded data. The only caveat is that we need more labeled points than using just CNN.

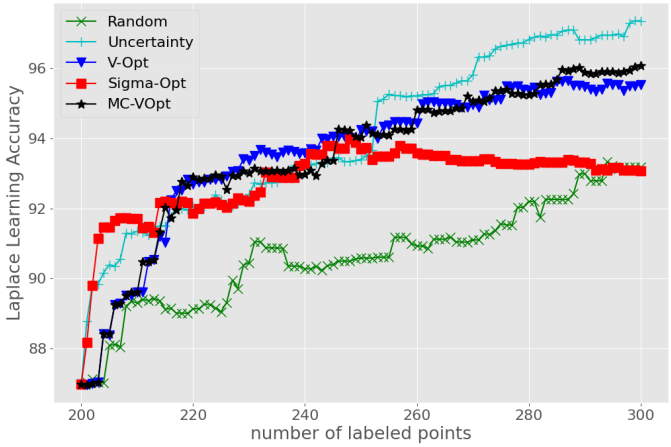

Figure 27. 100 iterations of active learning label selection on the encoded version of MSTAR through a CVAE with no training data required.

9. CONCLUSION

Our work has combined Convolutional Neural Networks, Variational Autoencoders, and GSSL classification methods and explored the applications of these methods to SAR data, namely with the MSTAR dataset. Our experimental results demonstrate that graph learning outperforms other classification methods on the MSTAR dataset when coupled with a CNN or CVAE. Promising results have been presented for both batch and sequential active learning methods with Laplace Learning on SAR data, but particularly promising results have been observed with sequential active learning.

Because CVAE does not require labeled training data, it has extraordinary potential for usage in low-label rate problems. Future work to be done includes refining various active learning methods to perform even better on the CVAE representations of SAR data. Applications to low-label rate problems could also be enhanced by an exploration of transfer learning or few shot learning on the CNN used to create the encoded version of the SAR data.

ACKNOWLEDGMENTS

We wish to thank Dr. Jeff Calder, Dr. Andrea Bertozzi, Kevin Miller, Bohan Chen, and Jason Brown, as well as the UCLA CAM REU program for providing us this incredible opportunity to further our knowledge and passion for applied mathematics.

REFERENCES

[1] Settles, B., "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009).

[2] Zelnik-Manor, L. and Perona, P., "Self-tuning spectral clustering," in [*Proceedings of the 17th International Conference on Neural Information Processing Systems*], *NIPS'04*, 1601–1608, MIT Press, Cambridge, MA, USA (2004).

[3] Bertozzi, A. L. and Flenner, A., "Diffuse interface models on graphs for classification of high dimensional data," *SIAM Rev.* **58**, 293–328 (Jan. 2016).

[4] Zhu, X., Ghahramani, Z., and Lafferty, J. D., "Semi-supervised learning using gaussian fields and harmonic functions," in [*Proceedings of the 20th International conference on Machine learning (ICML-03)*], 912–919 (2003).

[5] Calder, J., Cook, B., Thorpe, M., and Slepčev, D., "Poisson learning: Graph based semi-supervised learning at very low label rates," *CoRR* (2020).

[6] Kohn, R. V. and Sternberg, P., "Local minimisers and singular perturbations," *Proceedings of the Royal Society of Edinburgh: Section A Mathematics* **111**(1-2), 69–84 (1989).

[7] Garcia-Cardona, C., Merkurjev, E., Bertozzi, A. L., Flenner, A., and Percus, A. G., "Multiclass data segmentation using diffuse interface methods on graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(8), 1600–1613 (2014).

[8] Merriman, B., Bence, J. K., and Osher, S., "Motion of multiple junctions: a level set approach," *Journal of Computational Physics* **112**, 334–363 (1994).

[9] Boyd, Z. M., Bae, E., Tai, X.-C., and Bertozzi, A. L., "Simplified energy landscape for modularity using total variation," *SIAM Journal on Applied Mathematics* **78**(5), 2439–2464 (2018).

[10] Ji, M. and Han, J., "A variance minimization criterion to active learning on graphs," in [*Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*], Lawrence, N. D. and Girolami, M., eds., *Proceedings of Machine Learning Research* **22**, 556–564, PMLR, La Palma, Canary Islands (21–23 Apr 2012).

[11] Ma, Y., Garnett, R., and Schneider, J. G., "Σ-optimality for active learning on gaussian random fields.," in [*NIPS*], 2751–2759 (2013).

[12] Miller, K., Li, H., and Bertozzi, A. L., "Efficient graph-based active learning with probit likelihood via gaussian approximations," *arXiv preprint arXiv:2007.11126* (2020).

[13] Meng, Z., Merkurjev, E., Koniges, A., and Bertozzi, A. L., "Hyperspectral image classification using graph clustering methods," *Image Processing On Line* **7**, 218–245 (2017).

[14] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," in [*Advances in neural information processing systems*], 849–856 (2002).

[15] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," (2015).

[16] Coman, C. and Thaens, R., "A deep learning sar target classification experiment on mstar dataset," in [*2018 19th International Radar Symposium (IRS)*], 1–6 (2018).