Novel Batch Active Learning Approach and Its Application to Synthetic Aperture Radar Datasets

James Chapman[1, a], Bohan Chen[1, a], Zheng Tan[2, a],
Jeff Calder[3, b], Kevin Miller[3, a], and Andrea L. Bertozzi[3, a]

[a]University of California, Los Angeles, Department of Mathematics, 520 Portola Plaza, Los Angeles, CA 90095, USA

[b]University of Minnesota, School of Mathematics, 538 Vincent Hall, 206 Church Street SE, Minneapolis, MN 55455, USA

1. ABSTRACT

Active learning improves the performance of machine learning methods by judiciously selecting a limited number of unlabeled data points to query for labels, with the aim of maximally improving the underlying classifiers performance. Recent gains have been made using sequential active learning for synthetic aperture radar (SAR) data.[1] In each iteration, sequential active learning selects a query set of size one while batch active learning selects a query set of multiple datapoints. While batch active learning methods exhibit greater efficiency, the challenge lies in maintaining model accuracy relative to sequential active learning methods. We developed a novel, two-part approach for batch active learning: Dijkstra's Annulus Core-Set (DAC) for core-set generation and LocalMax for batch sampling. The batch active learning process that combines DAC and LocalMax achieves nearly identical accuracy as sequential active learning but is more efficient, proportional to the batch size. As an application, a pipeline is built based on transfer learning feature embedding, graph learning, DAC, and LocalMax to classify the FUSAR-Ship and OpenSARShip datasets. Our pipeline outperforms the state-of-the-art CNN-based methods.*

2. INTRODUCTION

Synthetic Aperture Radar (SAR) is a valuable tool in Automatic Target Recognition (ATR). SAR imaging uses a moving device, which repeatedly transmits and receives radio signals, to simulate a large radar dish and achieve high resolution images. Multiple standard SAR data sets have been established as benchmark data sets within the SAR community. For example, MSTAR contains SAR images of land based vehicles,[2] whereas OpenSARShip and FUSAR-Ship contain SAR images of different types of ships at sea.[3,4] The main problem here is to identify objects from SAR images. Although the images are of higher resolution, they are difficult to classify via human inspection, often requiring lots of time from domain experts. This motivates the use of machine learning to improve speed and accuracy for object classification in SAR images.

Due to its recent success in SAR classification and it's high data efficiency, we propose to use semi-supervised learning (SSL) for SAR classification.[1] Algorithms in this domain exploit the geometric structure of the entire dataset in addition to the small amount of label information. In particular, we use graph-based Laplace learning as the underlying classifier. In order to exploit the geometry of the SAR data, the methods must first obtain useful features from the

*Source Code: https://github.com/chapman20j/SAR_BAL

Please direct all correspondence to James Chapman: chapman20j@math.ucla.edu

images for making predictions. Convolutional neural network (CNN) architectures have been highly successful on image classification tasks as they learn rich, structured representations of image data. They are also equipped to deal with the large amounts of noise contained in SAR images. Past works have successfully used Convolutional Variational Auto-Encoders (CNNVAE) to embed SAR data into a lower dimensional embedding space.[1] Transfer learning presents another promising method for obtaining useful image features.[5–7] This involves training a neural network on similar data for which labels are much more readily available. Later convolutional layers in the neural network contain image features useful for the prior task and may be used for SAR classification. These features can be used immediately, or may be fine tuned to get more task specific features after replacing the fully connected layers by linear layers and training on the current dataset. We employ CNNVAE, transfer learning without fine tuning and transfer learning with fine tuning in this paper.

While SSL methods perform relatively well with few labeled data, their performance drastically improves when combined with active learning.[8] Active learning supports the machine learning process by judiciously selecting a small number of unlabeled datapoints to query for labels, with the aim of maximally improving the underlying classifiers performance.[9] Active learning has been shown to significantly improve classifier performance at very low label rates and minimize the cost of labeling data by domain experts.[1,9–11] Central to active learning is the development of an acquisition function. This function quantifies the benefit of acquiring the label of each datapoint in the candidate set $\mathcal{C}$ (a subset of the unlabeled datapoints). Based on the acquisition function, the active learning process selects a query set $\mathcal{Q} \subset \mathcal{C}$ to label. Sequential active learning selects the maximizer of the acquisition function

$$k^* = \underset{k \in \mathcal{C}}{\arg\max}\, \mathcal{A}(k),$$

where "sequential" refers to the case in which the query set has size one (i.e. $|\mathcal{Q}| = 1$). Batch active learning was developed to select a non-singleton query set in each step of the active learning process (i.e. $|\mathcal{Q}| > 1$). Batch active learning is more challenging than the sequential case as datapoints which maximize sequential acquisition functions usually contain similar information. Merely mimicking the sequential scheme to choose the top $|\mathcal{Q}|$ maximizers of the acquisition function $\mathcal{A}(k)$ is usually not optimal, since these maximizers are often close in the embedding space. Batch active learning methods remedy this issue by enforcing diversity in the query set, either by adding restrictions to sequential active learning or by designing objectives specific to batch active learning.[12–17] We opt for the former strategy and build on sequential acquisition functions. Later experiments compare our method, LocalMax, to other active learning methods.

Recent works in ATR fall under two main approaches: supervised learning and SSL. Deep learning has enabled many recent advances as it can obtain useful image features. Zhang, et al. developed a supervised deep learning model called Hog-ShipCLSNet to classify the OpenSAR Ship and FUSAR-Ship data sets.[18] Other notable supervised learning methods use transfer learning from simulated SAR datasets.[6,7] Simulating SAR data reduces the labeling burden on experts, but presents serious challenges due to distribution shift in the datasets.[19] A notable SSL contribution was made in Miller, et al. where the authors applied a CNNVAE and graph-based sequential active learning to classify images in MSTAR.[1] This result achieved state of the art performance on the MSTAR dataset.

In this paper, we improve the feature embedding process on OpenSARShip and FUSAR-Ship using transfer learning from neural networks pre-trained on ImageNet. We also present a novel core-set construction method, DAC, and batch active learning procedure, LocalMax, to decrease the human labeling time by an order of magnitude. This method is compared with other active learning methods, including acquisition-weighted sampling, sequential active learning,

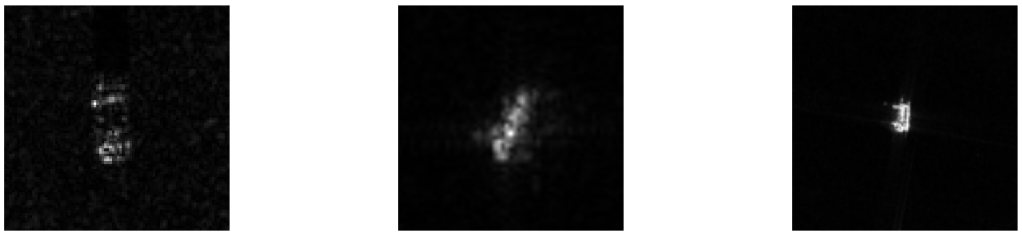


Figure 1 Three samples of SAR images. From left to right, the images show vehicles from MSTAR,[2] OpenSARShip[3] and FUSAR-Ship.[4]

random sampling and naive batch sampling. The combination of transfer learning and batch active learning beats the current state of the art methods on OpenSARShip and FUSAR-Ship, while simultaneously using less training data than state of the art methods.[18] Results are also compared on the MSTAR dataset as a standard benchmark.

3. MATH BACKGROUND

This section reviews some mathematical tools used in our pipeline, including CNNVAE, transfer learning, K-nearest neighbor similarity graph construction, graph Laplace learning and active learning. The pipeline starts by embedding the data using either a CNNVAE or transfer learning. The next step in the pipeline constructs a similarity graph based on the data embedding. Following this, the DAC algorithm computes a core-set. Finally, the active learning loop cycles through: fitting the model with Laplace learning, selecting new points for labeling with LocalMax and querying the human for labels. The full pipeline is shown in Figure 3.

3.1 Data Embeddings

As mentioned in Section 2, our semi-supervised learning methods depend on a good representation of data, where distances between data measure similarity between those data. CNN architectures provide a good way to process image data. We utilize CNNVAEs from previous work[1] and use transfer learning from pre-trained PyTorch CNN neural networks to process the SAR datasets. A convolutional layer near the end of the neural network is designated as the *feature layer* since it captures complex features of the dataset. The outputs of the feature layer are called *feature vectors*; these are used in our subsequent graph construction where graph-based learning will take place. The transfer learning case is shown in Figure 2, where the orange layers denote feature layers.

Transfer learning works by reusing a neural network trained on a similar dataset and task for a new task. One strategy is to immediately use the neural network parameters from a pretrained neural network. Another strategy is to fine tune the parameters of the original neural network with respect to the new dataset. We denote these methods by *zero-shot* transfer learning and *fine-tuned* transfer learning, respectively. Fine-tuned transfer learning is shown with greater detail in Figure 2. Alternatively, CNNVAEs work by first encoding and then decoding the dataset. The neural network architecture is designed so that the encoding is into a lower dimensional space. This forces the neural network to learn useful image features so that it can reconstruct the original image. In this method, the feature layer is chosen as the last CNN layer in the encoder portion of the CNNVAE.

3.2 Graph Construction

As mentioned in Section 2, our methods operate on a graph constructed from the data. Consider the set of d-dimensional feature vectors $X = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^d$. We build a graph $G = (X, W)$, where X is the set of vertices and

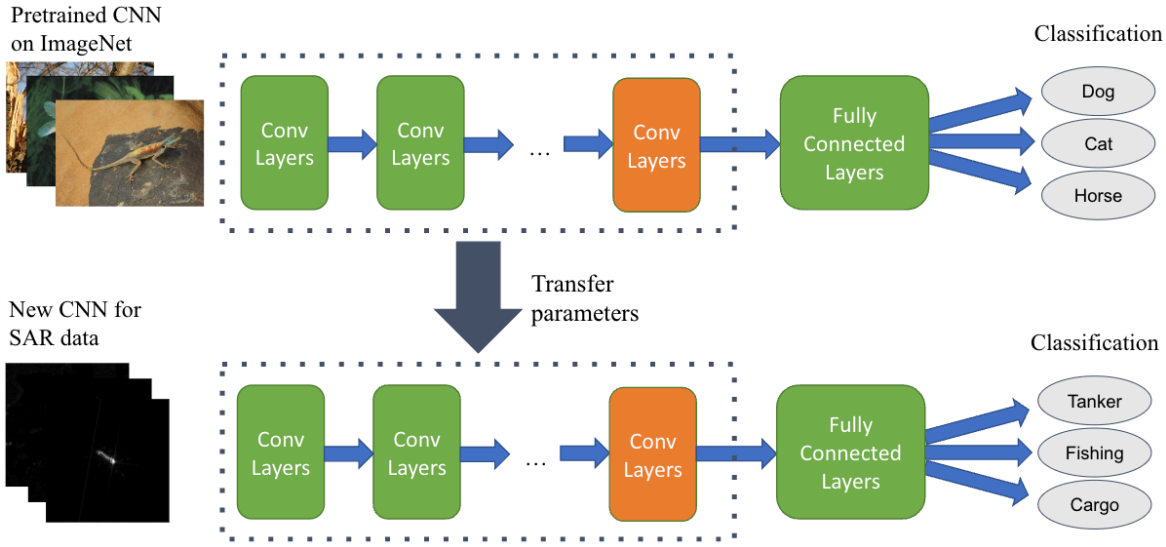


Figure 2 Flowchart for fine-tuned transfer learning. The parameters in the convolutional layers (contained in dotted boxes) of the pretrained CNN are transferred to the new CNN. In fine-tuned transfer learning, all the transferred parameters are trained for a few iterations on the new dataset. The training occurs by first adding new fully connected layers at the end of the neural network and performing supervised learning with the new dataset. When training is complete, only the layers in the dotted boxes are kept for the embedding process. The orange layer denotes the feature layer, and the outputs of this layer provide the feature vectors used later in the pipeline.

$W \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix. The entry W_{ij} denotes the weight on the edge between vertices x_i, x_j, $i \neq j$, which quantifies the similarity between the feature vectors x_i and x_j. In general, the weight matrix is defined by

$$W_{ij} = f\left(\frac{d(x_i,x_j)^2}{\sigma_i\sigma_j}\right), \tag{1}$$

where $d(x_i,x_j)$ is the distance between feature vectors x_i and x_j, $f(\cdot)$ is the kernel function and σ_i,σ_j are normalization constants corresponding to x_i, x_j, respectively. The choice of the distance function d in this paper is the angular distance, i.e.

$$d(x,y) = \arccos\left(\frac{x^\top y}{\|x\|_2\|y\|_2}\right). \tag{2}$$

The normalization constant σ_i associated to node i is chosen according to the distance to the K^{th} nearest neighbor of i, i.e., $\sigma_i = \sqrt{d(x_i,x_{i_K})}$, where x_{i_K} is the K^{th} nearest neighbor to x_i. The kernel function $f(\cdot)$ is the Gaussian (exponential) kernel $f(x) = \exp(-x)$.

In order to improve computational efficiency, we only calculate the weight W_{ij} between nearby pairs of nodes x_i, x_j. For each node x_i, we calculate edge weight W_{ij} between x_i and x_j if and only if x_j is among the K-nearest neighbors (KNN) of x_i in the sense of the angular distance (2). We use an approximate nearest neighbor search algorithm[20] to find the KNN of each node. This results in a KNN weight matrix $\bar{W}$ defined by

$$\bar{W}_{ij} = \begin{cases} W_{ij},\ j = i_1, i_2, \ldots, i_K, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

The parameter K should be chosen to ensure that the corresponding graph G is connected. We symmetrize the sparse

weight matrix to obtain our final weight matrix by redefining $W_{ij} := (\bar{W}_{ij} + \bar{W}_{ji})/2$. Note that W is sparse, symmetric and non-negative (i.e. $W_{ij} \geq 0$). In the following experiments of this paper, we choose the parameter $K = 50$ for the K-nearest neighbor search algorithm.

3.3 Graph Learning

With a graph $G = (X, W)$ constructed as described above, we now describe a graph-based approach for semi-supervised learning and present previous work in this field. We denote the ground truth one-hot encoding mapping by $\mathbf{y}^\dagger : X_L \to \{e_1, e_2, \ldots, e_{n_c}\}$, $\mathbf{y}^\dagger(x_j) = e_{y_j^\dagger}$, where $y_j^\dagger \in \{1, 2, \ldots, n_c\}$ is the ground-truth label of the node x_j and e_k is the k^{th} standard basis vector with all zeros except a 1 at the k^{th} entry. We have information on the ground truth labels for the subset $X_L \subset X$. The goal for the SSL task is to predict the labels of the *unlabeled* nodes $x_i \in X \setminus X_L$. We are interested in solving for a classifier $\hat{\mathbf{u}} : X \to \mathbb{R}^{n_c}$ that reflects the probability of belonging to a certain class. In other words, $\hat{\mathbf{u}}(x_i)$ is a vector whose j^{th} entry reflects the probability that x_i belongs to class j. The graph-based SSL model that we consider obtains a classifier by identifying $\hat{\mathbf{u}}$ with its matrix representation $\hat{U}_{i,j} = \hat{\mathbf{u}}(x_i)_j$ and solving the following optimization problem

$$\hat{U} = \underset{U \in \mathbb{R}^{N \times n_c}}{\operatorname{argmin}} \; J_\ell(U, \mathbf{y}^\dagger) = \underset{U \in \mathbb{R}^{N \times n_c}}{\operatorname{argmin}} \; \frac{1}{2}\langle U, LU \rangle_F + \sum_{j \in Z_0} \ell(\mathbf{u}(j), \mathbf{y}^\dagger(j)), \tag{4}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product for matrices. In equation (4), $L = D - W$ is the unnormalized graph Laplacian matrix,[21] where D is the diagonal matrix with diagonal entries $D_{j,j} = \sum_{k \in Z} W_{jk}$ for $1 \leq j \leq N$.

The first term of (4) optimizes over the smoothness of the classifier, whereas the second term imposes a penalty which ensures that the output of $\mathbf{u}$ at the labeled nodes stays close to the observed labelings $\mathbf{y}^\dagger$. The loss function $\ell : \mathbb{R}^{n_c} \times \mathbb{R}^{n_c} \to \mathbb{R}$ measures the difference between the prediction $\mathbf{u}(x_j)$ and the ground-truth $\mathbf{y}^\dagger(x_j)$ for $x_j \in X_L$. While there are several choices for the loss function, we simply apply a hard-constraint penalty

$$\ell_h(x, y) = \begin{cases} +\infty, \text{ if } x \neq y, \\ 0, \text{ if } x = y. \end{cases} \tag{5}$$

This hard-constraint penalty function ℓ_h forces the minimizer $\hat{U}$ to be exactly the same as the ground-truth $\mathbf{y}^\dagger$ on the observation set X_L. We refer to this SSL scheme as *Laplace learning*.[22] The predicted label of an unlabeled node $x_i \in X \setminus X_L$ of this graph SSL model is given by

$$\hat{y}_i = \underset{k}{\operatorname{argmax}} \, \{\hat{\mathbf{u}}_k(x_i) | \; k = 1, 2, \ldots, n_c\}.$$

3.4 Active Learning

In domains with high labeling costs (ie time and/or money), the accuracy of a machine learning classifier is often constrained by the size of the training set. Active learning was developed to address this issue by providing machine learning algorithms with a way to quantify the value of obtaining labels from data. With a good determination of the labeling value, the machine learning algorithm seeks to maximize the amount of information it gets from each piece of data. This enables models trained using active learning to attain near-optimal performance with a relatively small training/labeled set.[9]

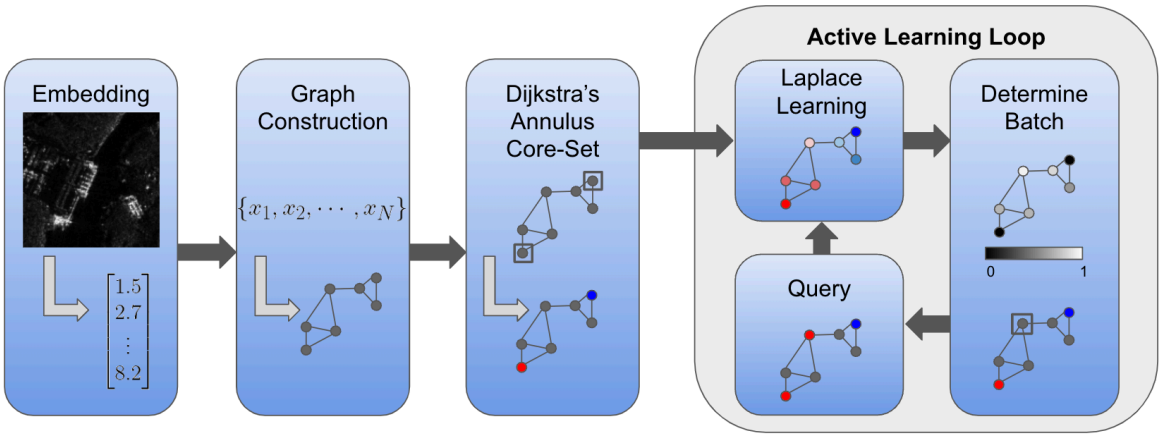


Figure 3 Flowchart of our batch active learning pipeline. First, images are embedded as feature vectors using either transfer learning or a CNNVAE. These feature vectors are then processed to create a weighted graph based on approximate k nearest neighbors. Following this, the core-set is computed using DAC. Lastly, the graph is passed to the active learning loop. The active learning loop includes the classification of unlabeled nodes via Laplace learning, the evaluation of an acquisition function based on the classification and the selection of a query set according to our LocalMax method. After a batch is determined, the active learning loop queries a human to label the selected query set in each iteration manually. The solid red and blue nodes denote classifications of the data, while dark grey nodes denote unlabeled data. The SAR image shown above is from FUSAR-Ship.[4]

Active learning is an iterative process, whereby the model is continually improved through acquiring more labeled data. In each step, given a candidate set $\mathcal{C}$, active learning aims to select a query set $\mathcal{Q} \subset \mathcal{C}$ that the model considers "most helpful" to obtain labels for. Usually, the candidate set $\mathcal{C}$ is the set of current unlabeled nodes at each step. At the core of the active learning process is the acquisition function $\mathcal{A} : \mathcal{C} \to \mathbb{R}$, which quantifies the value of acquiring a label for each unlabeled node in the dataset. We consider four choices for the acquisition function, namely, uncertainty (UC),[23–25] Model-Change (MC),[11,24] Variance Optimality (VOpt)[12] and Model-Change Variance Optimality (MCVOpt) acquisition functions.[1]

To recap, assume we want to classify a certain feature vector set $X = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^d$ through graph Laplace learning and active learning. The active learning process (illustrated in Figure 3) starts with an initial labeled set Y_0. In the k^{th} iteration, denote the current labeled set and current candidate set by $C_k = X \setminus Y_k$. We evaluate the acquisition function $\mathcal{A}_k$ on the candidate set C_k then select the query set $\mathcal{Q}_k \subset C_k$. An oracle then labels the data in the query set $\mathcal{Q}_k$. Lastly, we update $Y_{k+1} = Y_k \cup \mathcal{Q}_k$ and $C_{k+1} = X \setminus Y_{k+1}$ before repeating the process. In practice, the oracle is a human who can provide ground-truth labels for datapoints in the query set.

4. CORE-SET SELECTION AND BATCH ACTIVE LEARNING

As introduced in Section 3.4, active learning is an iterative process which selects a query set $\mathcal{Q}$ to obtain labels in each iteration. Sequential active learning selects one datapoint for each query (the case $|\mathcal{Q}| = 1$) while batch active learning selects multiple datapoints as the query set (the case $|\mathcal{Q}| > 1$). Typically, the human labeling time presents the largest bottleneck in the process. In sequential active learning, the human labeling process is limited by only being able to label one point at a time. It is advantageous for the active learning procedure to return a batch of points so that multiple humans/teams can work in parallel to label the data. Consider the example of labeling 200 points with a team of 10 people. In the sequential case $|\mathcal{Q}| = 1$, labeling requires 200 human queries. In the batch case with $|\mathcal{Q}| = 10$, only

20 human queries are required and the humans can label the data in parallel. This reduces the total labeling time by an order of magnitude as compared to the sequential case.

Many methods rely on using the current predictions of the model to evaluate the uncertainty, variance, or other criteria to quantify the information gained by labeling a new data point. These methods rely on a key assumption: the model has enough information to make a determination of what data would help it learn best. This leads to two interesting requirements: constructing a good set of initial labels, called a *core-set*, for the early stages of active learning and quantifying information shared by unlabeled points in a batch. It is important that we construct a good core-set so that the model can make good estimates of the acquisition function early in training when model accuracy is relatively low. The core-set construction can have a long-term impact on the performance of the active learning procedure and it is critical that the method adequately explore the data before exploiting knowledge with active learning. Fortunately, graph-based active learning methods perform well with relatively small amounts of labeled data.[1]

The main problem with transitioning from the sequential case to the batch case is that sequential active learning methods fail to capture shared information between unlabeled data points. Applying batch active learning naively often leads to batches of very similar points. For example, the k points with the highest acquisition value tend to be close to one another in the embedding space. This leads to a great amount of redundancy in the batch and the model learns as fast as in the sequential case, but with more labeling done at each step. Therefore, batch active learning methods must encourage a diverse selection of points which high acquisition values. Proper implementation of batch active learning must also take into account the time needed to acquire good batches. Optimizing $\mathcal{Q}$ with $|\mathcal{Q}| = B$ over all subsets of the unlabeled data is a combinatorial problem with complexity $O(N^B)$. This time complexity would drastically limit the size of the batches that could be produced, so heuristics should be used to efficiently maximize the acquisition value of batches.

4.1 Core-Set Selection

The goal of core-set selection is to sufficiently explore the data so that the active learning procedure can perform well. We propose Algorithm 1 for core-set selection as it generates a core-set which is nearly uniform in the dataset. For a given feature vector set X, we construct a graph $G = (X, W)$ according to Section 3.2. The algorithm iteratively selects nodes in X to construct a core-set such that all points are a distance at least r from each other but no more than distance R from another point. At each step of the core-set selection process, assuming the currently selected node set is Y (i.e. current labeled set), the algorithm creates an *annular set* C and a *seen set* S:

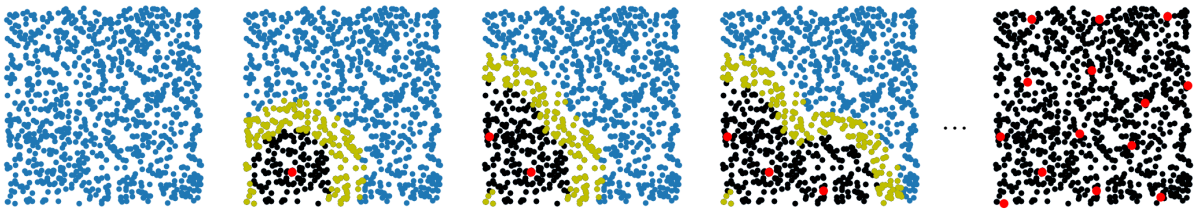
$$C = (\cup_{x\in Y} B_R(x)) \setminus \cup_{x\in Y} B_r(x) \qquad S = \cup_{x\in Y} B_r(x),$$

where

$$B_r(x) = \{y \in V(G) : \ d_G(x, y) < r\}$$

and $d_G(x, y)$ is the distance from x to y computed using Dijkstra's algorithm.[26] The annular set is the set of points that we may select from at each stage in the algorithm and the seen set is a set of points that the algorithm can no longer select from.

Algorithm 1 then randomly selects $x \in C$ and updates Y, S, C. Repeating this process gives a somewhat uniform covering of the data. Note that it may not always be possible to select $x \in C$ before $S = X$. In this case, the algorithm randomly jumps to another data point outside of the seen set. This can occur if R is too small or if there are well-



(a) Iteration 0 (b) Iteration 1 (c) Iteration 2 (d) Iteration 3 (e) Iteration 14

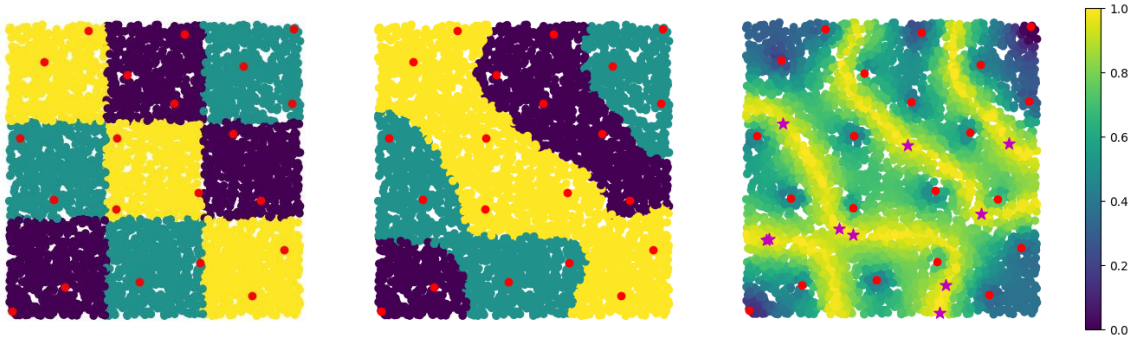
Figure 4 An example of the sampling process of the DAC algorithm with an outer density radius of 0.3. The dataset is generated by sampling uniformly at random in the unit square. The blue, black and gold points denote the unseen points, the seen points and the points in the annular set, respectively. In iteration 0, the annular set is empty and the unseen set isn't empty. This means the algorithm picks a point at random from the unseen points to add to the core-set. In subsequent iterations, the algorithm picks a point at random from the annular set. It then updates the annular region as described in Algorithm 1. This process terminates at iteration 14 when the entire dataset is the seen set. The set of red points in panel (e) is the output DAC core-set, which is nearly uniformly distributed in the whole dataset.

separated clusters in the data. The output of Algorithm 1 is the core-set Y, which serves as the initial labeled set in the active learning process. An example of the algorithm on a simple dataset is shown in Figure 4.

Algorithm 1 Dijkstras Annulus Core-Set (DAC)

Given: Graph $G = (X, W)$, initial labeled set Y, inner radius r and outer radius R
Initialize candidate set $C = \emptyset$ and already seen set S
for each $x \in Y$ **do** ▷ Compute candidate set from already labeled set
  Compute $B_r(x), B_R(x)$
  $S \leftarrow S \cup B_r(x)$
  $C \leftarrow (C \cup B_R(x)) \setminus B_r(x)$
end for
while $S \neq X$ **do**
  if $C = \emptyset$ **then**
    pick $x \in X \setminus S$ uniformly at random
  else if $C \neq \emptyset$ **then**
    pick $x \in C$ uniformly at random
  end if
  Compute $B_r(x)$, $B_R(x)$
  $Y \leftarrow Y \cup \{x\}$
  $S \leftarrow S \cup B_r(x)$
  $C \leftarrow (C \cup B_R(x)) \setminus B_r(x)$
end while
Return Y

Note that this algorithm is also used with r, R adaptively determined by the density of the data around a point. For example, the user could specify that r be picked so that 5% of the data lies in $B_R(x)$. Using the density radius leads to greater exploration in high density regions and less exploration in low density regions. This causes the core-set to focus more on where the majority of the data lies. Additionally, the density based covering is independent of the average distances between data points, which reduces the need for parameter tuning. We further reduce parameter tuning by setting $r = R/2$.



(a) Ground Truth (b) Classifier (c) Acquisition Values

Figure 5 An example of DAC and LocalMax on the checkerboard dataset. In all panels, red points denote the labeled core-set generated by DAC. Panel (a) shows the ground truth classification. Panel (b) shows the classification results of Laplace learning based on the labeled core-set. Panel (c) shows the heatmap of the uncertainty acquisition function evaluated on the dataset. For the uncertainty acquisition function, high acquisition values concentrate near the decision boundary. In panel (c), the purple stars denote points in the query set returned by LocalMax with a batch size of 10.

4.2 LocalMax

We propose a novel batch active learning approach, named LocalMax. Based on a feature vector set X and the corresponding similarity graph G, LocalMax selects a query set of multiple nodes that satisfy the local maximum condition (Definition 4.1) on the graph G from the candidate set. Informally, a node is a local maximum of a function on the nodes if and only if its function value is at least that of its neighbors.

DEFINITION 4.1 (**LOCAL MAX OF A GRAPH NODE FUNCTION**). *Consider a KNN-generated graph $G = (X, W)$, where X is the set of nodes and W is the edge weight matrix. For a graph node function $\mathcal{A} : X \to \mathbb{R}$, $x_i \in X$ is a **local maximum node** if and only if for any x_j adjacent to x_i, $\mathcal{A}(x_i) \geq \mathcal{A}(x_j)$. Equivalently, $x_i \in X$ is a local maximum if and only if:*

$$\mathcal{A}(x_i) \geq \mathcal{A}(x_j), \ \forall j \ s.t. \ W_{ij} > 0. \tag{6}$$

Assuming a batch size B, at iteration k LocalMax selects the query set $\mathcal{Q}_k$ as the top-B local maximums in the candidate set $\mathcal{C}_k$ (as detailed in Algorithm 2). LocalMax benefits from many useful properties including simplicity, efficiency and its grounding in well studied sequential acquisition functions. Building this acquisition function on sequential active learning allows us to borrow properties from the sequential acquisition functions. Controlling for local maxes enforces a minimum pairwise distance between points in the query set, which counteracts the redundancy seen in naively optimizing sequential acquisition functions.

LocalMax also maintains good computational complexity. The computational complexity of Algorithm 2 is $O(KN)$ where N is the number of nodes in the graph and K is the KNN parameter used when generating the graph. In practice, K is much smaller than N, so the computational complexity is $O(N)$. Let $\mathcal{G}(N)$ denote the model fitting time for Laplace learning on a graph G with N nodes and let $\mathcal{H}$ denote the human labeling time for each node. In the case that the weight matrix W is sparse and the graph Laplacian matrix L is well-conditioned, we have $\mathcal{G}(N) = O(N)$. The human labeling time, $\mathcal{H}$, is typically much greater than $O(N)$ since labeling SAR data can take significantly more time than is required by the rest of the active learning pipeline. Since human labeling can be processed in parallel, fewer batches are required

Algorithm 2 LocalMax Batch Active Learning

Require: A KNN graph $G = (X, W)$, X is indexed by Z. A labeled index set Z_0. An acquisition function $\mathcal{A} : Z - Z_0 \to \mathbb{R}^+$. Batch size B.
Ensure: The query set $\mathcal{Q}$.

```
Extend the domain of A to Z by defining A(j) = 0, ∀ j ∈ Z_0
S ← Z − Z_0; Q = ∅
while S ≠ ∅ and |Q| < B do
    k ← arg max_{k∈S} A(k)
    N(k) = {i ∈ Z : W_jk > 0}
    if A(k) ≥ A(i), ∀ i ∈ N(k) then
        Q ← Q ∪ {k}
    end if
    S ← S − N(k)
end while
```

to label the same amount of data. Consider the active learning process that samples in total m nodes to obtain ground-truth labels. The time complexities of sequential active learning and LocalMax batch active learning with batch size B are:

$$\text{Sequential Active Learning:} \quad m \times O(\mathcal{G}(N) + \mathcal{H}) \qquad = O(m\mathcal{H}), \tag{7}$$

$$\text{LocalMax Batch Active Learning:} \quad m/B \times O(\mathcal{G}(N) + O(N) + \mathcal{H}) \qquad = O(m\mathcal{H}/B). \tag{8}$$

Since $\mathcal{G}(N) = O(N)$ and $\mathcal{H} \gg O(N)$, LocalMax provides a B times speed up based on sequential active learning which is observed both theoretically and in practice as shown in Table 2.

5. EXPERIMENTS AND RESULTS

We now present the results of our active learning experiments on the MSTAR, OpenSARShip, and FUSAR-Ship datasets. We first test the accuracy and efficiency of our methods as compared to other batch active learning methods and sequential active learning. We then test the accuracy of our methods with different embedding techniques on each dataset. Lastly, we look at the impact of data augmentation and the choice of neural network architecture in transfer learning. It is important to note that our methods use less data than state-of-the-art methods. LocalMax surpasses state-of-the-art accuracy on OpenSARShip, and FUSAR-Ship with 35% and 68% of the data labeled, respectively. The previous state of the art utilized 44% and 70% of the data for OpenSARShip and FUSAR-Ship, respectively.[18]

In our transfer learning experiments, we use PyTorch CNNs pretrained on the ImageNet dataset to perform image classification on SAR datasets. Unless otherwise stated, we use AlexNet for OpenSARShip and ShuffleNet for FUSAR-Ship since preliminary experiments suggested that they would achieve the best performance among the neural networks tested. The transfer learning results for MSTAR were generally poor and we only present the results with transfer learning from ResNet. These choices are later examined in our comparison between different neural network architectures. Our experiments use both zero-shot and fine-tuned transfer learning. In all the embeddings mentioned, the data is first transformed using the following PyTorch data transformations in order: Resize, CenterCrop, RandomRotation, GaussianBlur, and ColorJitter. Also, LocalMax may not always find batches of the specified size (15), so it selects up to 15 points in a batch. As evidenced by the later efficiency improvements, we see that this occurs infrequently.

Table 1 contains all the parameters used in our experiments. The experiment time measures the time taken to complete the entire active learning process after the core-set selection, including batch selection and model fitting.

The time calculation neglects the human labeling time, so the performance enhancements seen in practice will be much larger. Accuracy is measured as the percent of correct predictions by the model in the unlabeled dataset. The source code to reproduce all the results is available.[27–29] All experiments were performed in Google Colab with high RAM.

Parameter	Value
Batch size	15
Transfer learning data	5%
Sequential Acquisition Function	Uncertainty

(a) General Parameters

	Final Labels (%)	TL Architecture
MSTAR	15%	ResNet
OpenSARShip	35%	AlexNet
FUSAR-Ship	68%	ShuffleNet

(b) Dataset Specific Paramters

Table 1 Tables of parameters used in our experiments. All experiments use these parameters unless otherwise stated. In Table 1a, "transfer learning data" refers to the amount of data used in fine-tuned transfer learning. This data is sampled uniformly at random and is then used as part of the core-set before performing DAC. In Table 1b, "final labels" refers to the size of the labeled dataset as a percent of the total dataset size at the end of the active learning process. Also, "TL architecture" refers to the pretrained PyTorch neural network used for transfer learning on each dataset.

5.1 Accuracy And Efficiency

As seen in Table 2, LocalMax generally outperforms the other batch active learning methods. Among the batch active learning methods tested, LocalMax attained the highest accuracy with comparable efficiency to the other methods. The time efficiency of LocalMax is slightly worse than random sampling, but this is due to the fact that random sampling is a very naive approach which achieves much lower accuracy than LocalMax in each dataset. The TopMax method has comparable time and accuracy to LocalMax, but in each test the accuracy is lower. This is likely due to the fact that TopMax does not enforce separation of the data in a batch and may select points with lots of shared information. Acq Sample also performs worse than LocalMax by a noticeable amount.

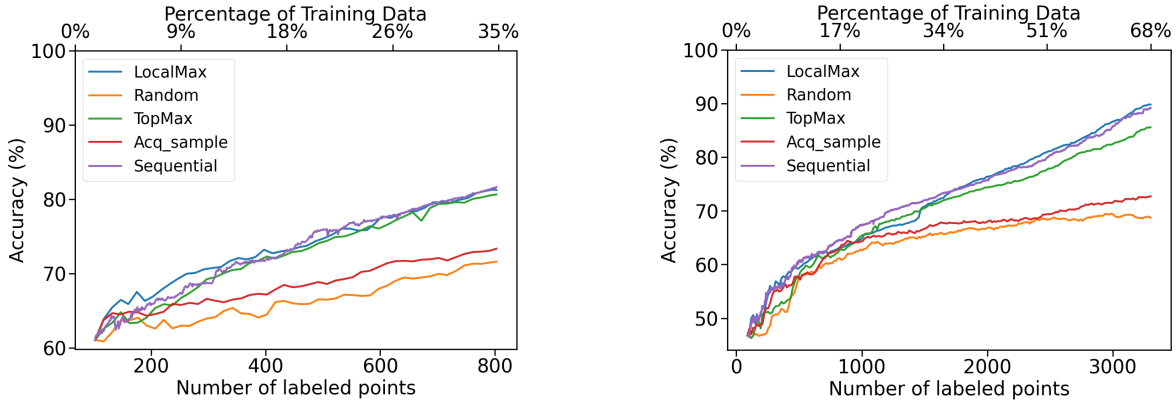
We can see from Table 2 that LocalMax has comparable accuracy to sequential active learning, deviating by at most 0.64% on each dataset. Additionally, LocalMax uses an order of magnitude less computational time than sequential active learning. The discrepency between theoretical efficiency analysis and the experimental time efficiency is due to the fact that human querying time is negligible in these experiments. The code immediately provides ground truth labels when queried, so most of the time in these experiments is observed in the model fit time. In practice, these time differences will better match theoretical predictions where the human query time is the bottleneck. These experiments attest to the efficiency and accuracy of LocalMax relative to other active learning methods.

More detailed plots of the accuracy can be seen in Figure 6. These plots show accuracy as a function of the size of the labeled set. In each case, we see a large jump in accuracy with few labels, which is characteristic of the active learning process. After this point, the accuracy appears to grow linearly with the amount of labeled data. Again, we notice that LocalMax and sequential active learning tend to perform best on both datasets. We can also see that random sampling fails to capture the importance of labeling certain data as its accuracy is much lower throughout the active learning process. This shows that LocalMax is improving the model in a more substantial way than by just increasing the size of the labeled dataset.

Following the comparison between different batch and sequential active learning methods, we analyze the impact of the acquisition function on LocalMax. The results of these experiments are shown in Figure 7, where we see that the uncertainty acquisition function performs best in all experiments. This matches results from previous works on

		LocalMax	*Random*	*TopMax*	*Acq_sample*	*Sequential*
Time Consumption	*MSTAR*	26.70s	**24.17s**	25.96s	26.71s	338.11s
	OpenSARShip	5.33s	**4.68s**	4.86s	4.86s	47.43s
	FUSAR-Ship	26.80s	**21.21s**	26.08s	21.55s	322.99s
Accuracy	*MSTAR*	99.69%	92.66%	99.69%	96.27%	**99.93%**
	OpenSARShip	81.25%	71.60%	80.64%	73.34%	**81.65%**
	FUSAR-Ship	**89.83%**	68.73%	85.59%	72.72%	89.19%

Table 2 Time consumption and accuracy comparison among different active learning methods. This experiment uses a CNNVAE embedding for MSTAR and zero-shot transfer learning for OpenSARShip and FUSAR-Ship. Additionally, the parameters for all the methods are listed in Table 1. LocalMax is the batch sampling method introduced in Section 4.2. Random is a batch active learning method which randomly chooses a new batch with the desired size. TopMax is a batch active learning method which chooses the n points with highest acquisition values. The acq_sample method assigns each point with a probability to be picked proportional to the acquisition value, and randomly samples n points as a batch. All batch active learning methods have comparable efficiency and are 9 to 15 times faster than the sequential case. The local max method always achieved higher accuracy than other batch active learning methods and is comparable to the accuracy of sequential active learning.



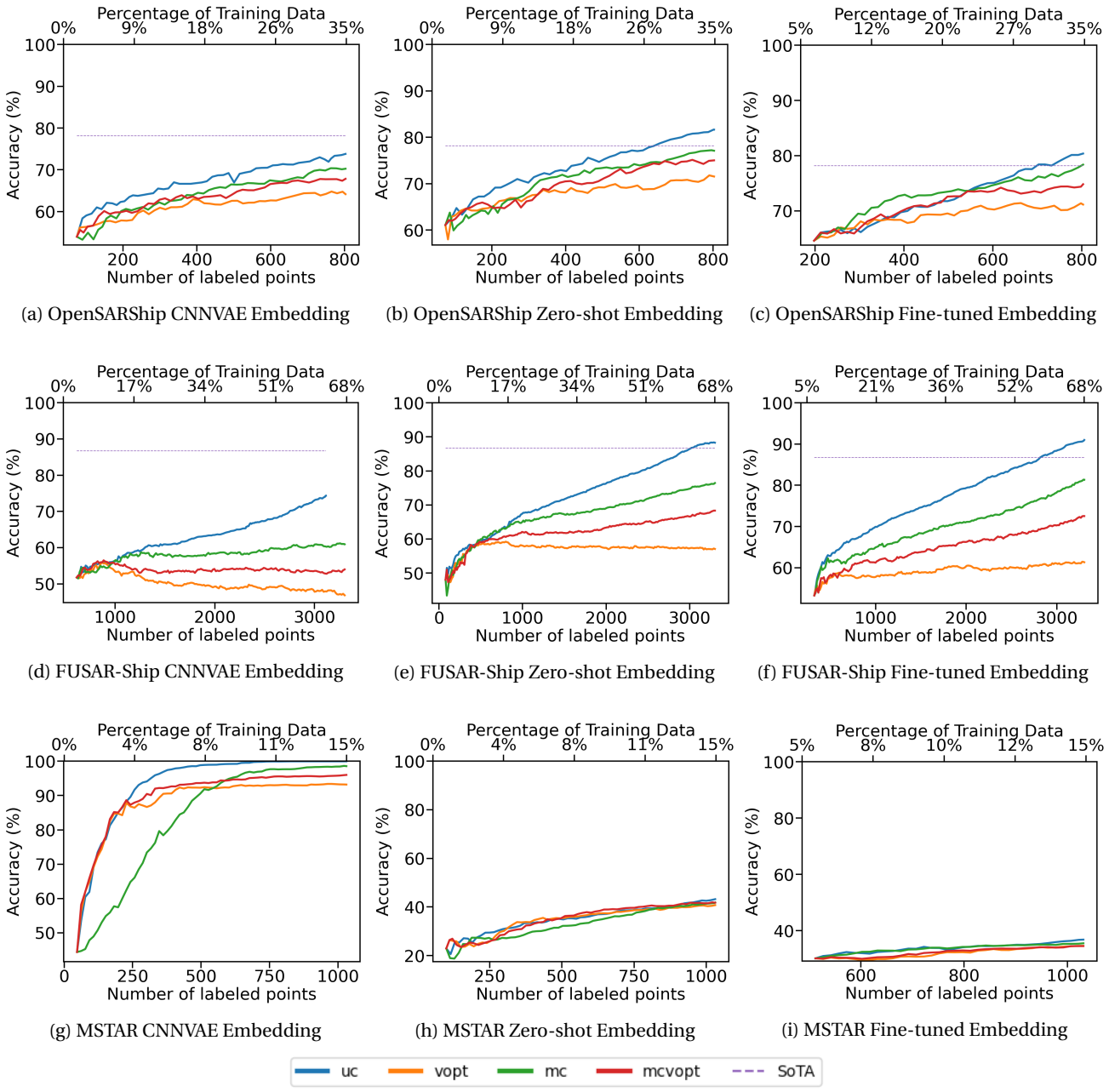
(a) OpenSARShip (b) FUSAR-Ship

Figure 6 Plots of accuracy v.s. the number of labeled points for five different active learning methods. Details about these active learning methods are shown in the caption of Table 2. Panel (a) and (b) contain the results for the OpenSARShip and FUSAR-Ship datasets, respectively. In each panel, our LocalMax method (blue curve) and the sequential active learning (purple curve) are almost identical and are the best performing methods. According to Table 2, LocalMax is much more efficient, proportional to the batch size.

sequential active learning.[1] Figure 7 also shows that the uncertainty based LocalMax beats state of the art in both transfer learning experiments on FUSAR-Ship and OpenSARShip.

5.2 Sensitivity Analysis

We now look at the sensitivity of our results based on choices in the pipeline. We first look at the impacts of data augmentation and fine tuning on the final results. Table 3 contains summary statistics for an experiment regarding the benefits to using data augmentation and fine tuning in the transfer learning portion of the pipeline. The results of this experiment are not conclusive between OpenSARShip and FUSAR-Ship. For OpenSARShip, we see that variance is consistently low for the embeddings and zero-shot transfer learning performed best. In contrast, the FUSAR-Ship results had notably higher accuracy and variance for fine-tuned transfer learning without data augmentation. In fact, adding data augmentation reduced variance for fine-tuned transfer learning on both datasets. It is also important to note that each of these experiments showed better accuracy than the previous state of the art. Additionally, zero-shot



(a) OpenSARShip CNNVAE Embedding (b) OpenSARShip Zero-shot Embedding (c) OpenSARShip Fine-tuned Embedding

(d) FUSAR-Ship CNNVAE Embedding (e) FUSAR-Ship Zero-shot Embedding (f) FUSAR-Ship Fine-tuned Embedding

(g) MSTAR CNNVAE Embedding (h) MSTAR Zero-shot Embedding (i) MSTAR Fine-tuned Embedding

Figure 7 Plots of accuracy v.s. number of labeled points for each embedding and dataset. In each panel, we show four curves generated by LocalMax using different active learning acquisition functions, UC, VOpt, MC and MCVOpt (Section 3.4), together with the state-of-the-art, CNN-based method (denoted SoTA[18]). There are nine panels in total - the Three rows from top to bottom correspond to the OpenSARShip, FUSAR-Ship and MSTAR datasets while three columns from left to right correspond to CNNVAE embedding, zero-shot transfer learning embedding and fine-tuned transfer learning embedding. The UC acquisition function has the best performance among all the acquisition functions tested. The parameters for these experiments are the same as those specified in Table 1.

transfer learning may be the most practical method as it attains comparable performance to the other embeddings, does not require labels in the new dataset, and has no training time.

LocalMax Accuracy with Different Embeddings

	State of the Art	Zero-shot	Fine-tuned	Fine-tuned + Augmentation
OpenSARShip	78.15% ± 0.57%	81.02% ± 0.76%	79.66% ± 0.90%	80.00% ± 0.75%
FUSAR-Ship	86.69% ± 0.47%	88.57% ± 0.35%	91.54% ± 3.07%	89.06% ± 1.90%

Table 3 Sample statistics of accuracy after 20 experiments of the batch active learning pipeline with zero-shot transfer learning, fine-tuned transfer learning, and fine-tuned transfer learning with data augmentation (laTst column). The number in each cell represents the mean ± one standard deviation across the 20 experiments. The zero-shot and fine-tuned embeddings are the same as mentioned in Section 3.1. The parameters in these experiments are the same as those specified in Table 1.

Lastly, we study the impact of neural network architecture on model performance. Table 4 shows the results of one run of LocalMax for different choices of neural network architectures. The range of model performance across architectures is 8.10% for OpenSARShip and 4.82% for FUSAR-Ship. Although this is somewhat large variation, this is a common issue encountered in deep supervised learning. It is important to note that the neural network architectures tested are standard neural networks and weren't designed for SAR data. It is possible that transfer learning from architectures designed for SAR data will experience less variance in the final accuracy.[6,7]

Zero-shot Embedding Active Learning Various CNN

	OpenSARShip	FUSAR-Ship
AlexNet	**80.24%**	85.14%
ResNet18	72.14%	87.39%
ShuffleNet	72.34%	88.22%
DenseNet	75.69%	**89.96%**
GoogLeNet	73.28%	86.74%
MobileNet V2	74.15%	86.68%
ResNeXt	76.29%	88.29%
Wide ResNet	73.14%	85.52%

Table 4 Accuracy values of one run of LocalMax for different choices of neural networks. Each experiment uses zero-shot transfer learning (no fine-tuning) and the parameter values specified in Table 1. The highest accuracy value in each column is bolded. As shown in the table, the range of model performance across architectures is 8.10% and 4.82% for OpenSARShip and FUSAR-Ship, respectively.

6. DISCUSSION

Improving our understanding of SAR data and the methods used for ATR is critical for designing new ATR algorithms. This paper seeks to understand the latter but gives rise to some surprising results the about SAR datasets tested. In particular, it is surprising to find that neural networks trained on ImageNet could perform well when applied to OpenSARShip and FUSAR-Ship. The images from ImageNet are standard pictures of objects and the main problem is to classify them into many different categories which have nothing to do with ships. However, the neural networks trained on ImageNet somehow capture image features that end up being useful for distinguishing different types of ships. Another interesting point is that transfer learning did not perform well with MSTAR. Understanding these differences could be a valuable research direction.

This is related to works on distribution shift in simulated-to-real (sim-to-real) scenarios.[6,7,19] The key challenge

here is to generate many synthetic training examples which reliably capture the testing distribution (true SAR data). We suspect that our methods could be helpful when trained on synthetic data. In particular, we believe that our methods will perform better when the neural networks are specifically designed to handle SAR data and are trained on that data. These specially designed neural networks should better capture symmetries in the data and the image features will likely be more useful for ATR.[18]

7. CONCLUSION

In this work, we developed a novel core-set construction method (DAC) and a batch active learning method, LocalMax. Using these methods together with the embedding techniques (CNNVAE and transfer learning) and Laplace learning, these methods beat state-of-the-art SAR classification methods on OpenSARShip and FUSAR-Ship. Additionally, they have dramatically improved efficiency relative to sequential methods, while maintaining the accuracy of sequential methods. These methods also attain higher accuracy than other common batch active learning methods on the datasets tested.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Geospatial-Intelligence Agency under Award No. HM0476-21-1-0003. Approved for public release, NGA-U-2023-00750. 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. Jeff Calder was also supported by NSF-CCF MoDL+ grant 2212318.

REFERENCES

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

[2] 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.

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

[4] Hou, X., Ao, W., Song, Q., Lai, J., Wang, H., and Xu, F., "Fusar-ship: Building a high-resolution sar-ais matchup dataset of gaofen-3 for ship detection and recognition," *Science China Information Sciences* **63**(4), 1–19 (2020).

[5] Bansal, M., Kumar, M., Sachdeva, M., and Mittal, A., "Transfer learning for image classification using vgg19: Caltech-101 image data set," *Journal of Ambient Intelligence and Humanized Computing* , 1–12 (2021).

[6] Arnold, J. M., Moore, L. J., and Zelnio, E. G., "Blending synthetic and measured data using transfer learning for synthetic aperture radar (sar) target classification," in [*Algorithms for Synthetic Aperture Radar Imagery XXV*], **10647**, 48–57, SPIE (2018).

[7] 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).

[8] Zhu, X. J., "Semi-supervised learning literature survey," (2005).

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

[10] Dasgupta, S., "Two faces of active learning," *Theoretical Computer Science* **412**, 1767–1781 (Apr. 2011).

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

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

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

[14] Cai, W., Zhang, Y., and Zhou, J., "Maximizing expected model change for active learning in regression," in [*2013 IEEE 13th international conference on data mining*], 51–60, IEEE (2013).

[15] Gal, Y., Islam, R., and Ghahramani, Z., "Deep bayesian active learning with image data," in [*International Conference on Machine Learning*], 1183–1192, PMLR (2017).

[16] Kushnir, D. and Venturi, L., "Diffusion-based deep active learning," *arXiv preprint arXiv:2003.10339* (2020).

[17] Zhdanov, F., "Diverse mini-batch active learning," *arXiv preprint arXiv:1901.05954* (2019).

[18] Zhang, T., Zhang, X., Ke, X., Liu, C., Xu, X., Zhan, X., Wang, C., Ahmad, I., Zhou, Y., Pan, D., Li, J., Su, H., Shi, J., and Wei, S., "Hog-shipclsnet: A novel deep learning network with hog feature fusion for sar ship classification," *IEEE Transactions on Geoscience and Remote Sensing* **60**, 1–22 (2022).

[19] 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*], **10987**, 39–54, SPIE (2019).

[20] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y., "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)* **45**(6), 891–923 (1998).

[21] Von Luxburg, U., "A tutorial on spectral clustering," *Statistics and computing* **17**(4), 395–416 (2007).

[22] 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).

[23] Bertozzi, A. L., Luo, X., Stuart, A. M., and Zygalakis, K. C., "Uncertainty quantification in the classification of high dimensional data," *SIAM/ASA J. Uncertainty Quantification* **6**(2), 568–595 (2018).

[24] Miller, K., Li, H., and Bertozzi, A. L., "Efficient graph-based active learning with probit likelihood via Gaussian approximations," in [*ICML Workshop on Experimental Design and Active Learning*], International Conference on Machine Learning (ICML) (July 2020). arXiv: 2007.11126.

[25] Qiao, Y., Shi, C., Wang, C., Li, H., Haberland, M., Luo, X., Stuart, A. M., and Bertozzi, A. L., "Uncertainty quantification for semi-supervised multi-class classification in image processing and ego-motion analysis of body-worn videos," *Electronic Imaging* **2019**(11), 264–1 (2019).

[26] Dijkstra, E. W., "A note on two problems in connexion with graphs:(numerische mathematik, 1 (1959), p 269-271)," (1959).

[27] Calder, J., "Graphlearning python package," (Jan. 2022).

[28] Calder, J., Cook, B., Thorpe, M., and Slepcev, D., "Poisson learning: Graph based semi-supervised learning at very low label rates," in [*International Conference on Machine Learning*], 1306–1316, PMLR (2020).

[29] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research* **12**, 2825–2830 (2011).