Utilizing Contrastive Learning for Graph-Based Active Learning of SAR Data*

Jason Brown[a], Riley O'Neill[b], Jeff Calder[b], and Andrea L. 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

ABSTRACT

Automatic target recognition with synthetic aperture radar (SAR) data is a challenging image classification problem due to the difficulty in acquiring the large labeled training sets required for conventional deep learning methods. Recent work[1] addressed this problem by utilizing powerful tools in graph-based semi-supervised learning and active learning, and achieved state of the art results on the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset with less labeled data compared to existing techniques. A key part of the previous work was the use of unsupervised deep learning, in particular, a convolutional variational autoencoder, to embed the MSTAR images into a meaningful feature space prior to constructing the similarity graph. In this paper, we develop a contrastive SimCLR framework for feature extraction from MSTAR images by using data augmentations specific to SAR imagery. We show that our contrastive embedding results in improved performance over the variational autoencoder similarity graph method in automatic target recognition on the MSTAR dataset. We also perform a comparative study of the quality of the autoencoder and contrastive embeddings by training support vector machines (SVM) at various label rates, applying spectral clustering, and evaluating graph-cut energies, all of which show that the contrastive learning embedding is superior to the autoencoder embedding.

Keywords: Contrastive Learning, Active Learning, Synthetic Aperture Radar, Graph-Based Learning

1. INTRODUCTION

Synthetic Aperture Radar (SAR) imaging is a remote sensing technique for collecting high resolution reconstructions from resolution-limited apertures mounted to moving objects, such as planes and satellites. For traditional, stationary radar sensors to yield high resolution images, a large aperture is required, which is infeasible for many applications. By using sensors mounted to objects moving along straight trajectories, SAR imaging provides high resolution scans from smaller apertures while being relatively easy to collect. SAR imaging finds wide usage in the detection and identification of specific targets, particularly for military applications. However, humans find difficulty in interpreting and labeling SAR data, which is required for acquiring large databases to train classification models on. This incentivizes the use of machine learning methods that are robust in low label rate environments.

Many state of the art (SOTA) methods for image classification rely on supervised learning, which typically requires a significant amount of labeled data to train neural networks that generalize well. This makes supervised learning less suited to use in classification of SAR data, where very few labels are available. To train a model that generalizes well with very little labeled data, we turn to semi-supervised and unsupervised learning methods. Semi-supervised learning (SSL) techniques use a combination of some labeled data with large amounts of unlabeled data, while unsupervised techniques use only unlabeled data. The effectiveness of these methods is derived from their ability to leverage the the structure and geometry of the unlabeled data.

* **Source code:** `https://github.com/jasbrown96/Contrastive-Active-Learning`

Further author information: (Send correspondence to J.B.)
J.B.: E-mail: jasbrown@g.ucla.edu

A more recent learning regime is active learning, which iteratively queries a human oracle for additional labeled information to improve performance. Like semi-supervised learning, active learning uses very few labels and leverages information from unlabeled data. Unlike semi-supervised learning, it identifies where new labels would be most beneficial in improving accuracy, and uses a human-in-the-loop procedure to iteratively expand the set of labeled data. First the model trains on a partially labeled dataset, then an acquisition function determines which unlabeled point(s) would be most beneficial to get labels for, after which the human adds the labels to the labeled set before starting the process over again. The process iterates until performance reaches a desired level or the energy to continue the process is excessive.

Recently work (e.g.[1–4]) shows the efficacy of graph-based methods in semi-supervised learning (SSL) and active learning. Treating each sample in the dataset as a node on the graph, the graph edges reflect the similarity between pairs of samples as measured by a similarity function. This imbues the dataset with a structure agnostic towards many of the high dimensional artefacts and noise that otherwise inhibit accurate predictions. After constructing a graph on a partially labeled dataset, the labels can be propagated throughout the graph using various semi-supervised methods. Largely rooted in partial differential equations (PDEs), these include the seminal Laplace learning,[5] also called label propagation, the graph MBO method,[6] p-Laplace learning,[4] and graph-based Poisson learning.[2,7] In such graph frameworks, it becomes quite simple to use acquisition functions to measure the desirability of unlabeled points. Notable acquisition functions include the uncertainty measure,[8] variance optimality,[9] sigma optimality,[10] and model change,[11] which employ various heuristics to determine the predicted desirability of unlabeled data. In practice, it is costly to apply these acquisition functions to every single unlabeled sample, so often a candidate set is chosen over which the search is conducted—this is known as pool-based sampling. A greedy sampling algorithm is typically used to iteratively select the single unlabeled sample with the highest acquisition function value from the candidate set. These samples are sent to the oracle for labelling for subsequent use in the semi-supervised learning algorithm.

In order for these graph-based methods to work well, there must be a powerful and accurate method for capturing the inherent structure of the data to create the similarity graph. If the graph is not well constructed, then the edges between samples will poorly capture the intrinsic relationship between the samples and hamper the flow of labels via the SSL methods. The process of determining the important characteristics of images is known as feature extraction. Many popular feature extraction methodologies have been advanced over the years, including principle component analysis, non-negative matrix factorization, and variational auto-encoders. Recent work by T. Chen et al. at Google suggest a new method for feature extraction known as SimCLR.[12] This is an unsupervised learning framework for training neural networks that employs *contrastive learning* for feature extraction. At a high level, SimCLR trains a neural network on the task of distinguishing images that are the identical, up to a transformation from a predetermined set or group. The key to a successful application of SimCLR is in the choice of the transformations (e.g., data augmentations) that one wants the model to be "invariant" to, which should resemble what one expects to see within the dataset. More specifically, the framework applies two random (distinct) augmentations to each image in a batch to create a *positive pair* of images. *Negative pairs* refer to augmented images produced from other images in the batch (possibly the same class). A neural network for embeddings is then trained by putting the image pairs through the network and maximizing the similarity between positive pairs while minimizing the similarity between negative pairs. SimCLR has demonstrated SOTA performance on ImageNet and numerous other datasets, notably yielding highly separable embeddings.[12]

In this paper, we build on previous work[1] in SAR classification in several directions. Firstly, we develop a SimCLR framework for MSTAR images by introducing novel data augmentations specific to SAR images. This results in markedly improved feature extraction and graph structures. We show that the contrastive SimCLR feature embeddings are of far higher quality (i.e., more distinct and separable classes) when compared with those obtained from variational autoencoders[1] and raw images. We evaluate the quality of the embeddings with SVM, spectral clustering, and graph cut energies. Secondly, we show the power of our SimCLR embeddings when paired with active learning frameworks, showing significantly faster learning and drastically improved accuracy at low label rates compared to Miller et al.[1]

The rest of the paper is organized as follows. Section 1.1 details the previous and related works for low label rate SAR classification. Section 2 showcases the methods used in this paper, primarily focusing on the efforts

of Miller et al.[1] and Chen et al.[12] More specifically, we discuss the methods used by Chen et al. in 2.1 and we discuss the methods used by Miller et al. for graph construction in 2.2.1, graph learning in 2.2.2, and active learning in 2.2.3. Section 3 details the specific measures taken to adapt the previously mentioned methods to the MSTAR dataset as well as a discussion about the neural network that was used for SimCLR. Section 4 contains the results from the experiments performed on MSTAR before the conclusion in section 5.

1.1 Previous and Related Work

Work on Automatic Target Recognition for SAR data focuses heavily on the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset, discussed in section 3.1. Most of the work directly applied to MSTAR can be separated into pre-deep learning approaches, such as support vector machines, and more modern deep learning approaches, such as convolutional neural networks. Pre-deep learning methods are varied in their approaches. Researchers have used feature extraction combined with scattering models[13,14] as one approach. Many approaches utilize SVMs for their classification; such as in a straightforward manner in,[15] paired with Adaboost,[16] or in parallel with a hand-tuned covariance embedding scheme.[17]

Convolutional Neural Networks (CNNs) are at the heart of some of the most successful state of the art machine learning methods, particularly for for image classification. CNNs utilize stacked convolutions, activation functions, and fully connected linear layers to learn features. The main issue with applying CNNs to MSTAR is that they often need large amounts of training data to successfully make accurate predictions. The MSTAR dataset is relatively small with less than 4000 chips. Furthermore, methods that successfully use very few labels are extremely desirable and CNNs typically require many labels in their training set. This problem has lead to various modifications that allow CNNs to function well despite the challenges. Some techniques for utilizing CNNs target the overfitting problem, while still keeping the general structure. One approach uses additional regularization during the training process, such as the max-norm regularization.[18] Another approach, the All-Convolutional Networks, replaced fully connected layers with additional convolutional layers.[19,20] Other researchers have simply tried to minimize the number of parameters in the CNN.[21] Other types of approaches include additional unsupervised methods, such as variational autoencoders, which typically do not require labels to learn features. The Euclidean Distance Restricted Autoencoder[22] method uses an autoencoder designed to embed images from the same class nearby to extract features before classifying the data with an SVM.

One particularly interesting approach to working with limited datasets is to generate additional synthetic data by attempting to replicate SAR images using computer software. In the SAMPLE dataset, researchers present a dataset of real and synthetic data based on the popular MSTAR dataset. The MSTAR dataset consists of 10 distinct classes of vehicles with the images being taken via an aerial radar. By using Computer Aided Design (CAD) models for the vehicles, researchers have reproduced synthetic SAR images of these same vehicles under the same conditions, leading to the SAMPLE dataset.[23] This posed an interesting problem in determining how synthetic data, which is implicitly labeled by construction, could be used to make predictions about unlabeled real world data. Unfortunately, it became apparent that the synthetic data was quite distinct from the real world data, leading to extra difficulties.[23] Some efforts, however, include training fully supervised convolutional neural networks on just the synthetic data and then using that to label the unlabeled data.[24,25] Other efforts include using GANs (generative adversarial neural networks) to augment the synthetic data to make it more similar to real world data, hopefully boosting its usefulness towards classifying real world data.[26]

2. METHODS

In this section, we discuss the mathematical formulation and neural network architectures used to develop the low-label rate classification algorithm.

2.1 Feature Extraction - Variational AutoEncoder

For feature extraction of images, there are several prevalent techniques used in industry, such as principal component analysis, non-negative matrix factorization, and variational autoencoders. Variational autoencoders[27,28] (VAE) have proven to be a successful methods for feature extraction of SAR data and were used in previous work.[1,20,22] VAEs are a generative class of neural networks that learn the important features of images via encoding the images to a low dimensional space before reconstructing the images in the high dimensional space.

They have three main components to their architecture: the encoder, the bottleneck, and the decoder. The encoder portion of the neural network primarily consists of 2D convolutions that compress the images to a low dimensional space. The bottleneck follows the encoder portion of the neural network and includes the reparameterization step, which consists of a multi-layer perceptron tasked with learning the mean and standard deviation from the image. Beyond the bottleneck step, a decoder network which primarily consists of transpose convolution layers will attempt to rebuild the original image based on the mean and standard deviation passed from the bottleneck layer. When properly trained, the decoder portion of the network can generate new images from samples of the latent space due to the enforced regularization in the training process. For the purposes of feature extraction, we instead focus on using the encoder network to retrieve the latent representations of images in our dataset.

2.2 Feature Extraction - Contrastive Learning

An alternate neural network approach to VAE feature extraction is a recently revitalized learning framework known as contrastive learning. The impetus of contrastive learning is that an encoder CNN trains by minimizing the distance of like-images and maximizing the separation of "unlike"-images in the encoding space (or a projection of it). The images that are designated as similar samples are called positive pairs whereas the images that are designated as dissimilar are denoted negatives. The name comes from comparing and contrasting images.[29] Recently, Chen et al.[12] published a seminal paper on self-supervised contrastive learning for visual representations in which they introduce a novel framework known as SimCLR . Since SimCLR is self-supervised, the method implicitly determines which samples are positive pairs to act as the labels that guide the feature learning. Given a batch of n images, a series of augmentations are performed such that each image will have 2 different resulting images after the augmentations, resulting in $2n$ augmented images from the batch. For an image x_i, the augmentations will produce two augmented images that we will call x_{i_1} and x_{i_2} respectively. The underlying heuristic for SimCLR is that since these two augmented samples, x_{i_1} and x_{i_2}, were both derived from the same image, x_i, then they should be considered positive pairs and all the other $2(n-1)$ augmented images in the batch should be considered negatives.

To quantify the similarity between the features extracted from two images, u and v, Chen[12] uses the cosine similarity function as

$$\text{sim}(u, v) = u^T v / ||u|| ||v||.$$

For a given positive pair, x_{i_1}, x_{i_2}, the loss from sample x_{i_1} is defined to be

$$l_{i_1} = -\log \frac{\exp(\text{sim}(x_{i_1}, x_{i_2})/\tau)}{\sum_{k=1}^{N} \sum_{j=1}^{2} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(x_{i_1}, x_{k_j})/\tau)}$$

where τ is a temperature parameter that must be fine tuned and controls the relaxation for the similarity function. The loss, summed over an entire batch, is called *NT-Xent*, which stands for normalized temperature-scaled cross entropy loss. In summary, the loss for a batch of n samples is

$$L = \frac{1}{2n} \sum_{i=1}^{n} [l_{i_1} + l_{i_2}].$$

Now we discuss the network architecture in order to properly utilize the framework. Given a batch of n images, denoted $x_1, ..., x_n$, each image will be augmented twice through some sequence of augmentations to create a new batch of size $2n$ consisting of images $x_{1_1}, x_{1_2}, ..., x_{n_1}, x_{n_2}$. Each one of these images in the new batch will be passed through the first phase of the neural network which is called the encoder. The encoder will primarily consist of convolutional layers tasked with feature extraction and will transform a sample x_{i_j} into a feature embedding denoted as h_{i_j}. Afterwards, a projection head, which is typically a multi layer perceptron with one or two hidden layers will be used. The projection head transforms a feature embedding h_{i_j} to a new space where the outputs are denoted as z_{i_j}. Afterwards, the NT-Xent loss function is evaluated on the projected features z_{i_j} and the encoder and projection head networks are simultaneously updated towards minimizing the loss. An outline of the SimCLR framework is shown in Figure 1. After having trained the neural network on a

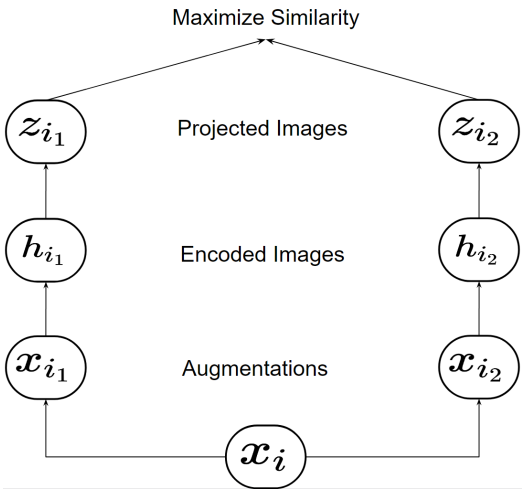


Figure 1. In the SimCLR framework, a sample image x_i is augmented twice creating x_{i_1} and x_{i_2}. These augmentated images are pushed through an encoder network and a projection head resulting in z_{i_1} and z_{i_2}. An objective of SimCLR is to maximize the similarity between positive pairs such as these.

given dataset, the projection head is discarded as the purpose of the SimCLR framework is to simply train the encoder to produce stronger embeddings. The projection head is primarily used to strengthen the learning rate and improves the robustness of the encoder network.[12]

2.2.1 Graph Embedding

Given a feature extraction process, we can view images as vectors embedded in $\mathbb{R}^d$, a much lower dimensional space, than the raw data. Quantitative comparisons can be performed on and between the image feature vectors to construct a similarity graph for the dataset. Like SimCLR, we use the cosine similarity metric to construct a k nearest neighbors graph from the embedded images. A fast nearest neighbors approximation algorithm is performed to boost computation. Once the k nearest neighbors for each sample are selected, the edge weights in the resulting graph must be determined and we choose a self-tuning similarity graph with edge weights given by

$$w_{ij} = \exp(-4||x_i - x_j||^2/d_k(x_i)^2),$$

where x_i, x_j represent the normalized features for the i^{th} and j^{th} images respectively, and $d_k(x_i)$ represents the distance between x_i and its k^{th} nearest neighbor.[1]

2.2.2 Graph Based Semi Supervised Learning

Let $\{x_1, ..., x_n\} := \mathcal{X} \subset \mathbb{R}^d$ represent the feature vectors of the images in d dimension and let $\mathcal{L} \subset \{1, 2, ..., n\}$ represent the set of indices for feature vectors that have an associated, known label. We define $\mathcal{U} = \{1, 2, ..., n\} - \mathcal{L}$ to represent the set of indices of unlabeled images. For a dataset with K classes, we let $y_j \in \{1, 2, ..., K\}$ represent the label for image j so that the labels correspond to their one-hot encoding as $e_{y_j} \in \mathbb{R}^K$. Let $\mathcal{G}(\mathcal{X}, \mathcal{W})$ be a graph where $\mathcal{X}$ represents the vertices and $\mathcal{W}$ represent the edge weights between the images. Given this graph framework, semi-supervised learning techniques have seen great success. In Zhu, Ghahramani, and Lafferty's work,[5] the Laplace learning on graphs is introduced and developed, which is the primary semi-supervised learning technique used throughout this paper. Given a set of labeled data $\mathcal{L}$ and a weighted adjacency matrix W corresponding to the graph, Laplace learning solves for the label function $f : \mathcal{X} \to \mathbb{R}^K$ such that

$$\left.\begin{aligned} f(x_i)^T &= \frac{1}{d_i}\sum_{j=1}^{n} W_{ij} f(x_j)^T \quad && \text{for } i \notin \mathcal{L} \\ f(x_i) &= \mathbf{e}_{y_i} && \text{for } i \in \mathcal{L} \end{aligned}\right\}, \tag{1}$$

where $d_i = \sum_{j \neq i} W_{ij}$. In principle, this solution generates a graph harmonic function which propagates the labels across the dataset. The solution to this energy equation can be written in terms of the graph Laplacian,

$L = D - W$ where D represents the degree matrix for the graph. Without loss of generality, re-ordering the nodes so that the labeled nodes are written first, the solution can be written as

$$F^* := \begin{pmatrix} f^*(\mathbf{x}_1)^T \\ f^*(\mathbf{x}_2)^T \\ \vdots \\ f^*(\mathbf{x}_n)^T \end{pmatrix} = \begin{pmatrix} Y \\ -L_{\mathcal{U},\mathcal{U}} L_{\mathcal{U},\mathcal{L}} Y \end{pmatrix},$$

where $L_{\mathcal{U},\mathcal{U}}$ and $L_{\mathcal{U},\mathcal{L}}$ represents the lower-right and lower-left blocks of L respectively and $Y \in \{0,1\}^{|\mathcal{L}\times K|}$ is the matrix whose rows are e_{y_j} for the corresponding labeled nodes.[1]

2.2.3 Active Learning

To most efficiently leverage the limited amount of labeled data, we employ active learning. Active learning uses acquisition functions on the predictions from the trained model to determine which unlabeled sample, $i \in \mathcal{U}$ should be labeled by a human expert, often called an oracle. We can now update the labeled set to include the newly queried sample, $\mathcal{L}_{new} = \mathcal{L}_{old} \cup \{i\}$, and then proceed to retrain or fine-tune the prediction model on the new labeled set. This process of querying a single label at a time is known as sequential active learning, as opposed to batch active learning which queries numerous labels at each iteration. The appeal of such a process is that, ideally, minimal effort is expended on labelling samples that do little to benefit the prediction model. Typically, active learning is preferred when either individual samples require significant time and energy to label or there are far too many samples to effectively label. SAR data falls under the first category, as the noisy images are nearly incomprehensible to an untrained human. For active learning to proceed, we need both a prediction model and an acquisition function. For the prediction model, we will be using the graph-based Laplace learning method discussed in section 2.2.2. We will discuss several acquisition functions in the following section, but the uncertainty acquisition function will be the most significant.

We consider prediction models taking in unlabeled nodes and outputting a vector of length K such that the i^{th} component corresponds to the likelihood that the given sample belongs to the i^{th} class. In practice, the predictions are thresholded so that the model can directly output class predictions, but by using the soft prediction vector, we can establish a notion of confidence and uncertainty in how the model predicts certain classes. An underlying heuristic for much of active learning is that the query target should be a sample that the model is very uncertain about, since it likely contains information that is novel to the model. This leads to the uncertainty acquisition function. Given a model prediction $v^*(x_i) \in \mathbb{R}^K$ for the sample $i \in \mathcal{U}$, we define the margin to be the difference between the first and second highest predictions:

$$\text{Margin}(x) := \max_k u(x)_k - \max_{l \neq \text{argmax}_k u(x_i)} u(x)_l,$$

where $y^*(i)$ is the largest prediction value in $u^*(i)$. The uncertainty acquisition function queries the sample that lies closest to the decision boundary, which represents the border between expected classes in the data. Heuristically, the uncertainty acquisition function will be able to target regions of the dataset that are low confidence and allows for refinement.[8] Another popular acquisition function is variance optimality (VOPT). VOPT is agnostic of the observed and predicted labels and essentially attempts to query the labels such that the unlabeled data suffers the least amount of variance in terms of its distribution.[30] The last acquisition function to be mentioned is Model Change.[11] Model Change is a look-forward acquisition function that uses Gaussian distributions to approximate the significance of an unlabeled point by estimating the amount of change it would cause in the model.[11] There has been work done to combine the ideas of model change with variance optimality resulting in MCVOPT as another, derivative acquisition function.[1]

3. METHODS & DATASET

In this section we describe the Moving and Stationary Target Acquisition and Recognition, or MSTAR, dataset, and the modifications of the SimCLR framework.

3.1 MSTAR Dataset

The MSTAR dataset[31] was published in 1998 by Sandia National Laboratory with funding from the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory. The MSTAR dataset consists of 6,874 SAR chip images, collected by a Sandia X-band radar operating at 9.60GHz with a bandwidth of 0.591GHz. Designed for automatic target recognition, the dataset features 10 distinct vehicle classes (Armored Personnel Carrier: BMP-2, BRDM-2, BTR-60, and BTR-70; Tank: T-62, T-72; Weapon System: 2S1; Air Defense Unit: ZSU-234; Truck: ZIL-131; Bulldozer: D7). It is standard to split the MSTAR dataset by the angle of capture where an angle of 15° corresponds to the training data and and angle of 17° corresponds to the testing data.[31] Although active learning does not require the entire training set of data, we still restrict the queried labels to be from the training set so that comparisons to other techniques and methods in the literature remain cohesive.

We preprocess the data in the same process described in Miller et al.[1] That is, firstly, the magnitude and phase images are center-cropped to 88x88 pixels. To reduce presumed noise, pixel values are clipped to the range of [0,1], which is an acceptable range for interpreting images. Furthermore, the images are forced into a 3-channel format where the first channel is the magnitude of the image and the second and third channels are the real and complex phases of the image respectively. Letting M be the magnitude of the image and P be the phase of the image, the 3 channel format is given as

$$\left(M,\ \ \frac{1}{2}(M\cos(P)+1),\ \ \frac{1}{2}(M\sin(P)+1)\right).$$

3.2 Selected Augmentations for SimCLR

SimCLR[12] frameworks are only as good as the augmentations employed in them: too harsh an augmentation (e.g. total image corruption) and the network is forced to learn from noise and undesirable artefacts of the transformation (or learn nothing at all); too few augmentations, and the network doesn't generalize well. Care must be taken to ensure that the augmentations used for a given dataset/task are *meaningful* and mimic unseen data. SimCLR frameworks for image classification (e.g. ImageNet, Cifar10) typically use color jitter, random cropping, random horizontal or vertical filps, and random blur. Below we consider the peculiarities of SAR images and the MSTAR dataset, assess which augmentations are suitable and unsuitable, and propose new custom augmentations which we ultimately use for training an encoder.

Firstly, since MSTAR images are taken from airplanes, the scanned vehicles cast a shadow where no radar signals were received. This shadow is always behind the vehicles, and we would naturally want our encoder model to learn this important feature, pertinent to MSTAR. Therefore, we opt not to use augmentations for the dataset that would rotate or flip the images vertically, as this would destroy the shadow effect always being

behind the target. Another concern is that SimCLR[12] image classification neural networks typically attribute great success to using a color jitter augmentation (as with ImageNet and Cifar10), which randomly shift the entire color distributions in an image. However, SAR images are not RGB images and instead have magnitude and phase information, so we elected to not make use of the color jitter augmentation.The last consideration is that, unlike ImageNet images, the MSTAR chips have the vehicle centered in the middle of the chip for each image. To be consistent with the dataset, we opted not to use random cropping, as most meaningful features are in the center of the image. This inspired a random center crop augmentation where an integer $40 \leq k \leq 88$ is randomly selected, and the image is then cropped around the center to produce a $k \times k$ image that would be resized to 32×32. Overall this builds scale and zoom invariance. The downsizing of the image to 32×32 mitigates memory usage issues and the dimension being a power of 2 allows for several max-pooling CNN layers, granting the encoder greater capacity for abstractification and less susceptibility to pixel-wise minutia. The other two standard data augmentations that were applied to the MSTAR images were a random horizontal flip, which flips the image horizontally 50% of the time and a random Gaussian blur transformation that used a 7×7 kernel and a random sigma value randomly selected between 0.1 and 2.0 for each augmentation.

3.3 Network Architecture for SimCLR

The seminal SimCLR paper[12] demonstrated that a standard ResNet[32] architecture is well-suited for the encoder network and recommends that a 2-layer perceptron is used for the projection head. However, Chen's later work[33] recommends using a slightly deeper projection head. To this end, we employ a 3-layer perceptron instead of a 2-layer perceptron. SimCLR[12] uses a deep ResNet50 for training on Imagenet data. In order to have large batch sizes under memory constraints and prevent overfitting, we opt to use the lighter ResNet18 model on MSTAR as well as the aforementioned downsampling to 32×32 resolution.

3.4 SimCLR and VAE Training Specifications

All code was implemented in Python. The source code to replicate our experiments and evaluation may be found on our GitHub repository[†]. The models were implemented using the PyTorch package, the graph learning and active learning methods were implemented from the GraphLearning Python package,[34] and the SimCLR Pytorch implementation was based on code from SupContrast (Supervised Contrastive) GitHub[‡].[35] The SimCLR ResNet18 was trained for 500 epochs and 1000 epochs with a learning rate of 0.05, batch size of 512, and a SimCLR temperature of 0.5. Training was done using two Nvidia RTX GeForce 2080 GPU's working in parallel. The VAE encoder used the pretrained weights from Miller et al,[1] which is available on their Github[§], the specifics of which can be seen in their paper.

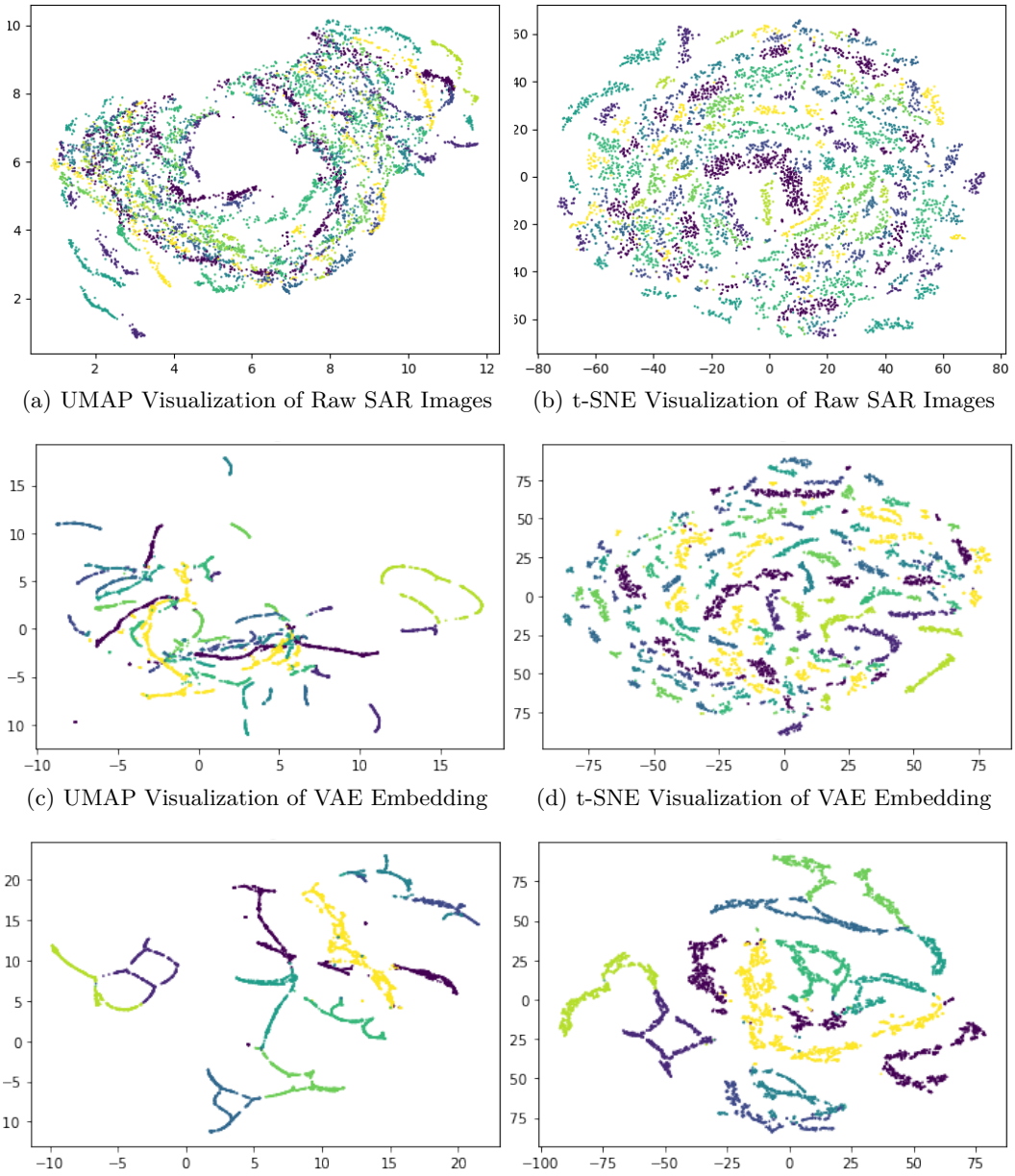
4. RESULTS

The primary motivation of using contrastive learning was to build upon the previous work of Miller et al.,[1] which used VAEs in the feature extraction process on MSTAR. The hypothesis is that using SimCLR as a stronger feature extraction process would lead to a more well-separated graph structure for improved label propagation. To this end, we compare directly with the results of Miller et al.[1]'s VAE, and also examine the use of raw images as a baseline. In 4.1, we compare the quality of the embeddings and resulting graphs from our SimCLR implementation on MSTAR, Miller et al.[1]'s VAE, and the raw images under a series of tests. In 4.2, we further examine the use of the VAE and SimCLR embeddings with active learning on MSTAR.

4.1 Embedding and Graph Quality Results

We assess embedding quality of our SimCLR ResNet18, Miller et al.[1]'s VAE, and the raw flattened SAR images by various different means: t-SNE and UMAP visualizations of the SimCLR and VAE embeddngs, accuracy of support-vector machine (SVM) classifiers over different training/testing splits, graph cut energies, and spectral clustering accuracy.

[†]https://github.com/jasbrown96/Contrastive-Active-Learning
[‡]https://github.com/HobbitLong/SupContrast
[§]https://github.com/jwcalder/MSTAR-Active-Learning



(a) UMAP Visualization of Raw SAR Images (b) t-SNE Visualization of Raw SAR Images

(c) UMAP Visualization of VAE Embedding (d) t-SNE Visualization of VAE Embedding

(e) UMAP Visualization of SimCLR Embedding (f) t-SNE Visualization of SimCLR Embedding

Figure 2. t-SNE and UMAP visualizations. Top row: unprocessed SAR images. Middle row: VAE embedding. Bottom row: SimCLR embedding, trained for 500 epochs. Colors correspond to samples of the same class. The SAR image embeddings exhibit some structure, but this is largely obscured; the classes are largely intermixed and non-connected. The VAE embeddings exhibit greater class cohesion with more connected strands, but intermixing is still evident. With SimCLR, individual classes are far more connected and better separated from others.

First, we compare UMAP[36] and t-SNE[37] visualizations of the SimCLR trained for 500 epochs, VAE, and raw image embeddings, shown in Figure 2. Notably, the classes appear well separated in the t-SNE embedding with very little mixture between labels, but most of the labeled clusters appear disjoint from their respective class. In contrast, the SimCLR t-SNE and UMAP visualizations in Figures 2(e) and 2(f) demonstrate much more cohesion within the labeled clusters. There is very little mixing and the labeled clusters are very well connected with their respective classes; intuitively the SimCLR embeddings should be much better for graph construction.

Embedding	Graph Cut Energy (GCE)	Spectral Clustering Accuracy (%)
SimCLR	**340.301**	**52.66**
VAE	410.941	25.70
Images	1005.815	21.99

Table 1. Graph cut energy (GCE) and spectral clustering accuracy (SCA, given as a percentage) for SimCLR and VAE embeddings as well as raw flattened images, using KNN graph with $k = 20$.

We now consider the graph cut energy and spectral clustering accuracy. We take the weight matrix $W \in \mathbb{R}^{n\times n}$ over n samples to be the K-nearest neighbor (KNN) graph on the respective embeddings (or raw flattened images) with $K = 20$:

$$W_{ij} = 1_{ij} \exp\left(\frac{-4\|x_i - x_j\|}{d_K(x_i)}\right)$$

1_{ij} is the binary adjacency matrix for K nearest neighbors, and $d_K(x_i)$ is the distance to the K-th neighbor of x_i. The graph cut energy GCE measures the weight of all graph edges that would need to be cut in order to split the graph into connected components corresponding to each class. This gives a measure of how well the graph-construction, which is based on the embedding quality, reflects the class membership. The GCE can be computed as

$$GCE(W, U) \equiv \mathrm{tr}(U^T L[W] U)$$

where $L[W] = \mathrm{Diag}(W1_n) - W$ is the graph Laplacian (1_n the all ones vector) and $U \in \mathbb{R}^{n\times k}$ are the 1-hot label vectors for the k classes.

Spectral clustering is a tractable relaxation of minimizing the graph cut energy to split the graph into connected components, still seeking to preserve local connectivity of the graph. It has gained wide usage for unsupervised classification problems in which K-means clustering is insufficient.[38] Numerous methods of normalization exist; we opt to use Ng-Jordan-Weiss normalization.[38,39] Let $D = \mathrm{Diag}(W1_n)$. The algorithm uses the symmetrized Laplacian

$$L_{sym} = D^{-1/2} L D^{-1/2}.$$

Letting $U = (u_1, ..., u_k)$ denote the matrix of eigenvectors corresponding to the first k smallest eigenvalues of L_{sym} ($k = 10$ for 10 classes), the algorithm normalizes U by row norms for $\tilde{U}[i,:] = U[i,:]/\|U[i,:]\|$, and performs K-Means clustering each $\tilde{U}[i,:]$ to determine the cluster of the corresponding point x_i.[38] By registering the identified clusters to the ground truth classes by maximal likelihood, we assess the accuracy of spectral clustering.

The spectral clustering accuracies and graph cut energies are shown in Table 1, comparing a single representative SimCLR model trained for 500 epochs to the VAE model and the raw images. The SimCLR embeddings clearly outperform VAE and the raw images by a sizeable margin (twice as good as VAE in spectral clustering).

Finally, we compare the accuracy of linear SVM. The SVM fitting was done over numerous train/test split ratios, from using 10 labeled points to 3600 for fitting. More split ratios were examined in the lower end of the spectrum, as this is the region of greater interest for low label rates. For each split ratio, 50 random partitions were used to fit an SVM classifier, whereafter the testing accuracies were averaged for the final result. Note that the same partition was consistently applied to the SimCLR embeddings, VAE embeddings, and raw images in each instance. Figure 3 shows the average testing accuracies vs the number of points used to fit the SVM classifiers. SimCLR embeddings trained for 500 epochs and 1000 epochs were compared with the VAE embedding and raw images. The SimCLR curves represent the average SVM performance across 21 distinctly trained models (the 50 partitions applied to each, averaged for each model, then averaged overall). Clearly the SimCLR embeddings outperform the VAE embeddings and raw images, particularly at low-label rates. Interestingly, the raw images outperform the VAE embedding until 620 labeled points. The 1000 epoch SimCLR models slightly outperform the 500 epochs SimCLR models, more notably at low label rates, but the perfomance is quite similar overall.

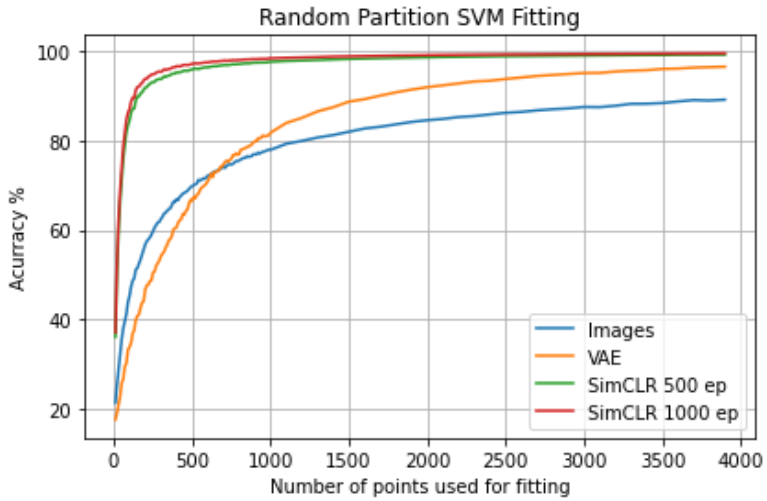


Figure 3. Different training/testing splits for fitting SVM classifiers on SimCLR, VAE, and Image embeddings, averaged over 50 random partitions for each split, partitions randomly generated (susceptible to large class imbalance/underepresentation at low label rates). The SimCLR curves represent the average performance over 21 distinctly trained models, 500 ep and 1000 ep trained for 500 and 1000 epochs respectively. The 1000 epoch SimCLR slightly outperforms the 500 epoch SimCLR, with both drastically outperforming VAE and the images. Interestingly the SAR images briefly outperform the VAE until 620 labeled points.

Note the partitions here are completely random and agnostic to class representation - the extremely low label rate splits may not even see a representative of each class, and class imbalances may persist at higher levels, which may hamper the performance of SVM classifiers. Toward this end, we also examine SVM with equal class representation in fitting at different label rates. Here an equal number of random representatives were selected uniformly from each class. As before, 50 partitions were done for each split level. The results of this can be seen in Figure 4, which compares the VAE embedding, raw images, and average performance of 21 distinct SimCLR models trained for 500 epochs and 1000 epochs. Here again, the SimCLR embeddings clearly outperform the VAE and raw images. The 500 epoch and 1000 epoch SimCLR models still perform similarly overall, again with the 1000 epoch model fairing slightly better at low label rates, and the raw images outperform the VAE embedding until 320 labeled points. Overall, the high accuracies at extremely low label rates suggest the classes in the SimCLR embedding are highly linearly separable and well partitioned - far fewer samples are needed for fitting to achieve good classification accuracy compared to the VAE and raw images.

4.2 Active Learning Results

The t-SNE and UMAP visualizations of the embeddings, shown in Figure 2, along with our other experiments for embedding and graph comparison suggest that graph-based learning methods will be far more effective with the SimCLR embeddings over the VAE embeddings. To see if this is indeed the case, we conduct graph-based active learning with VAE and SimCLR embeddings with various acquisition functions. The results with the SimCLR embeddings are shown in Figure 5 and the results with the VAE embeddings are shown in 6. The active learning results displayed for the SimCLR models are averaged over 21 separately trained models, with 500 and 1000 epochs respectively, to mitigate the effects of noise and properly represent the method. In this section, we compare the averaged active learning results from the model trained to 1000 epochs against the VAE embeddings. Active learning with the VAE embedding yields considerably strong results with the uncertainty acquisition function achieving 94.1% accuracy at approximately 300 labels, representing approximately 5% of the MSTAR dataset. In stark contrast, the SimCLR embedding reaches the same accuracy around 60 labels (about 1% of the dataset) - a drastic improvement at even lower label rates. Remarkably, at the very beginning of the learning rate process, the SimCLR accuracy is over 50% and near optimal accuracy is achieved with every acquisition function after reaching 300 labels. The uncertainty acquisition function performs the best and reaches 99.2% accuracy.

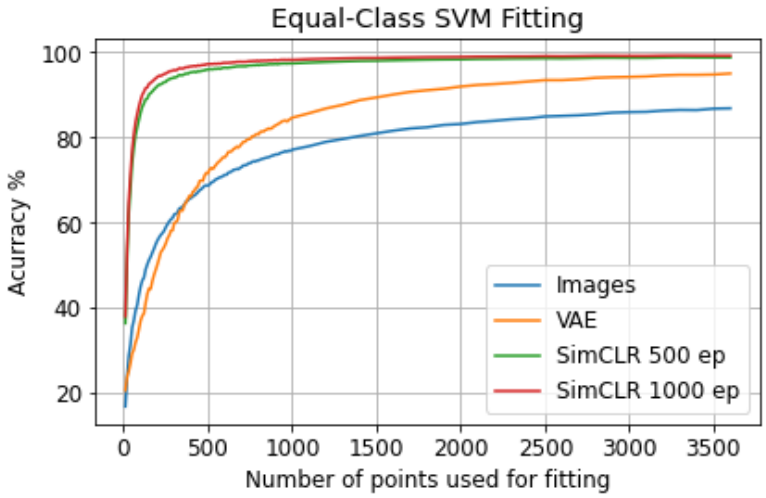
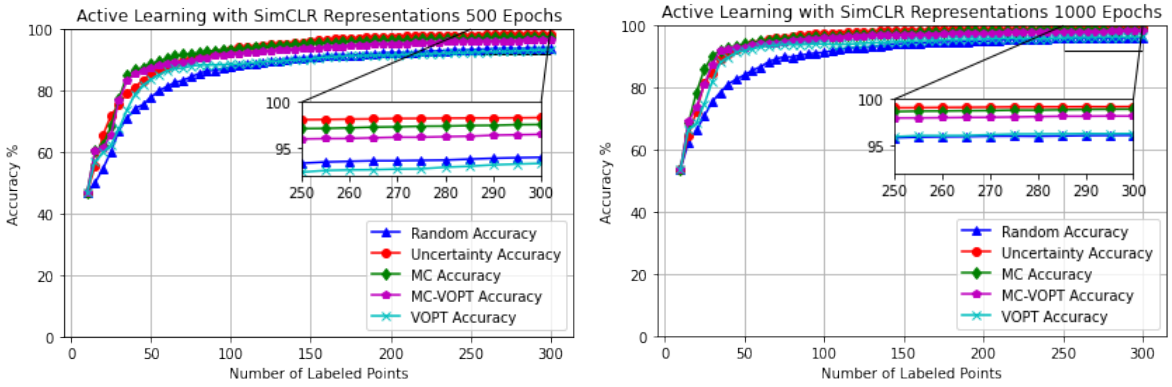


Figure 4. Equal class SVM fitting (same number of points used for each class in fitting) across different training/testing splits on SimCLR, VAE, and Image embeddings, averaged over 50 random partitions for each split. The SimCLR curves represent the average performance over 21 distinctly trained models, 500 ep and 1000 ep trained for 500 and 1000 epochs respectively. The 1000 epoch SimCLR slightly outperforms the 500 epoch SimCLR, with both drastically outperforming VAE and the images. Interestingly the SAR images briefly outperform the VAE until 320 labeled images.



(a) Accuracy of Active Learning with SimCLR Embeddings Trained to 500 Epochs

(b) Accuracy of Active Learning with SimCLR Embeddings Trained to 1000 Epochs

Figure 5. The plots above represent the accuracy of active learning with Laplace semi-supervised learning on SimCLR embeddings, with the models being trained to 500 Epochs in 5(a) and 1000 epochs in 5(b). The results displayed are averaged across 21 distinctly trained models with active learning applied to each model individually. Using 300 labels, the 500 epoch embeddings achieved an average accuracy of 98.3% and the 1000 epoch embeddings achieved an average accuracy of 99.2%.

Figure 7 compares the best performing acquisition function, uncertainty acquisition, using both embeddings. As mentioned previously, the SimCLR embeddings demonstrably outperform the prior embeddings in every way. In the initial setting, with only one label per each of the ten classes, the SimCLR embedding can achieve nearly 50% accuracy whereas the VAE embedding achieves 12% accuracy. With only 60 labels, the SimCLR accuracy has surpassed 90% and with over 200 labels, the accuracy is nearly optimal at around 98%. Comparatively, the VAE embedding keeps learning up to around 300 labels and achieves approximately 94% accuracy.

5. CONCLUSION

As demonstrated in the results section 4, the power of contrastive learning for feature extraction serves to be a useful tool in the space of SAR data, yielding more linear separability of classes and better partitioned

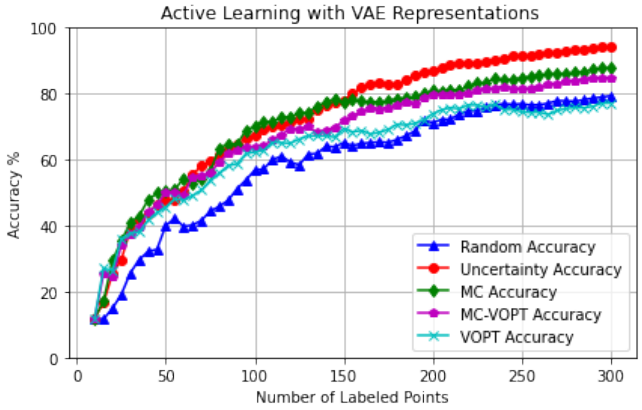


Figure 6. This plot represents the accuracy of active learning with Laplace semi-supervised learning on the VAE embeddings with the pretrained weights from Miller et al.[1] With 300 labels, the highest accuracy achieved is 94.2%.

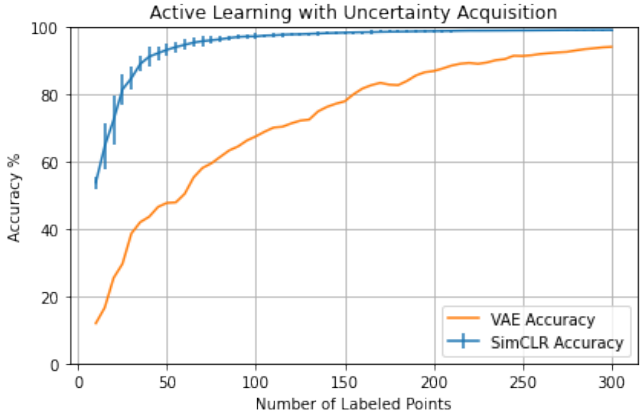


Figure 7. This plot offers a direct comparison between the graph based active learning performance with the SimCLR embeddings against the VAE embeddings. The SimCLR embeddings are trained to 1000 epochs and averaged over 21 distinctly trained models, with the vertical bands corresponding to 1 standard deviation in accuracy.

embeddings, with greater local homogeny and path connectedness. Combined with graph-based active learning, very few labels are necessary to achieve remarkable accuracy using the SimCLR embeddings - SOTA classification accuracies happen with far less labeled data required, compared to the VAE embeddings. One particular interest for future work is that the SimCLR framework is amenable to fine-tuning the encoder network over labeled data[33] in such a way that additional labeled data could lead to stronger embeddings, as well as strong machine learning models. This inspires interesting problems involving updating the encoder network inside the active learning loop, either via an encoder update step or even a novel acquisition function.

ACKNOWLEDGMENTS

We thank Dr. Jocelyn Chanussot for helpful suggestions at the inception of this work. 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-2023-00877. 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. RO was supported by NSF grants GRFP-2237827 and NSF DMS-1944925.

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], Zelnio, E. and Garber, F. D., eds., **12095**, 120950C, International Society for Optics and Photonics, SPIE (2022).

[2] Calder, J., Cook, B., Thorpe, M., and Slepčev, D., "Poisson Learning: Graph based semi-supervised learning at very low label rates," *Proceedings of the 37th International Conference on Machine Learning, PMLR* **119**, 1306–1316 (2020).

[3] Miller, K. and Calder, J., "Poisson reweighted Laplacian uncertainty sampling for graph-based active learning," *arxiv:2210.15786* (2022).

[4] 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).

[5] 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).

[6] 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).

[7] Calder, J., Cook, B., Thorpe, M., Slepčev, D., Zhang, Y., and Ke, S., "Graph-based semi-supervised learning with poisson equations," *In preparation* (2022).

[8] Settles, B., [*Active Learning*], vol. 6, Morgan & Claypool Publishers LLC (June 2012).

[9] Ji, M. and Han, J., "A variance minimization criterion to active learning on graphs," in [*Artificial Intelligence and Statistics*], 556–564, PMLR (2012).

[10] Ma, Y., Garnett, R., and Schneider, J., "σ-optimality for active learning on gaussian random fields," *Advances in Neural Information Processing Systems* **26** (2013).

[11] Miller, K. and Bertozzi, A. L., "Model-change active learning in graph-based semi-supervised learning," (Oct. 2021). arXiv: 2110.07739.

[12] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G., "A simple framework for contrastive learning of visual representations," *Proc. International conference on machine learning (ICML)* , 1597–1607, PMLR (2020).

[13] Koets, M. A. and Moses, R. L., "Feature extraction using attributed scattering center models on sar imagery," in [*Algorithms for Synthetic Aperture Radar Imagery VI*], **3721**, 104–115, SPIE (1999).

[14] Potter, L. C. and Moses, R. L., "Attributed scattering centers for SAR ATR," *IEEE Transactions on image processing* **6**(1), 79–91 (1997).

[15] Zhao, Q. and Principe, J. C., "Support vector machines for SAR automatic target recognition," *IEEE Transactions on Aerospace and Electronic Systems* **37**(2), 643–654 (2001).

[16] Wang, Y., Han, P., Lu, X., Wu, R., and Huang, J., "The performance comparison of adaboost and svm applied to SAR ATR," in [*2006 CIE international conference on radar*], 1–4, IEEE (2006).

[17] Dong, G. and Kuang, G., "Target recognition in SAR images via classification on riemannian manifolds," *IEEE Geoscience and Remote Sensing Letters* **12**(1), 199–203 (2014).

[18] 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 (2017).

[19] 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).

[20] 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).

[21] 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).

[22] 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).

[23] Lewis, B., Scarnati, T., Sudkamp, E., Nehrbass, J., Rosencrantz, S., and Zelnio, E., "A SAR dataset for ATR development: the synthetic and measured paired labeled experiment (SAMPLE)," in [*Algorithms for Synthetic Aperture Radar Imagery XXVI*], Zelnio, E. and Garber, F. D., eds., **10987**, 109870H, International Society for Optics and Photonics, SPIE (2019).

[24] Scarnati, T. and Lewis, B., "A deep learning approach to the synthetic and measured paired and labeled experiment (sample) challenge problem," in [*Algorithms for Synthetic Aperture Radar Imagery XXVI*], **10987**, 29–38, SPIE (2019).

[25] Inkawhich, N., Inkawhich, M. J., Davis, E. K., Majumder, U. K., Tripp, E., Capraro, C., and Chen, Y., "Bridging a gap in SAR-ATR: Training on fully synthetic and testing on measured data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **14**, 2942–2955 (2021).

[26] Swan, M., Major, A., Lear, J., Parks, C. G., and Zhan, J., "Enforcing feature correlation on cycle-consistent gan generated functions: a first step in closing the synthetic measured gap found in sar images," in [*Algorithms for Synthetic Aperture Radar Imagery XXIX*], **12095**, 115–125, SPIE (2022).

[27] Kingma, D. P. and Welling, M., "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114* (2013).

[28] Kingma, D. P. and Welling, M., "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning* **12**, 307–392 (Nov. 2019). Publisher: Now Publishers, Inc.

[29] Hadsell, R., Chopra, S., and LeCun, Y., "Dimensionality reduction by learning an invariant mapping," in [*2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*], **2**, 1735–1742 (2006).

[30] Ji, M. and Han, J., "A variance minimization criterion to active learning on graphs," in [*Artificial Intelligence and Statistics*], 556–564 (Mar. 2012).

[31] 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.

[32] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 770–778 (2016).

[33] Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E., "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems* **33**, 22243–22255 (2020).

[34] Calder, J., "GraphLearning Python Package," `doi:10.5281/zenodo.5850940` (2022).

[35] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D., "Supervised contrastive learning," *Advances in Neural Information Processing Systems* **33**, 18661–18673 (2020).

[36] McInnes, L., Healy, J., and Melville, J., "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426* (2018).

[37] Van der Maaten, L. and Hinton, G., "Visualizing data using t-SNE," *Journal of machine learning research* **9**(11) (2008).

[38] von Luxburg, U., "A tutorial on spectral clustering.," *Stat. Comput.* **17**(4), 395–416 (2007).

[39] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," in [*Avances in Neural Information Processing Systems*], 849–856, MIT Press (2001).