UNIVERSITY OF CALIFORNIA

Los Angeles

Improving Image Feature Detection and Classification in Low-Label Regime with Deep and Classical Methods

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Jason Maxwell Brown

2024

ABSTRACT OF THE DISSERTATION

Improving Image Feature Detection and Classification
in Low-Label Regime with Deep and Classical Methods

by

Jason Maxwell Brown
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2024
Professor Andrea L. Bertozzi, Chair

In today's world, data is fundamental to the growth of technology. However, it is very common that vast amounts of clean, well annotated data are unavailable and so there is a desire for methods that can function with small amounts of labeled data. This includes methods that utilize hand-crafted features and can be tuned manually for a dataset without having a variety of samples, as well as machine learning methods that can succeed in a low label rate regime by utilizing mutual information. This thesis discusses work involving hand-crafted features and variational methods for image detection, segmentation, and classification of shape-coded medical testing particles. Specifically, the Canny edge detector, Hough Transforms, and snake active contours are used for the object detection and segmentation problem, which offers a strong alternative to deep learning methods and has theoretical guarantees. Furthermore, this thesis explores neural networks, graph-based learning, and active learning methods that are robust to low label environments with applications in remote sensing and hyperspectral data. These methods offer a natural fusion of deep learning methods with transductive, variational label propagation. Altogether, these thesis offers a survey of image processing as a field and showcases a wide range of ideas that are all applicable for different problems.

The dissertation of Jason Maxwell Brown is approved.

University of California, Los Angeles

2024

To my friends, family, and mentors, who shaped me into the person I am today

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to thank my mentors, Andrea Bertozzi and Jeff Calder, who provided me with invaluable support and guidance throughout my years as a graduate student. I would also like to thank my collaborators, Bohan Chen, Harris Hardiman-Mostow, Adrien Weihs, and Alyssa Arnheim for their assistance and support, as well as Kevin Miller, Bon-Soon Lin, Kyung Ha, Bohyun Kim, and Jacob Moorman for their advice and helpful conversations.

I would like to thank my friends at UCLA, Yotam Yaniv, James Chapman, Michael Johnson, Dominic Yang, Cameron Kissler, Allison Schiffman, and Aiofe O'Brien for getting me out of the apartment and being an absolute pleasure to be around. Thank you also to my long-time friends, Elliot Newman, Brent Giles, Devin Bruce, Michael McHugh, Roman Schmelzer, Josh Brown, Alex Bice, and Kaj Jakobsen for always being around to talk and have fun with. It is truly amazing and special that we've been able to remain close friends for so long.

I would also like to specially thank my mother, father, and brother for always supporting me, raising me, and giving me a home to come back to. Of course, I would also like to especially thank Sara Chilimidos for her support throughout my entire endeavor. From the pandemic to my defense and beyond, you have given me unparalleled support and I treasure the time we have spent together. I love you all and I couldn't have done it without you.

Chapter 2 is a version of [3] and is a joint work with Alyssa Arnheim and Professors Dino Di Carlo and Andrea Bertozzi. Alyssa provided the experimental data that was examined and Dino Di Carlo and Andrea Bertozzi supervised the project. Andrea Bertozzi helped suggest methods and future directions. I personally researched and implemented the algorithms, carried out the experiments, tuned the parameters, authored much of the text, and proved the theoretical results.

Chapter 4 is a version of [4] and is a joint work with Riley O'Neill and professors Jeff Calder and Andrea Bertozzi. Jeff Calder and Andrea Bertozzi proposed and supervised the

project. Riley provided experiments that validated and measured the results. I contributed to the tuning and design of the main algorithm, and I was responsible for much of the writing and the results.

Chapter 5 is a version of [5] and is a joint work with Bohan Chen, Harris Hardiman-Mostow, Adrien Weihs, and professors Andrea Bertozzi and Jocelyn Chanussot. Jocelyn Chanussot proposed the problem and supplied the data and both Jocelyn Chanussot and Andrea Bertozzi supervised the project. Bohan, Harris, Adrien, and I each contributed to data exploration and classification using various methods.

The work done in Chapter 2 is based on research sponsored by Supported by Simons Math + X Investigator Award number 510776 and NSF grant DGE-1829071. The work in Chapter 5 is based on work sponsored by US National Science Foundation grants DMS-2318817, DMS-2152717, DMS-1952339, and DMS-2027277. Both of these chapters contain work published in IEEE.

VITA

2014-2016	Mathematics Tutor, Sierra College.
2014-2016	AS-T (Mathematics) and AS-T (Physics), Sierra College.
2016–2019	Grader and Workshop Leader, California Polytechnic State University, San Luis Obispo.
2019	B.S. (Pure Mathematics) and Minor (Physics), California Polytechnic State University, San Luis Obispo. Awarded Outstanding Graduating Senior.
2019-2024	Teaching Assistant, Mathematics Department, University of California, Los Angeles.
2021-2024	Research Assistant, Mathematics Department, University of California, Los Angeles.
2021	Summer Associate, RAND Corporation.

PUBLICATIONS

J. Brown, A. Arnheim, A. L. Bertozzi, and D. D. Carlo, "Detection and segmentation of shape-coded particles via Hough Transforms and snake active contours," in *2024 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI).* IEEE, 2024, pp. 85–88.

J. Brown, B. Chen, H. Hardiman-Mostow, A. Weihs, A. L. Bertozzi, and J. Chanussot, "Material identification in complex environments: Neural network approaches to hyperspectral image analysis," in *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, 2023, pp. 1–5

J. Brown, R. O'Neill, J. Calder, and A. L. Bertozzi, "Utilizing contrastive learning for graph-based active learning of SAR data," in *Algorithms for Synthetic Aperture Radar Imagery XXX*, vol. 12520. SPIE, 2023, pp. 181–195.

CHAPTER 1

Introduction

In this thesis, we explore semi-supervised learning methods for image processing. These methods are particularly valuable in addressing problems associated with limited training data. In real world applications, acquiring large amounts of well-annotated labels can be time-consuming or is infeasible due to insufficient data. For example, some datasets may contain millions of samples such that labeling a portion of them would be prohibitive for a human to do by hand. Other datasets may require expert knowledge in order to perform the labeling, such as images taken with special cameras or medical images used by specialists. Additionally, some datasets may be limited in scope, creating a bottleneck due to the inability to acquire more data. This could include images of rare events or data that requires significant time and effort to collect. In all of these cases, there are many viable methods for image processing. In this thesis, we broadly examine both hand-crafted and deep learning approaches.

For example, principal component analysis (PCA) and non-negative matrix factorization (NMF) are linear dimensionality reduction methods that have been successful in learning visual features in datasets. Given a collection of n images, $\mathcal{X} \subset \mathbb{R}^m$ where each image has m feature dimensions, principal component analysis finds k orthogonal vectors in $\mathbb{R}^m$ that capture the most variance in the data. The data samples are often stacked into a matrix $X \in \mathbb{R}^{m \times n}$ such that each column corresponds to an image, allowing for further analysis. For PCA, we assume that the rows of X have been centered about 0 by subtracting the mean value of the row across the elements. Principal component analysis has a strong correlation with singular value decomposition and the principal component vectors can be interpreted as the left singular vectors of X. In practice, this is often how they are calculated due to efficient implementation of the singular value decomposition [6]. Principal component analysis has applications that extend far beyond imaging problems, but a notable application has been in the classification of human faces. When trained on a dataset of human faces, the resulting principal components are referred to as eigenfaces and contain information about the primary facial features of humans [7]. Non-negative matrix factorization is also a linear dimensionality reduction technique but instead solves the matrix factorization problem

$$\min_{W,H} \|WH - X\|^2 \tag{1.1}$$

for W and H where $X \in {\mathbb{R}_+}^{m\times n}$, $W \in {\mathbb{R}_+}^{m\times k}$, and $H \in {\mathbb{R}_+}^{k\times n}$. Here $\mathbb{R}_+$ is used to denote the non-negative reals, hence the name non-negative matrix factorization. The matrix W is referred to as the dictionary matrix and the matrix H is referred to as the coefficient matrix, since the columns of (WH) are constructed as linear combinations of the columns of W with coefficients from H. An algorithm was designed by Lee and Seung for efficiently calculating W and H for the problem in Equation 1.1 as well as another divergence loss [8]. Non-negative matrix factorization has applications extending well beyond image processing, but notably has also been used on human facial datasets similar to PCA [9]. For this problem, the enforcement of non-negative dictionary vectors and coefficients means that the learned features are significantly different than PCA, which allows for negative principal components and relies on cancellation. The non-negativity constraint allows it to identify more interpretable features, such as eyes, noses, and lips. Both PCA and NMF are considered unsupervised techniques as they only require the data and no associated labels in order to produce their features.

Signal processing is a distinct field but has many applications and interpretations in image processing. The Fast Fourier Transform (FFT) is an incredibly powerful and significant development that allows for efficient conversion of signals to the frequency domain [10]. This algorithm, when applied to images using the 2D FFT, allows for the interpretation of images in terms of their frequency components. High frequency components reflect fine details in the image and may contain information about textures, for example. Low frequency components vary more slowly in space and therefore contain more broad information about larger structures. Since images are two-dimensional, the orientation of objects within them affects the direction of signals in the frequency domain. The Fourier Transform has many applications for image processing, but a notable one was the development of the discrete cosine transform for image compression [11]. Another technique related to signal processing involves convolving images with task-dependent kernels. Convolving images with kernels, which involves locally adding or differencing pixels, offers a straightforward interpretation. However, the convolution theorem provides another perspective by relating these operations to the image's frequency components. The convolution theorem generally states that convolution in the time domain is equivalent to a pointwise multiplication in the frequency domain. For images, this means that blurring filters, which average local pixel intensity, act as low pass filter in the frequency domain, smoothing out high frequency noise. In fact, natural blurring in images resulting from lens effects are quite common and are observed in the microscopy images in Chapter 2. There has been work using deconvolutions to undo the natural blurring, but it is often difficult to estimate the amount of blurring in natural images [12]. As a counterpart to blurring filters, edge detection filters, such as the Sobel filter, correspond to high pass filters in the frequency domain and remove low frequency regions of relatively constant pixel intensity. The Gabor filters are another notable family of filters that have been used for texture detection [13].

For edge detection, the Canny edge detector is one of the most popular choices [14]. The detection involves multiple steps. First, the image is smoothed with Gaussian blurring. The purpose of the blurring is to suppress noise but not degrade edges, so a specific Gaussian kernel would be chosen on an application basis. Then, the image gradients are calculated typically using methods like Sobel filters, as mentioned previously. Non-max suppression, which is process to remove multiple detections, is applied to filter out spurious edges. At this point, candidate pixels remain but they must be filtered for confidence. For this, two thresholds are introduced: a low and a high threshold. Any pixels with a gradient magnitude larger than the high threshold are high-confident edge pixels and all pixels with gradient magnitude below the low threshold are not considered edge pixels. By a process called hysteresis, all remaining candidate pixels above the low threshold are classified as edge pixels if they are connected to a high-confident edge pixel. This way, the double threshold can filter out isolated noise but preserve edges even if the edge has some regions of lower confidence. We use the Canny edge detector on microscropy images in Chapter 2 to isolate edges for detection purposes. The results of the Canny edge detector are shown in Figure 1.1 where different blurring and threshold values are applied. In Figure 1.1(a), a toy image of an annular square is generated and subject to Gaussian noise and lighting artifacts, which resembles those seen in data from Chapter 2. The image has pixel range between 0 and 1. Figure 1.1(b) shows the results of applying the Canny edge detector with no blurring and threshold values of 0.4 and 0.6 for the low and high thresholds respectively. Figure 1.1(c) shows the effect of the Canny edge detector with $\sigma = 3$ corresponding to the blur value and thresholds of 0.3 and 0.5 respectively. Blurring the image smooths the edges but also affects their intensity, meaning different thresholds need to be chosen.

For local feature detection, there are a plethora of techniques that can be used. Template matching is used for searching for a specific, known template within images [15]. Efficient extensions of template matching include Hough Transforms, for ellipses or lines, and generalized Hough Transforms, discussed in Chapter 2. Other examples of techniques include blob detection with Gaussians, corner detection with the Harris Operator, or feature detection with SIFT (scale-invariant feature transform) [13].

The final family of image processing techniques we provide a brief overview of is variational techniques. Variational methods solve for a function via minimizing an energy function which is often an integral over the function. Typically this will involve using calculus of variations to develop an Euler-Lagrange equation, which, upon solving, produces the minimizer

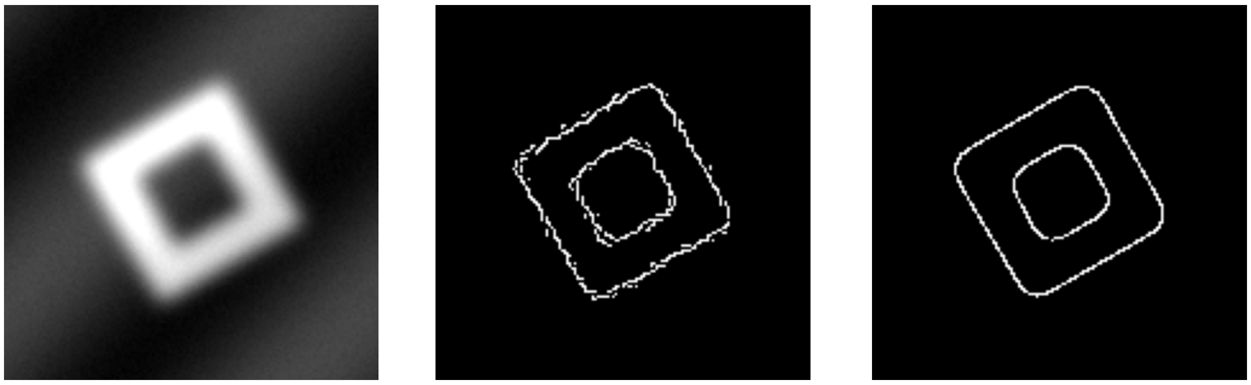


Figure 1.1: Canny edge detector. (a) Generated, noisy image of annular square. (b) Canny edge detector with no smoothing. (c) Canny edge detector with smoothing.

[16]. An example of a variational problem is solving for the brachistochrone curve, which is the curve by which a particle would descend most quickly under gravity. In the sense of image processing, variational methods have seen great success. The Chan-Vese algorithm is a hugely successful algorithm based on the Mumford Shah functional (a piecewise-constant variant). It solves for a functional that best divides an image into two regions such that there is uniformity across the regions. More specifically, they consider a level-set formulation to minimize the following energy function over constants c_1, c_2 and curve C where $\mu, \nu, \lambda_1, \lambda_2$ are fixed hyperparameters and u_0 is the image:

$$\begin{aligned} F\left(c_1, c_2, C\right) = & \mu \cdot \operatorname{Length}(C)+\nu \cdot \operatorname{Area}(\text{ inside }(C)) \\ & +\lambda_1 \int_{\text{inside }(C)}\left|u_0(x, y)-c_1\right|^2 d x d y \\ & +\lambda_2 \int_{\text{outside }(C)}\left|u_0(x, y)-c_2\right|^2 d x d y . \end{aligned} \tag{1.2}$$

We use variational methods in our work, such as snake active contours for image segmentation in Chapter 2 and graph based Laplace learning in Chapters 4 and 5 [17, 18].

The last ten years have seen a skyrocketing improvement in image processing through the modern development of deep learning techniques. These involve neural networks whose efficiency is made possible by advancements in hardware, specifically in graphics processing units (GPUs). Neural network methods caught many researchers' attention when AlexNet beat state of the art classification results on the ImageNet dataset in 2012 using convolutional neural network layers [19]. Since then, convolutional neural networks have been a cornerstone of image processing with many state of the art models seeing various applications. Before we discuss some of these applications, we first introduce neural networks as a machine learning model. Neural Networks are a class of machine learning models that have become prominent in recent years [20]. Typically in machine learning, a model is trained with data to minimize a loss function over that data. The model itself can have varying number of parameters that are to be learned in the process of minimizing the loss. Although there are models that have no learnable parameters, such as k nearest neighbors [21] which does not involve learning parameters through training, most models, including neural networks, will have learnable parameters. Neural Networks are a class of models that are built by sequentially stacking neural network layers on top of each other. These layers represent function operations and can include layers with learnable parameters, such as linear layers and convolutional layers. Other layers, such as ReLU, softmax, and sigmoid layers, do not have learnable parameters and are often called activation functions. The choices of the layer types, number of layers, and the number of parameters in the layers are often tailored to specific tasks or datasets and known as the architecture. Once a neural network is constructed, the parameters need to be learned via the input data and the loss function. To do this, the training data is passed through the neural network to generate an output which is used in the loss function. Via a process called backpropagation, gradients derived from the loss function flow back through the neural network from the final layers to the initial layers. As the layers receive their gradients, the appropriate parameters are updated via gradient descent algorithms. According to the universal approximation theorem, sufficient choices of layers, loss functions, and training data allow neural networks to approximate any function with relatively high degrees of accuracy [22]. Realistically, in order to achieve a well learned model, it is necessary to design a neural network architecture that can appropriately model the function as well as supply it with a sufficient amount of varied training data. Solving this problem while training to minimize the size of the neural network as well as using as few training samples as possible is a challenge that we explore.

An important consideration for neural networks is the amount of labeled data required to train them. Typical convolutional neural networks used for classification will be trained with a cross entropy loss function and full supervision. For the problems we be consider, full supervision is often not practical or possible. Therefore we use unsupervised or semi-supervised methods. Unsupervised machine learning methods are commonly used for learning features about data, such as in dimensionality reduction or clustering. In Chapter 3 we discuss two unsupervised neural network models that are used for learning features from images: the autoencoder and SimCLR. For semi-supervised methods, many deep learning models will incorporate a two term loss function where one term uses the supervised, labeled data and the other term uses the unlabeled data [23]. These methods can be successful but disconnect the labeled and unlabeled data. Transductive learning is a regime in which both the labeled and unlabeled data is used simultaneously to make predictions. Intuitively these approaches are able to leverage relational and geometric information between all of the data, which can produce more holistic predictions. In our work, we discuss graph based Laplace learning which is a transductive, variational, semi-supervised learning model that propagates labels from labeled data to unlabeled data. The construction of a similarity graph between data samples allows for an implicit consideration of relations and geometry in the features. Combining this method with unsupervised neural network feature extraction is a very powerful technique that is explored in Chapters 4 and 5.

In Chapter 2 we discuss traditional image processing techniques for image detection and segmentation. Unlike most machine learning methods, these methods have no learnable parameters. They often utilize hand-crafted features that experts can tune manually for the given task. The benefit of these methods is that if the user is aware of the conditions of the data and there is sufficient uniformity, then very few data samples would be required to tune the models by hand. In particular, Chapter 2 will detail methods for object detection including the circle Hough Transform and generalized Hough Transforms as well as an iterative segmentation technique known as snake active contours. These methods will be reviewed, analyzed, and applied to a small dataset of custom made particles for medical testing. In Chapter 3 we begin our transition away from traditional techniques and towards the deep, machine learning techniques that have seen such prevalence in the recent years. For the sake of image processing in the low label regime, we consider and explore techniques for learning feature embeddings of images in the unsupervised regime. The methods we review in this section are autoencoders in Section 3.2.1 and contrastive learning in Section 3.2.2. Moreover, we also introduce and develop graph-based semi supervised learning as well as active learning in Sections 3.4, 3.5, and 3.6, which together with a trained feature extractor can provide for a strong classification pipeline. In Chapter 4 we examine synthetic aperture radar (SAR) datasets. First we introduce background on SAR data and then introduce two datasets, where one is a simple user modified dataset and the other is a public, benchmark dataset. These are remote sensing datasets and are often difficult to manually label since they require domain specific knowledge. Using the methodology in Chapter 3, we show strong classification performance on both of these datasets including state of the art performance on the benchmark dataset. In Chapter 5 we examine hyperspectral data. We include an introduction to hyperspectral data and many interesting applications in the field. Unique for having detailed spectral information compared to normal images, this data, similar to SAR data, is also difficult to manually label. We utilize similar methodology largely introduced in Chapter 3 but with domain specific considerations in mind. We apply these methods towards hyperspectral pixel classification for the detection of recycling material in waste management. Chapter 6 serves as the conclusion to this body of work and summarizes many of the significant results.

CHAPTER 2

Traditional Image Processing with Applications to Particle Detection and Classification

2.1 Introduction

The work discussed in this chapter is largely adapted from the paper ”Detection and segmentation of shape-coded particles via Hough Transforms and snake active contours”, published in the Southwest Symposium on Image Analysis and Interpretation (SSIAI) in 2024 © 2024 IEEE [3]. Notably, Sections 2.3.4 and 2.3.5 contain new material that is not present in the mentioned work. Additional details are present in the other sections as well. This discussion on the broad topic of traditional image processing techniques focuses on image detection, segmentation, and classification motivated by a particle microscopy dataset. This dataset was made available by Professor Dino Di Carlo and fellow graduate student Alyssa Arnheim and the contributions to the imaging problems were done with assistance of Professor Andrea Bertozzi. My contributions include exploring image processing techniques, tuning methods, writing code, performing experiments, and analyzing convergence properties. The problem is introduced in Section 2.2 and more details on the dataset are provided in Section 2.2.1. A background on related works and techniques is covered in Section 2.3.1. The detection and segmentation of the particles as well as an analysis on the convergence of the segmentation technique is discussed in Section 2.3. The conclusion and results from the work are summarized in Section 2.4.

2.2 Particle on a Chip

To tackle the issue of mass parallel medical testing, special microscopic annular particles have been developed [24]. These particles serve as the solid surface for a particle-based immunoassay, capable of trapping and detecting biomarkers of interest. The particles consist of concentric polymer layers that have a hydrophobic outer layer and hydrophilic inner layer. The inner layer traps the fluid containing the biomarkers and the outer layer prevents mixing with other fluids in the well. The particles are manufactured with a 3D printed device to co-flow four concentric streams of polymer precursors, which are polymerized with UV light. The cross-sectional shapes, such as the squares and circles we examine, are designed with a microfluidic nozzle design in the process [25]. Prior work using these particles reported the detection of a heart failure biomarker across clinically relevant ranges [26]. High fluorescence in the internal cavity of the particle indicates the presence of the biomarker.

Ideally, these particles would be used in multiplexed diagnostics, offering advantages such as reduced reagent usage, enhanced accuracy through swarm-sensing, and parallelization. These particles exhibit the capability to stabilize nanoliter aqueous droplets in the inner annular region, facilitating signal amplification while maintaining independent particle-based reactions. The resulting endpoint fluorescence of these droplets correlates with the biomarker concentration in the patient sample, providing valuable information for disease diagnosis. Due to the intended swarm-sensing nature, it is vital to have an autonomous image processing technique that is able to differentiate between particles in the imaging well. By automatically detecting and separating the particles, researchers will be able to efficiently and reliably determine the amount of fluorescence in each reaction, therefore determining the presence of the desired biomarkers. In order to differentiate the particles within an imaging well with an autonomous method, it is necessary to make the particles visually distinct. We explore a shape distinction, wherein both circular and square shaped particles can be used simultaneously. Each distinct shape can be assigned to different patients or biomarkers, enabling the collection of more comprehensive information in each diagnostic test. For this framework, we explore methods that can detect the distinct particle shapes and, consequently, classify the particle as a circle or a square. Furthermore a segmentation of the particles can be readily acquired. Our approach to this problem is covered in Section 2.3.

2.2.1 Dataset

The dataset consists of images with size of 2048x2048 pixels taken using a Nikon Eclipse Ti2 microscope. The particles are photographed within the well of a 24-well plate. Images are collected at 4x magnification and all bright-field images were captured using phase microscopy to accentuate the features of the particles. The images were taken in both a bright field and dark environments. The bright field images allowed the particles in the imaging well to be clearly observed with their distinct markings while the dark environment showed little features other than fluorescence in particles with detected biomarkers. Together, the bright field images would be used to spatially locate the particles and classify them based on their distinctions, while the dark images are used to measure the fluorescence intensity, which correlates to the amount of biomarker in the particle. For our work, we only use the bright field images. Although it is possible to use the fluorescence in dark images to locate particles, this method would be suitable only for training purposes, as non-fluorescing particles would not be detectable in such conditions.

In this specific dataset, there are many features that make it notable and distinct which influence our design choices in the future. First, it is important to note the significant cost, in terms of both energy and time, required to generate and image batches of these particles. Because of this, the dataset is rather small, with the number of viable particles per class numbering on the order of 50-150 particles. This already indicates that the viability of machine learning techniques will be hampered, as the limited training data will likely cause overfitting. Another notable feature of the dataset is that the particle's shape and size are

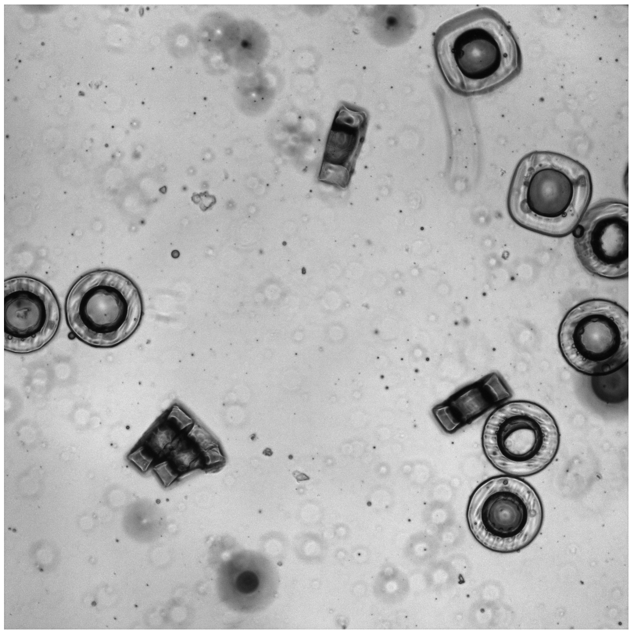


Figure 2.1: Example of an image from a mixed particle well with circular and square particles. Two square particles can be observed in the upper right of the image whereas the rest of the particles are circular. Particles on their side and various artefacts add noise to detection.

generated with a relatively high degree of uniformity. This further supports the choice of using hand-crafted image processing features since we can reliably tune a model using very few samples and be confident that the model will generalize well to unseen data. Unfortunately, the nature of imaging particles in an aqueous solution under variable lighting conditions introduces significant noise in the images. The noise includes overlapping particles, particles that are on their side, liquid condensate, light and dark regions, and significant blurring.

2.3 Detection and Segmentation

2.3.1 Background

A recent, independent work uses deep learning methods for a similar problem, analyzing a dataset with ten different particle shapes, each represented by approximately 102-152 sam-

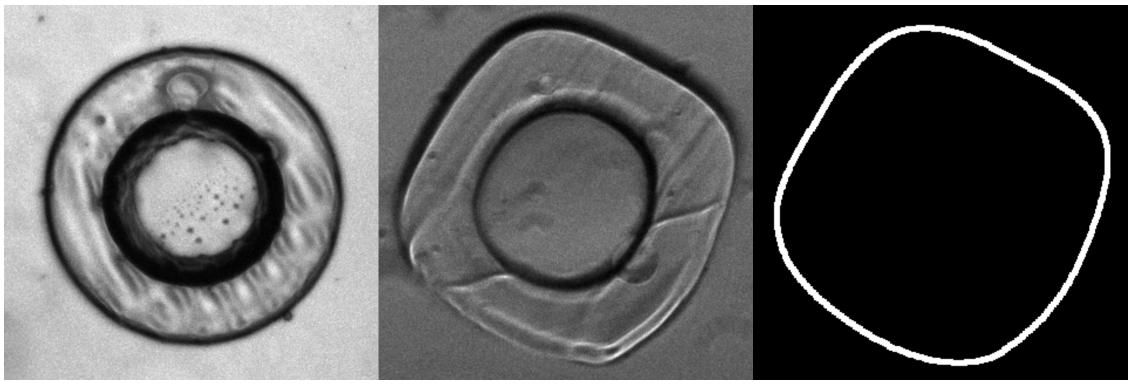


Figure 2.2: Cropped images of particles. (a) Circle particle. (b) Square particle. (c) Square template. © 2024 IEEE

ples [27]. A natural limitation of deep learning is the need for sufficiently large and varied datasets to train on as well as substantial amounts of time and energy dedicated towards annotating the datasets. Deep learning is known to produce variable results when a model trained on one dataset is applied to another, especially under differing collection conditions. Moreover, adversarial attacks on neural network models have shown that classification results can often be sensitive to relatively imperceptible perturbations [28]. For this particular application, the particles can be mass produced and delivered to laboratories that have different configurations for lighting and imaging, making it difficult to develop uniform performance bounds for detection. Instead, we consider a classical approach for identifying circle and square particles, with similar performance to [27]. The benefit of this method is that it can be be calibrated on a small number of images, adapted to many different laboratory environments, and does not depend on deep learning hardware. The proposed method is also constructed in a way that would allow for rigorous theory to be developed for parts of the detection pipeline. We present some examples of such theory later in this chapter.

Our work is, in part, inspired by earlier work on blood cell segmentation. For example, [29, 30] uses a k-means clustering on the pixel values to separate the blood cells from the background followed by the watershed segmentation method. Another method [31] involves the Canny edge detection method and shape geometry. For dispersion control of circular particles [32], the circle Hough Transform has been used and compared with deep methods, as well as the ellipse and line segment detector method [33]. Unlike some of the blood cell datasets, the particles in our dataset are largely transparent (see Figures 2.2 and 2.1), therefore requiring edge based methods. We combine the Canny edge detection [14] method with both the circle Hough Transform [34] and the Merlin-Farber Algorithm [35], which is a form of template matching [15]. After detection, we employ snake active contours [17, 36] to segment the boundaries of the square-shaped particles. The Python code for our method is available on GitHub[1].

2.3.2 Circle Hough Transform

In this section, we provide a brief review of the circle Hough Transform, introduced by Duda and Hart in [34]. The circle Hough Transform provides an efficient way to detect circles of a specified radius, either known or estimated, from within an image. The idea of the method is that if a circle is present in an image, then there will be edge pixels a distance of r from the center of the circle. Thus, for each detected edge pixel in an image, we can track all possible centroids that might correspond to that edge pixel. If there is a circle in the image, then the circle's centroid will be considered a viable candidate for a large number of edge pixels. The process is demonstrated in Figure 2.3 where we see that four points taken from the boundary of a circle all vote on the centroid of the circle.

To implement this process, we first generate an accumulator matrix, A, which tracks potential centroids based on the votes from edge pixels. If we are only interested in detecting circles with a centroid within the image, then we fix the size of A to be that of the original image. Since we are only interested in processing edge pixels, we also use the Canny edge

[1] `https://github.com/jasbrown96/ParticleDetectionSeg`

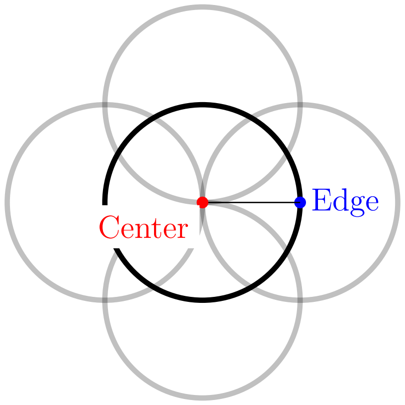


Figure 2.3: Demonstration of the Circles Hough Transform. Selecting four points on the edge, the circle of radius r centered at those points intersects at the true centroid.

detector algorithm to refine our original image down to simply edge pixels. For each detected edge pixel, we increment the count for each entry in the accumulator matrix that is a distance r from the pixel location, ensuring we remain within the matrix boundaries. After this is done, the values in the accumulator matrix will be interpreted as 'votes' for centroid locations, representing the number of edge pixels that were a distance r away. We can then threshold the votes and apply non-max suppression to determine actual centroids in the image. Thresholding will set a criteria for what is considered a centroid detection and non-max suppression is used to select the most confident detection in a local region, eliminating redundant detections to produce a single value.

2.3.3 Template Matching and Hough Transforms

Template matching is a method to detect specific shapes in images. Given some image $I(x, y)$ and a template $T(x, y)$, the simplest and straightforward way to proceed with the template matching involves a cross-correlation of the template with the image

$$(T \star I)(x, y) = \sum_{x_t, y_t} T(x_t, y_t) \cdot I(x_t + x, y_t + y).$$

When the template aligns with the desired shape in the image, centered at some (x^*, y^*), the value of $(T \star I)(x^*, y^*)$ will be large, indicating a strong match [15].

This approach is computationally time-consuming, does not benefit from sparsity in the images, and generally may require many templates to account for shape, size, and orientation. In our case, we are interested in using template matching to detect the square particles. We can leverage some properties of the squares to alleviate the aforementioned downsides of template matching.

First, we may assume that the square particles generally exhibit a high degree of uniformity in terms of shape and size, as well as 90° rotational symmetry. These factors greatly reduce the search space and allow us to use relatively few templates in the process. For the work in this section, we use the template shown in Figure 2.2(c). Our template bank consists of only 10 templates that uniformly span the 90°s of variation, providing uniform coverage of all possible orientations. In other words, we use the template in Figure 2.2(c) as well as 9 other templates that are identical up to rotation. Moreover, the important, distinguishing information in the particles is only in the outer edge, since their interior is largely transparent. Similar to the circle Hough Transform, we first pass the image through a Canny edge detector and then only consider template matching with the detected edges, ensuring sparsity. Considering the sparsity of these images, we use the Merlin-Farber Algorithm for the square particles, which greatly reduces the overhead cost of the cross correlation [35]. To describe the algorithm, we define the edges of the image as

$$E(x,y) = \begin{cases} 1 & \text{(x,y) is an edge} \\ 0 & \text{otherwise} \end{cases}$$

and the binary edge template $T(x,y)$ defined similarly. Let (a,b) represent the reference point for the template, which is typically the center of the template. For each edge in the template, (x^*, y^*), we construct a vector, $r_{(x^*,y^*)} = (a - x^*, b - y^*)$, that points from the edge to the reference location. The collection of these vectors can be used in a similar fashion to the circle Hough Transform, where a given edge pixel can now vote in the accumulator matrix towards a potential reference point.

To accomplish this, we consider the set of vectors that point from edge pixels to the reference pixel to be R. When searching for the template in the edge image, each pixel votes on the possible reference point by using the vectors in R. Figure 2.4 shows this process for a single edge point. In Figure 2.4(a), we start with an arrow-shaped example template with a reference point clearly marked and denoted as "Center". In Figure 2.4(b), an arbitrary edge point is selected and the vector pointing from the edge point to the reference point is shown. In Figure 2.4(c), we act as if we are viewing the edge information in the image. The same vector that pointed from the edge point to the reference point is shown. Moreover, this process gives intuition to the idea that if all of the vectors in R were added to the edge point, the original template, rotated by 180°s, appears centered on the edge pixel. This makes sense as the original template is generated by the vectors pointing from the center to the edge. The flipped template is generated by rotating those vectors 180°s, effectively reversing their direction from pointing to the edge from the center, to pointing to the center from the edge.

To implement the Merlin-Farber Algorithm, we first create an accumulator matrix to keep track of the votes from the edge pixels, similar to the circle Hough Transform. Then we rotate our template by 180°s. This rotated template is used as a mask and added to the accumulator matrix wherever an edge pixel is detected. This process is outlined in (Algorithm 1) and visually shown in Figure 2.5.

For the sake of our work, the Merlin-Farber Algorithm described previously is sufficient. However, within the broader image processing community, the generalized Hough Transform, introduced by Ballard, is often considered superior due to its enhanced efficiency [37]. In the Merlin-Farber Algorithm, the edge pixels vote for the reference point based on all of the vectors collected from the template. In large or high resolution templates, this set of vectors can be quite large and it may require processing a large number of votes.

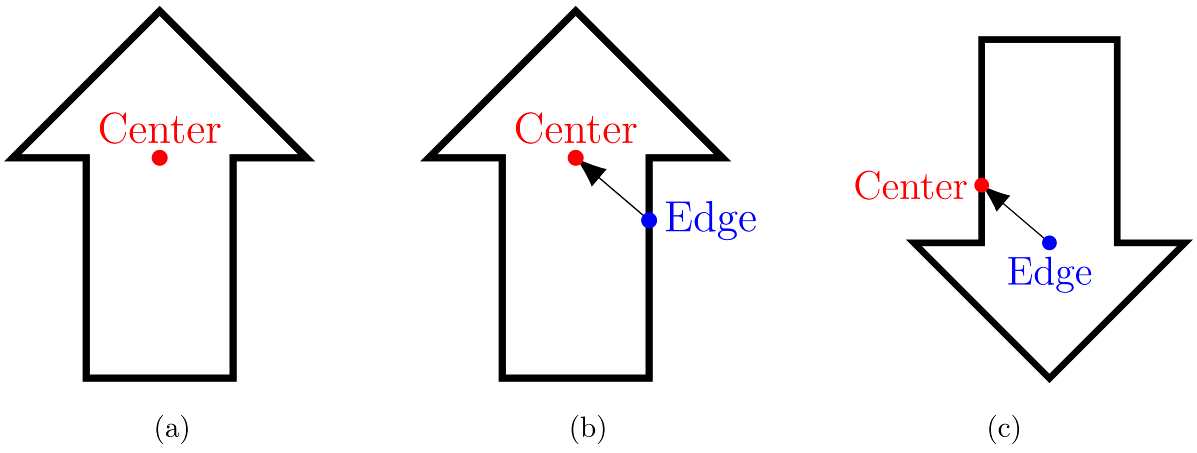


Figure 2.4: Principle of Merlin-Farber Algorithm. (a) An arrow represents an example template and a reference point is marked as "Center". (b) Selecting an arbitrary edge point allows a vector pointing to the reference to be constructed. (c) If the template were rotated 180° and centered on the edge point, then the vector would point to the original center location.

Algorithm 1 Merlin-Farber Algorithm

```
INPUT: The edges, $E \in \{0,1\}^{n_1 \times m_1}$, and a template $T \in \{0,1\}^{n_2 \times m_2}$.
OUTPUT: The accumulator matrix $A \in \mathbb{R}^{n_1 \times m_1}$.
Rotate the template, T, by 180°s.
Initialize empty, padded accumulator matrix A.
For each (i,j) such that E(i,j)=1
    Update accumulator matrix:$A(i-\frac{n_2}{2}:i+\frac{n_2}{2}, j-\frac{m_2}{2}, j+\frac{m_2}{2}) += T$
Remove padding from $A$
```

The generalized Hough Transform reduces the number of vectors used in the voting process, reducing computation. The generalized Hough Transform has an additional step where an R-table is constructed. For each edge pixel in the template, the vector pointing to the reference point and the image gradient at that point are recorded in the R-table. The image gradient is essentially used to keep track of the orientation of the edge pixel with respect to the reference point. In this R table, rotations can be built in by rotating the edge gradient and stored vector respectively. Scaling can be more easily built in by scaling the

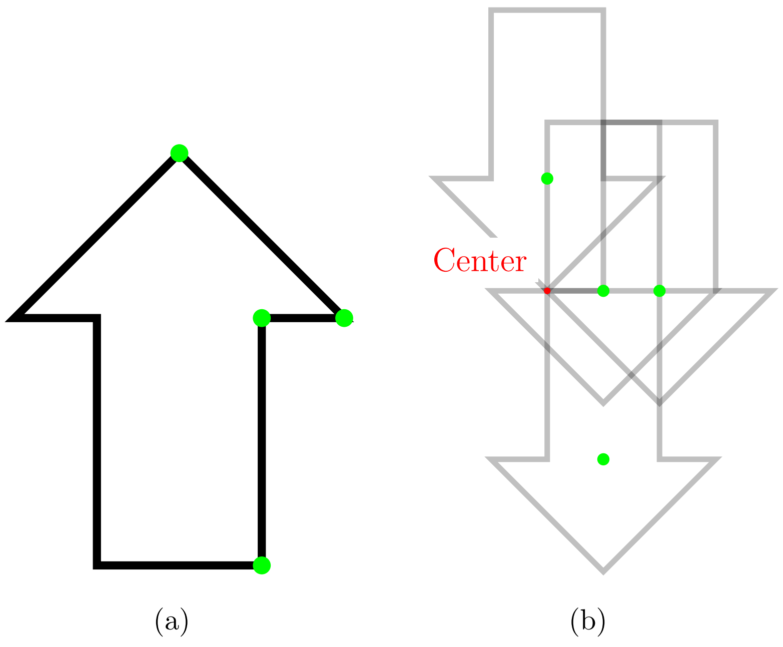


Figure 2.5: The Merlin-Farber Algorithm. (a) A given template is marked with four edge points. (b) By rotating the template 180° and overlaying it, centered on the edge points, the templates all intersect at the center of the arrow. The templates act as the votes from the edge pixels.

size of the vectors in R. In practice, when an edge pixel is detected in an image, its image gradient is matched in the R table and the corresponding vectors that are associated with the edge gradient will be used for voting. By using the image gradients to filter out vectors for the voting process, the voting requires less computations and therefore is quicker and more efficient. Future work could develop this method for a more optimized approach to particle detection.

2.3.4 Segmentation

Once the particles have been detected and the centroids located, the next step is to segment the particles. For the circles, segmentation is quite straightforward since the radius is determined during the detection process, where a range of radii are swept over and checked for best match. Implicitly, a segmentation is found by simply selecting all of the pixels that are a radius's distance away from the located centroid.

For the square particles, more work is required. Although the Merlin-Farber Algorithm does give insight into the boundary of the particle, there is less uniformity in the square particles and the orientations may not line up exactly. For these reasons, we look for a method that offers higher precision and adaptability. We consider the variational segmentation method known as snakes, which are active, edge-based contours that iterate until convergence [17]. Snakes are initialized to some task-dependent curve and then automatically evolve during an energy minimization process. The energy in this variational problem can be interpreted as a cost or penalty on a given snake curve, and that by minimizing the energy or cost, we are locally adapting the snake to achieve desirable properties. In a snake, there are several energy terms, often grouped into internal, external, and image energy terms, shown in Equation 2.1 here [38]

$$E_{\text{snake}} = \int_0^1 E_{\text{intern}}(\mathbf{x})ds + \int_0^1 E_{\text{extern}}(\mathbf{x})ds + \int_0^1 E_{\text{image}}(\mathbf{x})ds. \tag{2.1}$$

The internal energy terms control the stiffness and tension of the snake, which are regularization terms to ensure smoothness and tightness of the curve. The external energy is useful for certain applications where the user of the snake might manually influence the snake and attract or repel it from certain regions. The image energy contains the terms that use the image information and control the attraction to edges, lines, and termination. For our problem, we are not interested in using an external energy as the process is autonomous and does not involve any operator management. For the image energy, we only consider the edge functional although we note that further tuning with the termination energy term could potentially lead to stronger results.

Here we introduce the internal and edge energy terms for a snake contour, $\mathbf{x}(s,t) : [0,1] \times \mathbb{R}^+ \to \mathbb{R}^2$ that depends on a space and time parameter and a given image $I(x,y)$. We use periodic boundary conditions on our snake, ensuring it is a closed curve, to segment the particles. Thus we require that $x(0,t) = x(1,t)$. The following terms represent the internal and edge energy respectively:

$$\begin{aligned}\int_0^1 E_{\text{intern}}(\mathbf{x})ds &= \int_0^1 \underbrace{a(s)\left|\mathbf{x}_s(s)\right|^2}_{\text{Tension}} + \underbrace{\beta(s)\left|\mathbf{x}_{ss}(s)\right|^2}_{\text{Stiffness}} ds \\ \int_0^1 E_{\text{edge}}(\mathbf{x})ds &= -\frac{w}{2}\int_0^1 \left|\frac{\partial I}{\partial \mathbf{x}}\right|^2 ds\end{aligned} \tag{2.2}$$

where $a(s), b(s)$, and w are hyperparameters controlling the tension, stiffness, and edge attraction. Usually $a(s)$ and $b(s)$ are constant, so in the future we just use $a(s) = \frac{\alpha}{2}$ and $b(s) = \frac{\beta}{2}$. When we refer to derivatives with respect to curves, we are referring to the image gradient (or Hessian) evaluated at the curve. In this sense, the edge energy tracks the magnitude of the image gradient along the curve and by minimizing the expression, we are attempting to locally maximize the size of the image gradient along the snake. We choose to exclude the line energy term that attracts the snake to light or dark regions, since the boundary of the square may have both light and dark intensities depending on the direction of the light source and shadows (see Figure 2.2(b)).

The internal energy for the snake controls the length and smoothness of the contour. Since $\left|\mathbf{x}_s(s)\right|^2$ can be interpreted as the square magnitude of the 'velocity', the tension term prevents the snake from stretching too far. The second derivative term $\left|\mathbf{x}_{ss}(s)\right|^2$ can be interpreted as the square magnitude of 'acceleration' and minimizing this term can prevent sharp corners from appearing.

In order to locally minimize the total energy function

$$E(\mathbf{x}) = \int_0^1 \frac{\alpha}{2}\left|\mathbf{x}_s(s)\right|^2 + \frac{\beta}{2}\left|\mathbf{x}_{ss}(s)\right|^2 - \frac{w}{2}\left|\frac{\partial I}{\partial \mathbf{x}}\right|^2 ds \tag{2.3}$$

we use gradient descent. For details on taking the derivative in a calculus of variations sense, we refer to Section 4.2 in [38]. The resulting updates for the gradient steps, with step size of δt, are

$$\delta \mathbf{x} = -\delta t\left(-w\Delta I \nabla I - a\mathbf{x}'' + \beta \mathbf{x}''''\right) \tag{2.4}$$

where ΔI is the Hessian matrix representing the second derivative of the image with respect to the curve and ∇I is the image gradient with respect to the curve. From this interpretation, we can see that convergence will be achieved when the gradient is 0, which corresponds to the Euler-Lagrange equation

$$\underbrace{-w\Delta I\nabla I - a\mathbf{x}'' + \beta\mathbf{x}''''}_{\text{Energy Gradient}} = 0. \tag{2.5}$$

The terms in the Euler-Lagrange equation, as well as those in the gradient descent formula, can be interpreted as energy forces that compel the snake to evolve in a certain direction, guiding it towards a local energy minimization. So a curve that solves the snake problem can be either thought of as locally minimizing an energy function or reaching an equilibrium between the forces, which is the interpretation we use in our analysis [36].

Normally, these snakes, as introduced by Kass [17] and discussed above, will contract and converge on edges. However, proper convergence to the target particle requires the initial curve for the snake to be outside of the particle. This means convergence becomes challenging when particles overlap or are adjacent, introducing additional edges that may attract the snake. To solve this problem, we use balloon snakes, introduced by Cohen [36]. A balloon force on the snake is an additional force term that dictates whether the curve should expand or contract. With this additional force term, we can choose initializations for the snake to be within the particle and allow the snake to expand outwards and converge from within, minimizing the influence of noise. The initialization for the snake is chosen to be a circle centered on the square particle with radius large enough to exclude the inner annuli but small enough to stay within the particle. Due to the uniformity of the particles, this is easily achieved.

The balloon force is expressed as a modification on the image force as seen here

$$F = k\mathbf{n}(s) - k_1\frac{\nabla P}{|\nabla P|} \tag{2.6}$$

where $P = -\left|\frac{\partial I}{\partial \mathbf{x}}\right|^2$ and $\mathbf{n}(s)$ is the normal direction to the curve. The motivation provided by Cohen for this force is that by choosing k to be a positive constant, we are adding a force that expands the closed curve x outwards, ballooning it. In order for convergence

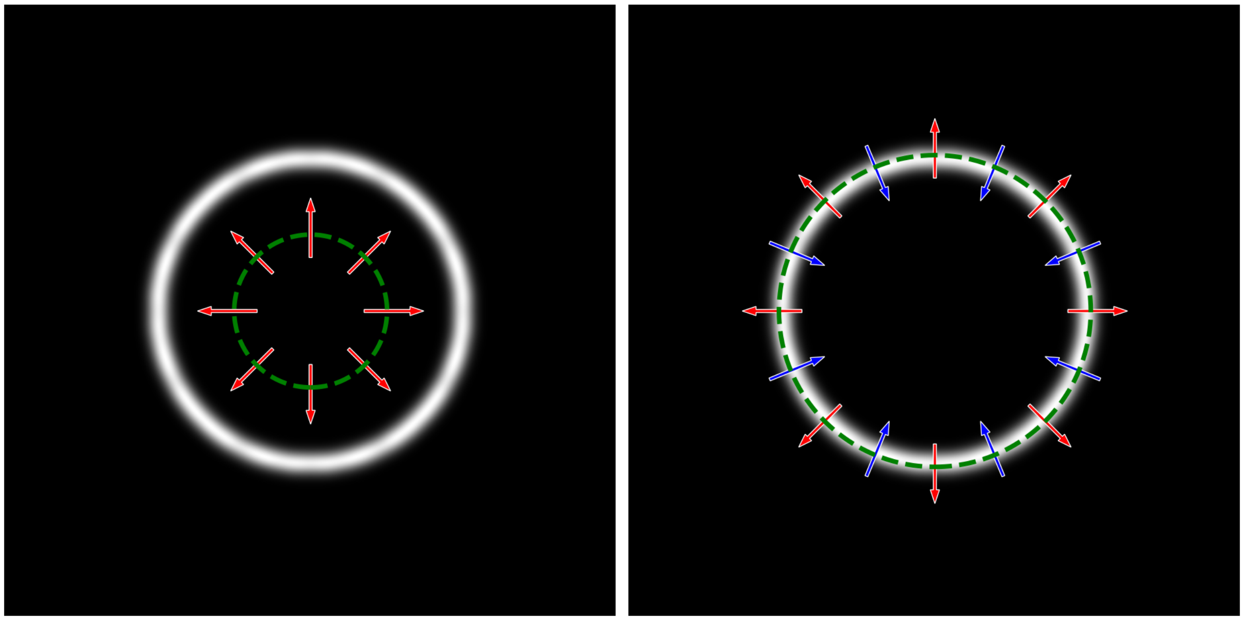


Figure 2.6: A blurred white circle is used as an example to demonstrate the snake forces reaching equilibrium. (a) The initial snake contour is shown in the green dashed line. The balloon force is represented by the red arrows pointing outward, compelling the snake to expand radially. (b) After expanding, the image forces will balance with the balloon forces and convergence will reached just outside the circle.

to be reached, the balloon force will need to eventually reach equilibrium with the image force, so it's recommended to choose k and k_1 to be approximately of the same order with k slightly smaller than k_1 to ensure that the balloon force does not push the snake past all edges. We expect that for a positive balloon force, the snake will converge just outside of the target edge, as the image force will need to be contractive for cancellation. This is visually demonstrated in Figure 2.6. The equilibrium of forces is represented by the following equation, where all forces are balanced and sum to zero:

$$-k\mathbf{n}(s) - k_1 \frac{\Delta I \nabla I}{|\Delta I \nabla I|} - a\mathbf{x}'' + \beta\mathbf{x}'''' = 0. \tag{2.7}$$

Thus far, we have been following Cohen's details regarding the balloon force, but we note that the above expression normalizes the edge force and simply sets it to some constant k_1. This may be undesirable since it provides a constant force towards the edges, regardless of their magnitude. For some applications, this may be desirable, but we instead consider the equation without the normalization on the the edge intensity since we desire that the snake ignores weak edges and clings to the more intense edges we assume will be the particle boundaries. We return the hyperparameter w from Equation 2.3 to replace the normalized image force:

$$-k\mathbf{n}(s) - w\Delta I \nabla I - a\mathbf{x}'' + \beta\mathbf{x}'''' = 0. \tag{2.8}$$

Similar to the method used for the standard snake formulation, we use gradient descent but adapted with the extra balloon force term. Gradient descent would give updates as

$$\delta\mathbf{x} = -\delta t \left(-k\mathbf{n}(s) - w\Delta I \nabla I - \alpha\mathbf{x}'' + \beta\mathbf{x}''''\right). \tag{2.9}$$

The next challenge is in balancing the hyperparameters for the coefficients in the above equation. In a practical sense, the dataset we are working with has variable lighting conditions so the edge intensities will be varied. Notably, we are concerned with balloon forces that are too large, pushing the snake outside the particle, or too small, allowing for convergence on the inner annuli or noise. To solve this, we fix the internal and image energy parameters and use an adaptive search to tine-tune the balloon force for each of the particles, ensuring that the area enclosed by the snake is near the estimated area of the square particles. We know the approximate size of the square particles and can calculate the area enclosed by the snake by using methods such as the shoelace formula or other similar formulas [39]. Simply, if the area enclosed by the snake is too large, we reduce the balloon force and if the area enclosed by the snake is too small, we increase the balloon force until we reach a certain number of iterations or the area is acceptable. We reason that if there is an interval of acceptable balloon force coefficients, k, then a simple binary search would find a value of k in the interval since we would be able to identify k values that are too large or small. To provide some level of justification for the convergence of the balloon snake, we consider a simplified case below and show convergence for the snake under certain conditions.

2.3.5 Convergence of Snake

In this section, we prove conditions on the balloon force that guarantee a snake can converge in a simple case of a circular initialization converging to a circular image. For an image, I, energy parameters w, α, β consistent with Equation 2.3 and a given balloon force k, convergence is reached when equilibrium is obtained in the forces, seen in Equation 2.8. Since there may be a range of balloon force values that give convergence, we frame the proof in such a way as to provide bounds for k. It is worth noting that we do not expect the snake to converge directly to the edge, but rather slightly outside the edge, since the expanding balloon force equalizes with the contracting image and internal forces.

Theorem 1. *Let $I(r) : [0,2] \to [0,1]$ be a C^2, radially symmetric image as a function of the distance from the origin. Let $x : \big[0,1\big] \times \mathbb{R}^+ \to \mathbb{R}^2$ be a snake curve with initialization $x(s,0) = r(\cos(s), \sin(s))$ where $0 < r_{in} < r < r_{out}$. Let $0 < w, \alpha, \beta$ be hyperparameters corresponding to the edge energy, tension, and stiffness terms respectively from equation 2.9. Then the curve $x(s,t)$ will converge to a circle of radius r_{eq} satisfying $r_{in} < r_{eq} < r_{out}$ via gradient descent from 2.9 if the balloon force parameter k satisfies*

$$-wI_rI_{rr}(r_{in}) + (\alpha+\beta)r_{in} < k < -wI_rI_{rr}(r_{out}) + (\alpha+\beta)r_{out}.$$

Proof. First, we provide a brief justification that all of the image forces in the gradient:

$$k\mathbf{n}(s) - w\Delta I \nabla I - \alpha \mathbf{x}'' + \beta \mathbf{x}'''' \tag{2.10}$$

act in the radial direction, where the derivatives on I are with respect to the curve x and the derivatives on x are with respect to the spatial parameter. The image is radially symmetric, so $\Delta I \nabla I$ is only nonzero along $\mathbf{n}(s)$, the outward pointing normal vector. Also, for a circular snake with radius $r(t)$,

$$\mathbf{x}'' = -r(t)\mathbf{n}(s) \quad \text{and} \quad \mathbf{x}'''' = r(t)\mathbf{n}(s). \tag{2.11}$$

Therefore, because of preservation of radial symmetry in gradient descent, we can instead express the equation of the snake as

$$x(s,t) = r(t)\mathbf{n}(s) \tag{2.12}$$

where the time derivative of $r(t)$ is given by

$$r'(t) = -k - wI_rI_{rr}(r(t)) + (\alpha+\beta)r(t). \tag{2.13}$$

This proof now consists of two claims. First, we show that the gradient is negative when $r(t) = r_{in}$, corresponding to the snake expanding, and also the gradient is positive when $r(t) = r_{out}$, corresponding to the snake contracting.

When $r(t) = r_{in}$, we have assumed that $-wI_rI_{rr}(r_{in}) + (\alpha+\beta)r_{in} < k$. So when $r(t) = r_{in}$,

$$r'(t) = -k - wI_rI_{rr}(r_{in}) + (\alpha+\beta)r_{in} < 0. \tag{2.14}$$

Since this gradient is negative, then $r(t)$ will increase under gradient descent and the snake contour will expand.

When $r(t) = r_{out}$, we assumed that $k < -wI_rI_{rr}(r_{out}) + (\alpha+\beta)r_{out}$. So when $r(t) = r_{out}$,

$$r'(t) = -k - wI_rI_{rr}(r_{out}) + (\alpha+\beta)r_{out} > 0. \tag{2.15}$$

Since this gradient is is positive, then $r(t)$ will decrease and the snake contour will contract. Since the snake will expand when $r(t) = r_{in}$, contract when $r(t) = r_{out}$, and is initialized with initial radius r satisfying $r_{in} < r < r_{out}$, then the radius of the snake is trapped between the boundaries. The region between r_{in} and r_{out} is an absorbing set. Since I is a C^2 function, this region would contain some value r_{eq} such that

$$-k - wI_rI_{rr}(r_{eq}) + (\alpha+\beta)r_{eq} = 0. \tag{2.16}$$

With gradient descent minimizing the magnitude of the of the gradient, the snake would converge to such an r_{eq}. $\square$

The first thing to note about this proof is that we assumed that the snake was already initialized within the local well for convergence. In practice, we expect to initialize our snake with a radius smaller than r_{in} and let the balloon force and image energy term draw the snake into the absorbing set. Another important feature to recall is that with the balloon force, we do not expect to converge directly to the edge in the image, as the direct edge of the image would correspond to $I_r I_{rr} = 0$ rather than the condition in Equation 2.16. Below we consider two separate examples which illuminate characteristics of the snake equation.

Example 2.3.1. First, we consider the ordinary differential equation outlined in Equation 2.13, with the assumption that the image term were absent or otherwise $I_r I_{rr} = 0$. This would allow the differential equation to simplify to

$$r'(t) = (\alpha + \beta) r(t) - k. \tag{2.17}$$

If $\alpha + \beta = 0$, then the balloon force is the only remaining force and the snake would expand or contract indefinitely, based on the sign of k. Assuming $\alpha + \beta > 0$, this equation has solutions of the form $r(t) = \frac{k}{\alpha+\beta} + ce^{(\alpha+\beta)t}$ for some constant c. Convergence, however, is reached when $r'(t) = 0$, which occurs when

$$r(t) = \frac{k}{\alpha + \beta}. \tag{2.18}$$

This is the radial value at which the balloon force and the regularization forces come to a balance naturally on their own. Given the radial formulation posed in Theorem 1, this will also result in a circular snake.

If the radius of the initialized snake is not exactly $\frac{k}{\alpha+\beta}$, then the initial condition would result in some nonzero c in Equation 2.17. If $c > 0$, then the radius of the snake is larger than equilibrium and $r'(t) > 0$ would cause gradient descent to contract the snake. Similarly, when $c < 0$, we see that the snake would expand. Thus, whether the snake is initialized with a radius value too small or large, it will converge to the equilibrium radius of $\frac{k}{\alpha+\beta}$.

Example 2.3.2. The second example we consider is a step function blurred by a Gaussian kernel for smoothness. Assume that the radial image $I(r)$, defined by a step function with an edge at $r = 1$ is given by:

$$I(r) = \begin{cases} 0 & \text{if } r < 1 \\ 1 & \text{if } r \geq 1 \end{cases}. \tag{2.19}$$

Applying Gaussian smoothing via convolution will produce

$$J(r) = \int_{-\infty}^{\infty} I(\tau)G(r-\tau)d\tau \tag{2.20}$$

where

$$G(r) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-r^2}{2\sigma^2}} \tag{2.21}$$

is a Gaussian function with σ being the standard deviation. Because I is the step function and G is symmetric, we see that

$$\begin{aligned} J(r) &= \int_{-\infty}^{\infty} I(\tau)G(r-\tau)d\tau \\ &= \int_{1}^{\infty} G(r-\tau)d\tau \\ &= \int_{-\infty}^{r-1} G(\tau)d\tau. \end{aligned} \tag{2.22}$$

With this blurred image, we now apply the fundamental theorem of calculus to get the first and second derivatives

$$J_r(r) = G(r-1) \quad \text{and} \quad J_{rr}(r) = G_r(r-1). \tag{2.23}$$

Therefore, the product of the first and second derivatives is

$$J_r J_{rr}(r) = GG_r(r-1) = \frac{1-r}{2\pi\sigma^4} e^{-\frac{(r-1)^2}{\sigma^2}}. \tag{2.24}$$

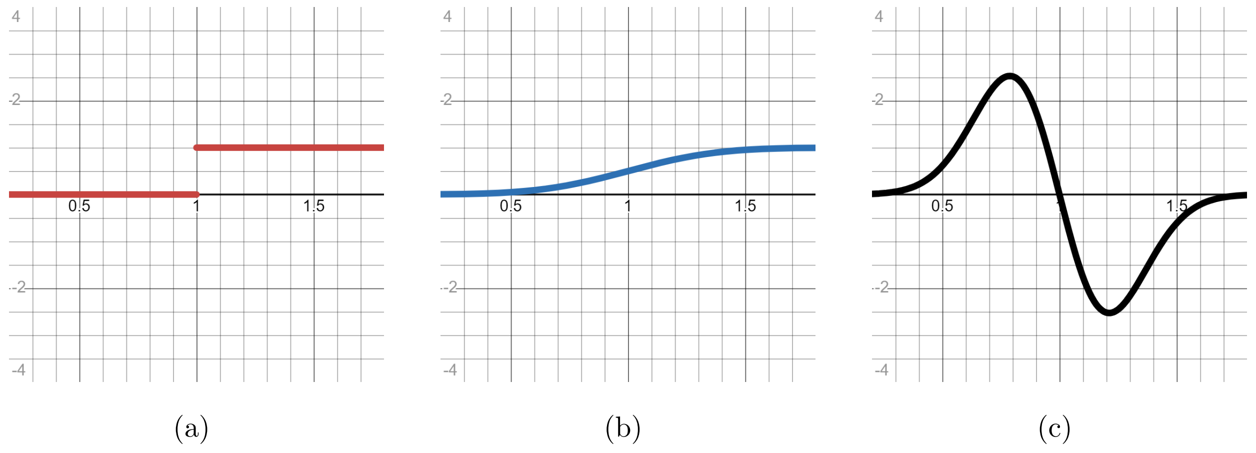


Figure 2.7: Functions from Example 2.3.2. (a) Step function, I. (b) Gaussian smoothed step function, J. (c) Image force term J_rJ_{rr}.

The functions I, J, and J_rJ_{rr} are visually plotted in Figure 2.7. $J_rJ_{rr}(r)$ has a local extrema at $r = 1 \pm \frac{\sqrt{2\sigma^2}}{2}$ with corresponding values

$$J_rJ_{rr}(1 - \frac{\sqrt{2\sigma^2}}{2}) = \frac{\sqrt{2}}{4\pi\sigma^3\sqrt{e}} \quad \text{and} \quad J_rJ_{rr}(1 + \frac{\sqrt{2\sigma^2}}{2}) = \frac{-\sqrt{2}}{4\pi\sigma^3\sqrt{e}}. \tag{2.25}$$

Treating these extremal values as r_{in} and r_{out} from Theorem 1 and assuming $0 < w, \alpha, \beta$ are fixed hyperparameters corresponding to the edge energy, tension, and stiffness terms respectively from equation 2.9, then this region would act as a potential well for any snake with balloon force values, k satisfying

$$-w\frac{\sqrt{2}}{4\pi\sigma^3\sqrt{e}} + (\alpha+\beta)(1 - \frac{\sqrt{2\sigma^2}}{2}) < k < w\frac{\sqrt{2}}{4\pi\sigma^3\sqrt{e}} + (\alpha+\beta)(1 + \frac{\sqrt{2\sigma^2}}{2}). \tag{2.26}$$

2.3.6 Detection Pipeline

The process for detecting and segmenting the circle and square particles is shown in Figure 2.8 and outlined here:

1. Canny Edge Detector: First we apply the Canny edge detector to the images with a pre-chosen low and high threshold that strike a balance between preserving edges and removing noise.

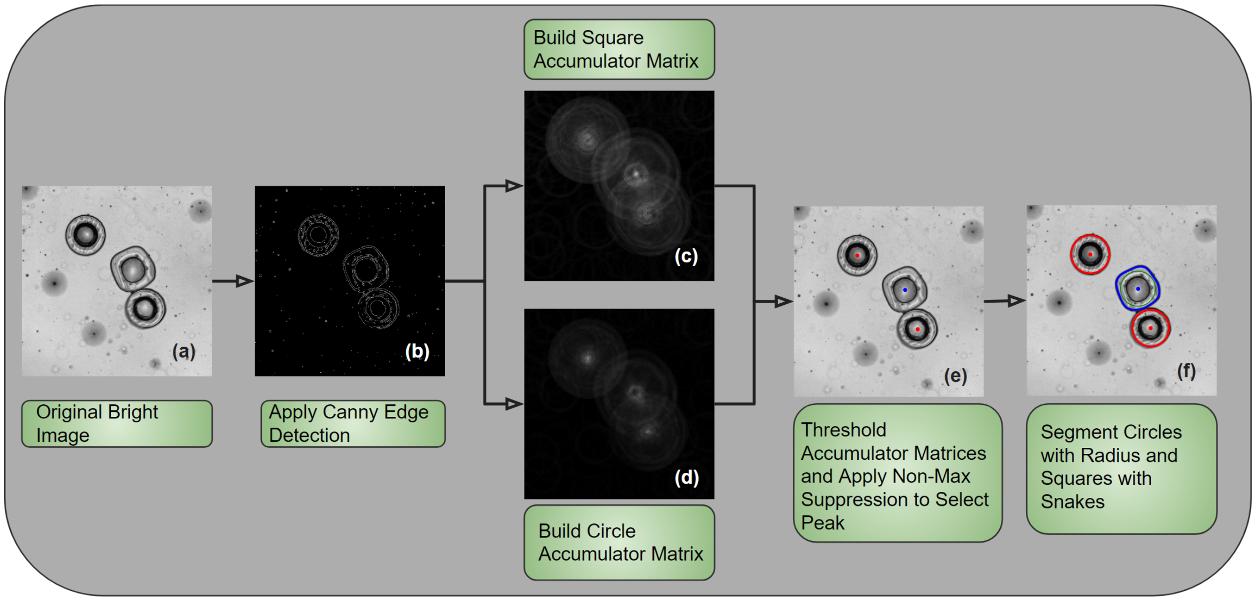

Build Square Accumulator Matrix
Original Bright Image
(a)
Apply Canny Edge Detection
(b)
(c)
(d)
Build Circle Accumulator Matrix
(e)
Threshold Accumulator Matrices and Apply Non-Max Suppression to Select Peak
(f)
Segment Circles with Radius and Squares with Snakes

Figure 2.8: Flowchart for the procedure. (a) The original bright image before any processing has been done. (b) The edges detected by the Canny edge detector. (c) The accumulator matrix from the Merlin-Farber Algorithm. Note the brightest points in each image correspond to the detected centroids for the shapes. The square accumulator matrix has more noise indicated by greater overall brightness. (d) The circle accumulator matrix from the circle Hough Transform. (e) The detected centroids after thresholding the respective matrices and applying non-max suppression. The blue dot corresponds to the square centroid and the red dots correspond to the circle centroids. (f) The segmentation results. The circles are segmented from the knowledge of their radii and the square is segmented with a snake. The initial curve for the snake is shown by the green line and the final contour is shown in blue. © 2024 IEEE

2. Circle Hough Transform: Next, we apply the circle Hough Transform with a predetermined threshold and search over known radiuses for the circles to identify circle centroids and radii.

3. Merlin-Farber Algorithm: Then, we use the template bank for square particles with the Merlin-Farber Algorithm on the edge image to detect the centers of the squares (as detailed in Algorithm 1).

4. Post Process Peaks: To post process, we first apply non-max suppression and enforce that centroids of any detected squares or circles are separated by a fixed distance corresponding to the radius of the particles. Secondly, if a particle is detected as both a square and circle, we default to the circle classification due to greater uniformity of the circle particles and lower error in the circle Hough Transform.

5. Segment: For the squares, we initialize a circular snake about the centroid with a radius large enough to exclude the inner annular region. We use a positive edge attraction and adaptive balloon force to ensure that the area of the snake is within tolerance of the square. For the circles, we use the radii returned from the circle Hough Transform to select the correct pixels.

2.3.7 Experiments and Results

Recall that one of the motivations for choosing the discussed methods, as opposed to deep learning methods, was the limited availability of data. With that in mind, the dataset available with square and circular particles for hyperparameter tuning and testing is relatively limited. We choose 15 images that contain combinations of square and circular particles under various conditions to tune hyperparameters and use five additional images for testing.

Omitting Gaussian blurring prior to the Canny edge detector will cause noise to be sharper and thus more prevalent in the edge detection. Despite this, we prioritize the detection of the particle images and so choose to omit Gaussian blurring. For the Canny edge detector, we use a sigma value of 0, low threshold of 100, and high threshold of 150. Setting Canny thresholds too low can introduce excessive noise, greatly increasing the computational cost for the Merlin-Farber Algorithm and necessitating further tuning of the thresholds for the object detection. A downside of using higher thresholds is that particles in darker regions of the images are less prominent and their edges may be undetected. Failure to detect the edges will lead to a failure to detect with the edge based object detection methods. For the circle Hough Transform, we use the Sci-Kit image library and a threshold of 0.25 [40]. We search over circles with radius between 120 and 130 pixels.

In order to use the Merlin-Farber Algorithm for the square particles, we need to generate various templates that we expect to match with. We use the template seen in Figure 2.2(c) as the guideline and then nine rotated variants of the template with rotation spacing of 10°, utilizing the 90° rotational symmetry in the squares. More templates would improve accuracy but increase computation time. Similarly, decreasing the number of templates would decrease the computational time, but would also decrease the accuracy by potentially missing particles with specific orientations. With these 10 references, we use a threshold of 60 to indicate a detection of a square. In the non-max suppression, we select the greatest peaks and enforce a distance of 150 pixels between peaks, roughly correlating to the radius of the particles. The snakes for the square particles are initialized as circles centered on the detected centroid with a radius of 110 pixels. We use $\alpha = 0.03, \beta = 10, \gamma = 0.001$, and $w = 1.5$ as the snake parameters from Equation 2.3 and begin the balloon force search with $k = 0.003$, where γ is the parameter corresponding to the time step in the gradient descent. The code for the snake active contour is based on the implementation from scikit [40], so the experimental parameters may not have a one-to-one correspondence with their discussion. For example, the internal force appears dependent on the number of nodes used to discretize the snake; increasing the number of nodes will smooth the snake and decrease the magnitude of the derivatives. For these experiments, we use 400 nodes in the snake. The upper and lower limits for the allowed snake area is 58000 and 54500 square pixels respectively. If the area of the converged snake is too small, we increase the value of k until it is too large and then perform a binary search. We similarly modify the value of k if the snake area is initially too small.

We note that in these experiments, particles near or on the boundary were not included for detection and segmentation purposes. In the 20 image dataset, we count that there are 62 reasonably identifiable square particles and 49 reasonably identifiable circle particles. Our method correctly detects and segments 59 of the square particles and 40 of the circle particles.

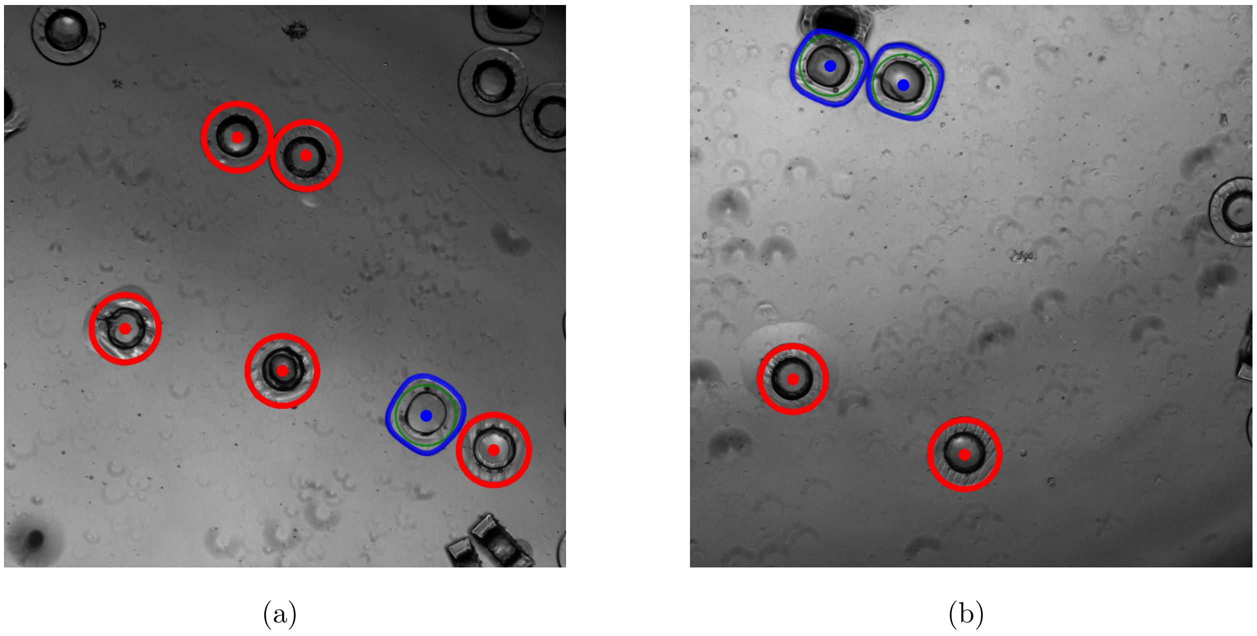


Figure 2.9: Results applied to mixed particle wells that include both squares and circles, showing performance in mixed particle wells. © 2024 IEEE

There are no false positives, which is ideal for medical testing. The false negatives all occurred on darkened regions of the images, likely resulting from a failure in edge detection.

Figures 2.9, 2.10, and 2.11 show an overlay of the detected centroids and segmentations for six notable images that were chosen to showcase various conditions. In Figures 2.9(a) and 2.9(b), the images contain a mixture of the circles and squares demonstrating differentiation between types. Figures 2.10(a) and 2.10(b) showcase non-mixed particle detection from images that have many particles. In these images, particles are adjacent to one another and experience noise effects such as overlapping particles and fluid condensate as well as lighting variations. Figures 2.11(a) and 2.11(b) show particles on the edge of the imaging well that are obscured and thus are very difficult to detect due to low contrast. In summary, although some of the particles are not detected, a large amount of them are despite the presence of noise in the image. Moreover there are no mis-classifications, which indicates a high precision score and is ideal for the application purpose.

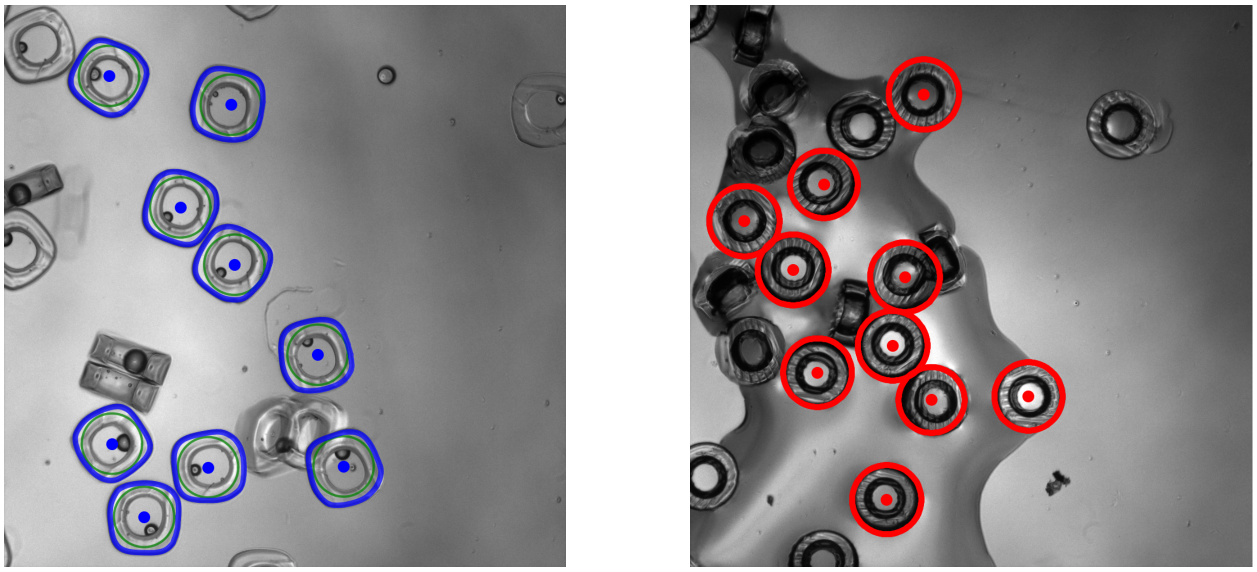
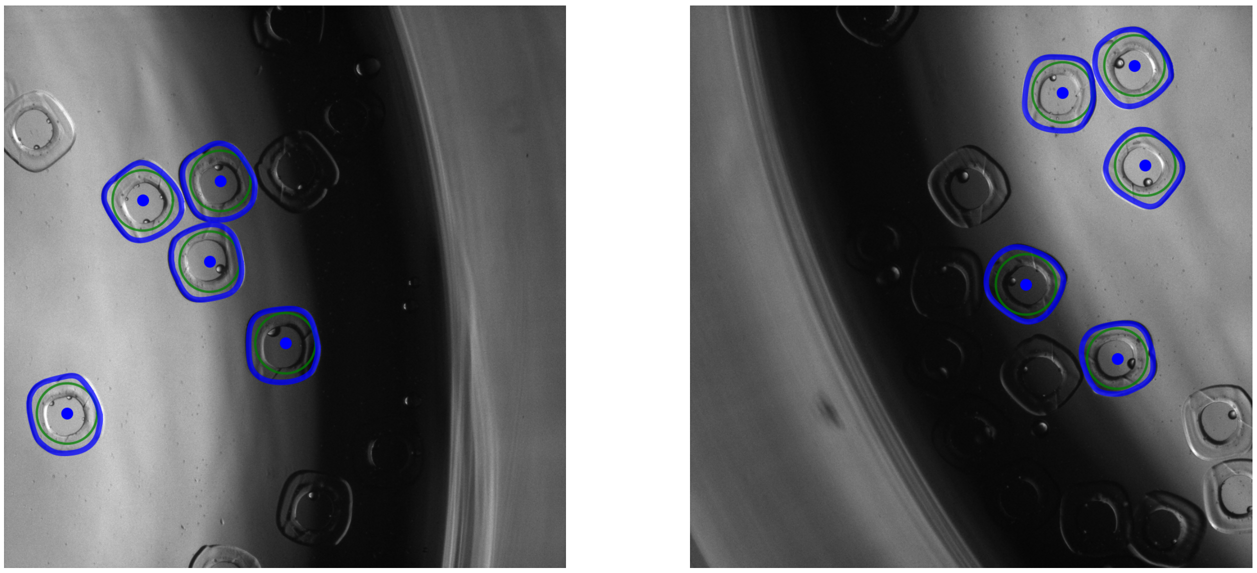

(a) (b)

Figure 2.10: Results from the method applied to images that contain only squares and circles respectively, showing independent performance for each particle type and robustness to noise such as overlapping particles and fluid condensate. © 2024 IEEE

2.4 Conclusion

The problem of detecting and segmenting the mixed particles under various lighting conditions is quite challenging. This presentation of classical techniques for solving the problem not only automates the process of detecting and segmenting the particles, but is done in a fashion that is interpretable and allows for simple calibration of parameters for the instrument being used, which is often not possible with deep learning techniques. Further work would consider additional particle morphologies to increase the scalability of the method, as well as methods to increase robustness to noise and the extreme lighting conditions found near the edge of the imaging wells. We would also explore the Ballard Hough Transform which could reduce the computational cost of the detection algorithm.



(a) (b)

Figure 2.11: Results from the method applied to squares near the edge of the imaging well. Particles which are largely obscured are not detected by the process. © 2024 IEEE

CHAPTER 3

Review of Feature Extraction, Graph Learning, and Active Learning

3.1 Introduction

In this chapter, we transition from the classical methods explored in the previous section to methods that utilize deep neural networks and graph learning. Our focus is on unsupervised neural network encoders, particularly useful in scenarios with a low label rate. These neural networks will be paired with a graph-based semi-supervised learning methods.

The first section of this chapter establishes a classical and well known neural network feature extraction architecture, known as autoencoders, as well as their slightly modified version the variational autoencoder. We note that there are popular linear feature extraction techniques such as principal component analysis and non-negative matrix factorization [9], but the nonlinearity of neural networks allows the models to learn much more complex features. Then, we discuss a more recent contrastive learning architecture known as SimCLR [41]. After reviewing these methods, we discuss Laplace learning [18], a graph based semi-supervised learning method that will be used to propagate labels through the latent space, as well as active learning, which is a machine learning regime designed for optimal annotated data usage.

3.2 Neural Network Encoders

In this section, we review two popular unsupervised neural network architectures that act as image encoders. Both of these architectures will use similar neural network layers, namely convolutional and linear layers, but will be structured very differently and have significantly different loss functions, which influences the way the models encode their data.

3.2.1 AutoEncoder

Autoencoders are a class of neural networks that are designed to perform dimensionality reduction by using an encoder, bottleneck, and decoder [42]. The purpose of the encoder is to receive high dimensional data, such as images, and meaningfully transform the data into a lower dimensional encoding. For images, the encoder is often a sequence of convolutional layers, max pooling layers, and ReLU layers. The bottleneck represents the layers that operate on the low dimensional encodings and is commonly a sequence of linear and ReLU layers. We call sequences of linear and activation layers (usually ReLU) multi-layer perceptrons moving forward. The decoder receives the output from the bottleneck and increases the dimensions back to the original size of the data. While the encoder normally uses convolutional layers, which typically decrease the dimensionality of the data, the decoder will often use transpose convolutional layers which act similarly but increase the dimensionality of the data.

In training, autoencoders encode data samples to a low dimensional latent representation and then decode the latent representation to recover the original data samples. A viable loss function that can be used, for example, is the mean squared error between the original image and the reconstruction. By enforcing this pipeline, the encoders are implicitly learning the most important features of the data. This is because in order for the decoder to successfully decode the original data samples, enough information must be preserved in the encoding process for accurate reconstruction. This intuition informs design choices in the network, implying that for an autoencoder to succeed, the encoder must be appropriately designed to learn the data's features, the bottleneck must be large enough to store critical information, and the decoder must be capable of accurately reconstructing the original data. If any one of these networks fails or are insufficiently designed, then the entire autoencoder will perform poorly. In practice, one wants to encode the data in the smallest dimension possible while preserving a sufficient amount of information from the sample. The purpose of autoencoders, as mentioned, is typically for dimensionality reduction. This means that once the autoencoder is trained, the decoder and potentially the bottleneck layers are discarded and the encoder is instead used for some other task. The decoder, which generates data from encodings, is trained only on the embeddings from the specific autoencoder and thus doesn't have much use elsewhere. Its purpose was simply to enforce that the encoder preserved information in the encoding.

Variational autoencoders (VAEs) are a modification of traditional autoencoders that can act as a generative class of neural networks [43, 44]. The main distinction between a VAE and a standard autoencoder is that a VAE attempts to regularize the latent space of the encodings by treating the latent space as a Gaussian distribution. The bottleneck includes the reparameterization step, which typically consists of a multi-layer perceptron tasked with learning the mean and standard deviation from the image. Beyond the bottleneck step, the decoder network will attempt to rebuild the original image based on the mean and standard deviation passed from the bottleneck layer. When properly trained, the decoder network of a VAE 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 still instead focus on using the encoder network to retrieve the latent representations of images in our dataset. The regularization of the encodings in the latent space adds additional smoothness to the encodings, whereas the latent space of a traditional autoencoder will likely be discontinuous and not smoothly vary between samples.

3.2.2 Contrastive Learning (SimCLR)

An alternate neural network approach to VAE feature extraction is a recently popularized learning framework known as contrastive learning. Contrastive learning uses labeled data and trains an encoder to minimize the distance in the latent space between samples from the same class and maximize the distance between samples of different classes. The images that are designated as similar samples are called positive pairs whereas the images that are designated as dissimilar are denoted negative pairs. The term contrastive learning comes from comparing and contrasting images to learn distinctions and features [45]. Training the encoder to minimize the contrastive loss is a pretext task, as in application we are often using it simply as an encoder for other tasks. This pretext task promotes learning of visual features and enforces a latent space that clusters or separates features by their perceived similarity. Recently, Chen et al. [41] published a seminal work on self-supervised contrastive learning for visual representations, introducing a novel framework known as SimCLR. SimCLR is self-supervised, meaning the labels attributed to the data samples are assigned by the model for learning purposes. In this way, SimCLR is effectively unsupervised since no ground truth labels are required for training.

Given a batch of n images, a series of augmentations are performed on each image to produce two distinct versions, resulting in $2n$ augmented images from the batch. For an image x_i, the augmentations will produce two augmented images that we 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 of 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 [41] uses the cosine similarity function:

$$\text{sim}(u, v) = u^T v/||u||||v||. \tag{3.1}$$

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

$$l_{i_1} = -\log \frac{\exp(\text{sim}(x_{i_1}, x_{i_2})/\tau)}{\sum_{k=1}^{N} \sum_{j=1}^{2} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(x_{i_1}, x_{k_j})/\tau)} \tag{3.2}$$

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

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

As stated, SimCLR is self-supervised and therefore uses no ground truth, labeled information regarding the samples. The success of SimCLR largely stems from the choice of augmentations which can make samples from the same class more indistinguishable from each other, preventing the model from separating them in the latent space, while still being differentiable from samples from other classes, allowing the model to separate them. Although proper augmentations can help, the model will still attempt to repel samples from the same ground truth class in the latent space. This is undesirable, but with full knowledge of all of the ground truth labels, all samples from the same class can be considered positive samples, attracting them in the latent space, whereas all samples from other classes can be effectively repelled. A supervised variant, known as SupCon, was developed by Prannay et al. and modifies SimCLR to use labeled information in this manner [46]. With this framework, it may not even be necessary to use a two-view approach, wherein each sample is augmented in two distinct ways to form pairs. The supervised loss is

$$\mathcal{L}_{\text{out}}^{\text{sup}} = \sum_{i \in I} \mathcal{L}_{\text{out},i}^{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p/\tau)}{\sum_{a \in A(i)} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_a/\tau)} \tag{3.3}$$

where $P(i) \equiv \left\{p \in A(i) : \tilde{\boldsymbol{y}}_p = \tilde{\boldsymbol{y}}_i\right\}$ is the set of indices of all positive samples, sharing the same ground truth label. SupCon also shows state of the art results on the ImageNet dataset and can show consistent outperformance over cross-entropy loss frameworks [46]. We use SupCon in Chapter 5 to learn features for pixel spectra in hyperspectral imagery.

Now we discuss the network architecture in order to properly utilize the SimCLR framework for learning visual features. Assuming we have the augmented batch of $2n$ images, $\{x_{1_1}, x_{1_2}, ..., x_{n_1}, x_{n_2}\}$, each one of these images will be passed through the first phase of the encoder. The encoder will transform a sample x_{i_j} into a feature embedding denoted as h_{i_j}. Afterwards, a projection head, which is typically a shallow 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 3.1. After having trained the neural network on a given dataset, the projection head is often 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 [41].

The key to a successful application of SimCLR is in the choice of the transformations or data augmentations that one wants the model to be invariant to, which should resemble what one expects to see within the dataset. For example, in a typical image dataset, we expect the network to be invariant to augmentations such as rotation, reflections, crops, color jitter, and other noise effects that can be emulated with artificial transformations. These transformations still preserve the class structure of the image and therefore, via training, we are enforcing that the network becomes invarient to these transformations. SimCLR has demonstrated state of the art performance on ImageNet and numerous other datasets, by generating latent space embeddings that separate well under linear classifiers [41].

3.3 Embedding Visualizations

In this section we provide a brief review of two popular data visualization algorithms. When using a feature extraction methods to encode data, it can often be difficult to understand

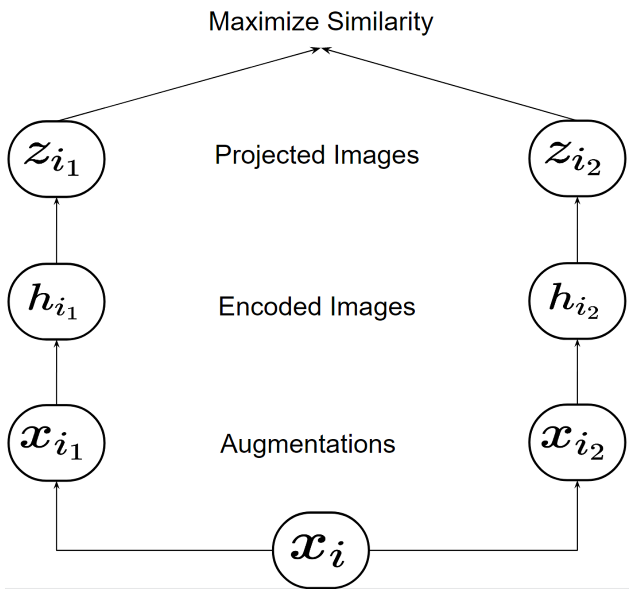


Figure 3.1: SimCLR framework. A sample image x_i is augmented twice creating x_{i_1} and x_{i_2}. These augmented images are pushed through an encoder network and a projection head resulting in z_{i_1} and z_{i_2}. An objective of SimCLR is to maximize the similarity between positive pairs such as these and minimize the similarity with the other samples in the batch.

how the samples are being embedded in the latent space relative to one another, especially if the latent space is still above two or three dimensions, beyond our ability to effectively visualize in plots. Fortunately, there are popular methods that can reduce high dimensional data to two dimensions, allowing quick and easy interpretation of similarities and relative locations of embeddings by humans. We use these embeddings to verify the effectiveness of particular neural network encoder models in Chapter 4.

A classic method to visualize the data in two dimensions is to use a t-SNE visualization [47], which stands for t-Distributed Stochastic Neighbor Embedding and is an extension of Stochastic Neighbor Embedding [48]. The principal idea of the t-SNE method is to compute pairwise probabilistic similarities in the high-dimensional space that normalized for density and then minimize an energy functional over embeddings in a lower dimension that preserve the pairwise similarities. This method is widely used and produces very interpretable visualizations.

Another, more recent method is UMAP [49], which stands for Uniform Manifold Approximation and Projection. Like t-SNE, UMAP computes pairwise distances in the high dimensional space and attempts to preserve the pairwise distances in the lower dimensional space. The fundamental concepts of UMAP, however, are rooted in algebraic topology and topological data analysis and involve creating a similarity graph from the high dimensional features. Notably, UMAP often runs faster than t-SNE and may scale better for larger datasets.

3.4 Graph Embedding

Given a feature extraction method, we can view images as vectors embedded in $\mathbb{R}^d$, a much lower dimensional space than the raw data. At this point, there has been no effort made to classify the data. Because we are in the low label rate regime, we are interested in utilizing transductive, semi-supervised learning models, which are models that use both labeled and unlabeled data simultaneously to make predictions on the unlabeled data. In order to do so, we create a similarity graph, which contains only the relational information from the data.

A graph is a mathematical structure containing nodes and edges. We can write a graph as $\mathcal{G}(X, W)$ where X denotes the nodes and W denotes the edge weights between the nodes. In the context of image embeddings, the set X corresponds to the image samples and W is an (often symmetric) matrix where $W_{i,j} \geq 0$ represents the similarity between image samples. In order to construct this graph, we first need a similarity metric $w(x_i, x_j)$ that meaningfully captures similarity in the latent space. Desirable properties for the similarity metric include returning large values when the samples are near each other in the latent space and small or negligible values when the samples are distant. To emphasize locality in the latent space, we use the following similarity function

$$w_{ij} = e^{-4||x_i - x_j||^2 / d_k(x_i)^2} \tag{3.4}$$

where $d_k(x_i)$ represents the distance between x_i and its k^{th} nearest neighbor [2]. The negative exponential term promotes locality, enforcing that distant samples have a very small similarity. The term $d_k(x_i)^2$ in the denominator is a normalization term for the graph construction that reduces the edge weights for nodes that are located in very dense regions, where the k^{th} nearest neighbor is relatively close and thus $d_k(x_i)^2$ is small. This can reduce overdependence on on these highly centralized nodes [50]. The metric for measuring distance can, however, be domain dependent. For example, there has been much success using the cosine distance for applications to hyperspectral imagery and we will use it for works in this thesis [51, 52, 53]. The choice to use an angular metric is also reinforced by the SimCLR paper, which uses cosine similarity in Equation 3.1 [41]. The cosine distance for two vectors, x_i and x_j, is

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = 1 - \cos(\theta) \tag{3.5}$$

where θ is the angle between the vectors. A common way to calculate the cosine distance is to use the standard Euclidean distance after normalizing the feature vectors first. To show equivalence, up to constant factor, between normalized Euclidean distance and cosine distance, we show the following:

$$\begin{aligned}
2(1-\cos(\theta)) &= 2(1-\frac{x_i \cdot x_j}{\|x_i\|\,\|x_j\|}) \\
&= 2 - 2\frac{x_i \cdot x_j}{\|x_i\|\,\|x_j\|} \\
&= \frac{\|x_i\|^2}{\|x_i\|^2} + \frac{\|x_j\|^2}{\|x_j\|^2} - 2\frac{x_i \cdot x_j}{\|x_i\|\,\|x_j\|} \\
&= \frac{x_i}{\|x_i\|} \cdot \frac{x_i}{\|x_i\|} + \frac{x_j}{\|x_j\|} \cdot \frac{x_j}{\|x_j\|} - 2\frac{x_i}{\|x_i\|} \cdot \frac{x_j}{\|x_j\|} \\
&= (\frac{x_i}{\|x_i\|} - \frac{x_j}{\|x_j\|}) \cdot (\frac{x_i}{\|x_i\|} - \frac{x_j}{\|x_j\|}) \\
&= \left\| \frac{x_i}{\|x_i\|} - \frac{x_j}{\|x_j\|} \right\|^2 .
\end{aligned} \tag{3.6}$$

For large amounts of data, it is important to take into account the amount of computational cost required for the graph. If the graph is constructed from n data samples, then the adjacency matrix, W, would be a dense $n \times n$ matrix. If we need to create the adjacency matrix often, perhaps in each batch of a training cycle, or if n is very large, it becomes very expensive to perform the pairwise distance calculations repeatedly. Moreover, even if a dense adjacency matrix were computed, it could still be expensive to perform repeated matrix multiplications or inversions which may be required for later tasks. For this reason, we seek a way to reduce the memory size of the matrix W. The first method we discuss is to use a sparse formulation and the second is to use a low rank approximation.

For the sparse approximation, we only include the k largest similarities for each sample, for some value k, and exclude the other pairwise comparisons, setting their edge weights zero. In order to find the largest similarities, we need to use an approximate nearest neighbors search algorithm. We use Annoy [54] (Approximate Nearest Neighbors Oh Yeah) and find the nearest k neighbors using the Euclidean distance after normalizing the features (for cosine distance). This immediately reduces the cost of constructing the matrix, W, as the number of pairwise comparisons is reduced from n^2 to $n \times k$ where $k << n$. With W being sparse, this allows for sparse implementations using the scipy sparse matrix library, leading to more efficient storage, and sparse operations, minimizing unnecessary computations [55].

Alternatively, low rank approximations can be used for the construction of the weight matrix. As opposed to sparse approximations, low rank approximations work by constructing smaller matrices that, when multiplied, recover the original matrix with minimal information loss. For the Nystrom approximation [56], we begin by taking some small random subset of the samples that we intend to use for generating the adjacency matrix. From this subset, we use interpolation to approximate the rest of the adjacency matrix. If our dataset has n samples and our random subset has m samples, then we can construct the adjacency matrix $A \in \mathbb{R}^{m \times m}$ between just the samples in the subset and we can construct the adjacency matrix $B \in \mathbb{R}^{(n-m) \times m}$ between the selected samples and the rest of the samples. Assuming without loss of generality that the random subset of samples is the first m samples, our weights matrix W would be

$$W = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{3.7}$$

where $C \in \mathbb{R}^{(n-m)x(n-m)}$ represents the intra-weights of the remaining data points not in the random subset. The matrix of approximate eigenvectors $\bar{U}$ is given by the Nystrom Extension to be

$$\bar{U} = \begin{bmatrix} U \\ B^T U \lambda^{-1} \end{bmatrix}$$

where U is the matrix of true eigenvectors and λ is the matrix of eigenvalues [56]. From this, the approximate version of W would be

$$\begin{aligned} \hat{W} &= \bar{U} \lambda \bar{U}^T \\ &= \begin{bmatrix} A & B \\ B^T & B^T A^{-1} B \end{bmatrix} \\ &= \begin{bmatrix} A \\ B^T \end{bmatrix} A^{-1} \begin{bmatrix} A \\ B \end{bmatrix}. \end{aligned}$$

Essentially, the missing block of data, C, is approximated with $B^T A^{-1} B$. Some more work is done beyond this point to ensure that the eigenvectors represented in $\bar{U}$ are orthogonalizedm which is omitted here. Using the Nystrom low rank approximation has proven to be effective for approximating the symmetric graph Laplacian, a matrix derived from the adjacency matrix discussed in Section 3.5, for classification of high dimensional data [57, 58, 59].

3.5 Graph Based Semi Supervised Learning

Recent work (e.g. [2, 60, 61, 62]) shows the efficacy of graph-based methods in semi-supervised learning (SSL) and active learning 3.6. Graph based methods imbue 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 [18], also called label propagation, the graph MBO method [63], p-Laplace learning [62], and graph-based Poisson learning [60]. In this section, we introduce the graph based, semi-supervised learning algorithm called Laplace learning [18].

Let $\{x_1, ..., x_n\} := \mathcal{X} \subset \mathbb{R}^d$ denote the feature vectors of the images in d dimension and let $\mathcal{L} \subset \{1, 2, ..., n\}$ denote 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 data. For a dataset with K classes, we let $y_j \in \{1, 2, ..., K\}$ denote the label for image j so that the labels correspond to their one-hot encoding as $e_{y_j} \in \mathbb{R}^K$. Let $\mathcal{G}(X, W)$ be a graph where X denotes the vertices and W denotes the edge weights between the data. Given this graph framework, semi-supervised learning techniques have seen great success. Zhu el al. introduced Laplace learning in "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions" [18], which is the primary semi-supervised learning technique used throughout this thesis. Laplace learning solves for the label function $f : X \to \mathbb{R}^K$ such that

$$f(x_i) = \begin{cases} \frac{1}{d_i} \sum_{j=1}^{n} W_{ij} f(x_j) & \text{for } i \notin \mathcal{L} \\ \mathbf{e}_{y_i} & \text{for } i \in \mathcal{L}, \end{cases} \tag{3.8}$$

where $d_i = \sum_{j \neq i} W_{ij}$. This formulation can be interpreted as assigning the label prediction for unlabeled nodes as the weighted average of its neighbors in the graph while keeping the known, labeled information fixed to the ground truth. From this perspective, we can intuitively reason that Laplace learning is similar to solving the steady state heat equation on the graph where the ground truth labeled nodes are akin to fixed boundary conditions, such as hot or cold sinks [64]. In a partial differential equations sense, solving the steady state heat equation implies that the solution is fixed in time and thus

$$\Delta f = f_t \implies \Delta f = 0 \tag{3.9}$$

where t is the time parameter and x is the spatial parameter. This second equation is the Laplace Equation as the second spatial derivative is known as the Laplacian.

Here, we introduce the graph Laplacian, $L = D - W$, where D is the diagonal matrix with $D_{ii} = d_i$ mentioned above. The graph Laplacian takes its name from the Laplace operator due to its similarities mentioned above [50]. The graph Laplacian has many extremely useful features. For example, all graph Laplacians are positive semi-definite by construction and the multiplicity of the zero eigenvalue indicates the number clusters present in the graph structure, with connections to spectral clustering. Each connected component, $C \subset X$ of the graph introduces a new linearly independent vector into the kernel, which is the indicator vector $\mathbf{1}_C$ where

$$[\mathbf{1}_C]_i = \begin{cases} 1 & i \in C \\ 0 & i \notin C \end{cases} . \tag{3.10}$$

With this in mind and treating f as a matrix in $\mathbb{R}^{n \times K}$ such that each row is a label prediction vector in $\mathbb{R}^K$, the problem of Laplace learning can be reformulated as

$$\begin{aligned} [Lf]_i &= \mathbf{0} \quad \text{for } i \notin \mathcal{L} \\ f_i &= \mathbf{e}_{y_i} \quad \text{for } i \in \mathcal{L} \end{aligned} \tag{3.11}$$

since

$$[Lv]_i = \sum_{j \neq i} w_{ij}(v_i - v_j) \tag{3.12}$$

for any vector v.

We have so far approached this formulation of Laplace learning from a perspective of averaging the label prediction of nearby nodes in the graph. It can also be interpreted in the form of an variational minimization problem with energy

$$E(f) = f^T Lf + \lambda \left| F(f-y) \right|^2 \tag{3.13}$$

where F is a diagonal indicator matrix, with $F_{ii} = 1$ if i is labeled and zero otherwise, and λ is a hyperparameter [16]. Since

$$v^T Lv = \frac{1}{2} \sum_{i,j} w_{ij}(v_i - v_j)^2 \tag{3.14}$$

for any vector v, we see that the first energy term controls the smoothness of the label predictions across the nodes. This term is known as the graph Dirichlet energy. The second energy term controls the relaxation in enforcing the ground truth labels. Minimizing the energy functional while taking the limit as λ goes to infinity recovers the strongly enforced label problem. Via variational calculus, the solution to the energy minimization problem is exactly Equation 3.11 when the labels are strongly enforced.

In principle, Laplace learning generates a graph harmonic function which is interpreted as propagating the labels across the dataset. Without loss of generality, re-ordering the nodes so that the labeled nodes are ordered first, the solution can be written as

$$f := \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} Y \\ -L_{\mathcal{U},\mathcal{U}}^{-1} L_{\mathcal{U},\mathcal{L}} Y \end{pmatrix}, \tag{3.15}$$

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 [2]. This solution only exists when every unlabeled node has some connection to a labeled node. If there is a connected component of unlabeled nodes with no connection to a labeled node, then enforcing that the graph Laplacian is 0 over the component will not produce a unique solution. Here we briefly justify from a linear algebra perspective why $L_{\mathcal{U},\mathcal{U}}$ is invertible when the connected component of unlabeled nodes is connected to at least one labeled node.

Theorem 2. *Assume G is a graph such that unlabeled nodes in $\mathcal{U}$ form a connected component and there is at least one connection to a labeled node from $\mathcal{L}$. Then $L_{\mathcal{U},\mathcal{U}}$ is invertible.*

Proof. First we consider the subgraph $G_{\mathcal{U}}$ by selecting only the unlabeled nodes and the edges between them. This generates the graph Laplacian L', which is positive semi-definite and whose kernel is spanned by the vector of ones since there is only a single connected component. Then we can see that

$$L_{\mathcal{U},\mathcal{U}} = L' + D' \tag{3.16}$$

where D' is the diagonal matrix with $0 \leq D'_{ii} = \sum_{j\in\mathcal{L}} w_{ij}$ for $i \in \mathcal{U}$. Since there is at least one connection from a labeled node to unlabeled nodes, then there is at least one nonzero entry of D'. Clearly D' is also a positive semi-definite matrix.

In order to show that $L_{\mathcal{U},\mathcal{U}}$ is invertible, we prove that the only element in the kernel is the 0 vector. Let v be a vector such that $L_{\mathcal{U},\mathcal{U}} v = 0$. It then follows that

$$v^T L_{\mathcal{U},\mathcal{U}} v = v^T (L' + D') v = v^T L' v + v^T D' v = \mathbf{0}. \tag{3.17}$$

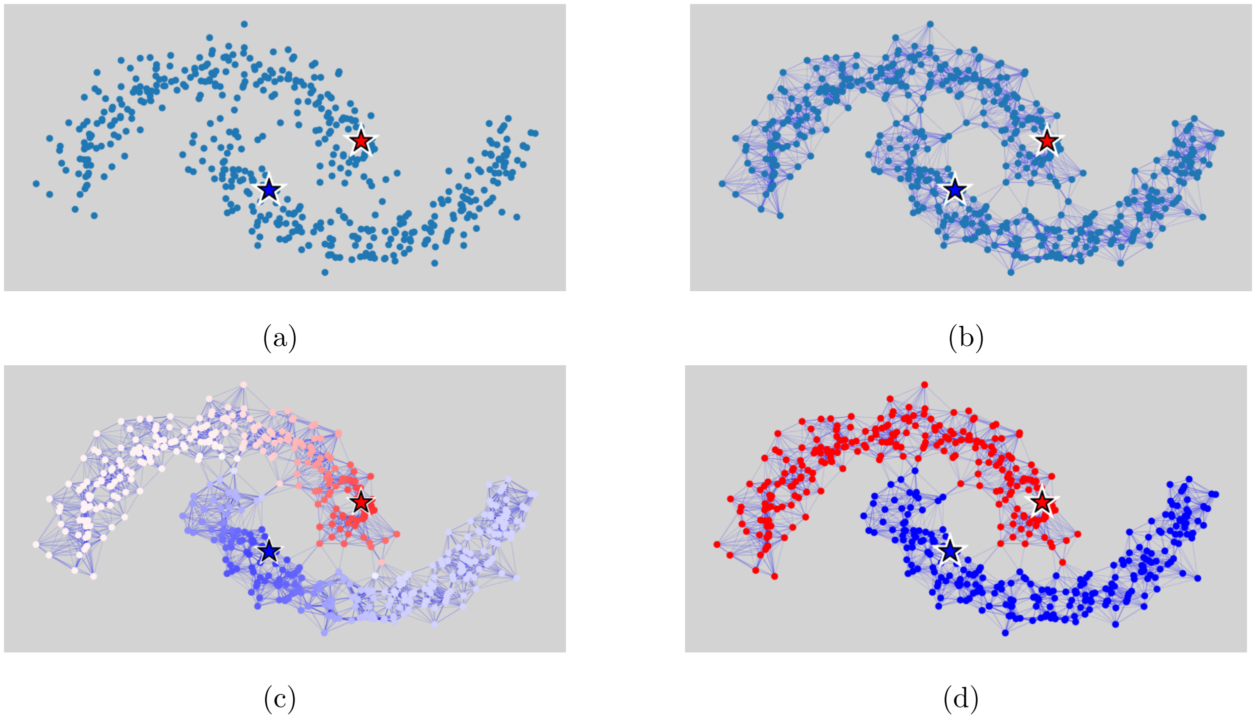
Since both of the matrices L' and D' are positive semi-definite, then this condition implies that

$$v^T L' v = 0 \quad \text{and} \quad v^T D' v = \mathbf{0}. \tag{3.18}$$

Since L' is a graph Laplacian, this implies that $v = c\mathbf{1}$ for some $c \in \mathbb{R}$. Since D' is diagonal with some positive value D_{ii} for some i, then this condition requires that $v_i = 0$. Therefore, using both of these conditions, we see that v is actually the $\mathbf{0}$ vector. $\square$

In practice, it may not be the case that all of the unlabeled nodes form a connected component. In such a case, we would be able to divide the graph Laplacian, L, into block matrices corresponding to each connected component and show that each block matrix is invertible by the above theorem, assuming that the connected components share an edge with a labeled sample. The solution to Laplace learning, Equation 3.15, produces the prediction matrix $f \in \mathbb{R}^{n \times k}$ where each row is interpreted as the class predictions for the corresponding sample. Therefore the predicted class for sample i is taken as the $\text{argmax}_j(f_{ij})$.

Figure 3.2 visually shows how Laplace learning functions with feature data and two labeled nodes. For this experiment, we generated 500 samples from the two moons dataset with a noise value of 0.12, seen in Figure 3.2(a), and chose one sample from each class to be labeled, represented by the stars [40]. We use the graph learning library to perform a k nearest neighbors search with $k = 15$ and generate a graph with similarity function Equation 3.4 using the Euclidean distance instead of cosine distance. The edges from the graph structure are overlaid onto the feature vectors, seen in Figure 3.2(b), although we note the edge weights are not equivalent. Using Laplace learning on the resulting graph produces prediction values for the classes, shown in Figure 3.2(c). Thresholding the prediction values shows the most likely predicted class and is seen in 3.2(d). The graph learning python library was used for this experiment [65].



(a) (b) (c) (d)

Figure 3.2: An example of how how data can be classified using Laplace learning. (a) Feature vectors from two moons dataset with two labeled nodes represented as a red and blue star. (b) A nearest neighbors search with k=15 has been used and the resulting edges, with varying weights, are shown creating a graph. (c) The soft label predictions from using Laplace learning are shown with colors indicating label prediction corresponding to the classes from the red and blue stars. (d) The thresholded label predictions from using Laplace learning.

3.6 Active Learning

As we are interested in the low label rate regime for image classification, we explore a machine learning approach known as active learning [66]. In active learning, a model is used to make predictions on a class and then, based on those predictions, new samples are selected to be manually labeled by a human expert. This process repeats until the users are satisfied with the prediction results from the model or sufficient energy has been expended in labeling the samples. The purpose of this process is to minimize the amount of redundant information in labeled samples, thus only querying the most useful samples. In doing so, the strategy aims to maximize the model accuracy with the fewest ground truth labels. Typically, active learning is preferred when either individual samples require significant time, energy, or expert knowledge to label or there are far too many samples to effectively label a substantial amount of them.

Given some dataset $\{x_1, ..., x_n\}$, let $\mathcal{L} \subset \{1, 2, ..., n\}$ denote the set of indices for data with an associated, known label and $\mathcal{U} = \{1, 2, ..., n\} - \mathcal{L}$ denote the set of indices of unlabeled data. 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. When a sample is queried and labeled, it would then be used to update the labeled set, $\mathcal{L}_{new} = \mathcal{L}_{old} \cup \{i\}$. Then the prediction model would be retrained or fine-tuned on the new labeled set before restarting the process over again.

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. Both methods have their pros and cons. Sequential active learning allows for a greedy approach, where each queried sample is a maximum of the acquisition function, often leading to strong results. The downside of this is that the active learning process needs to be repeated many times for sufficiently many samples to be labeled by the oracle. With batch active learning, the queried samples can be labeled in parallel, requiring fewer iterations of the active learning loop. The difficulty with batch active learning is avoiding querying samples that share too much mutual information. It is common to measure mutual information using Fisher information [67, 66] or by ensuring that the batched samples are distant in some latent space [68].

For active learning to proceed, we need both a prediction model and an acquisition function. For the prediction model, we use the graph-based Laplace learning method discussed in Section 3.5. Although we explore several acquisition functions, the uncertainty acquisition function will be the primary focus in Chapter 4.

We consider prediction models taking in unlabeled nodes and outputting a vector of length K such that the i^{th} component corresponds to the probability that the given sample belongs to the i^{th} class. When evaluating a model, the predictions are thresholded to predict the most likely class, but by using the unthresholded prediction vector we can establish a notion of confidence and uncertainty in the model's predictions. 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 $f(x_i) \in \mathbb{R}^K$ for the sample $i \in \mathcal{U}$, we define the margin to be the difference between the first and second highest predictions:

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

Another popular way to measure uncertainty is to use Shannon entropy [69]:

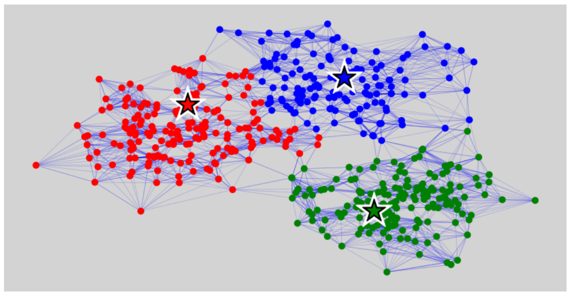
$$\text{Entropy}(x) := \max_k u(x)_k - \sum_i P(y_i|x;\theta) \log P(y_i|x;\theta).$$

While the margin-based uncertainty simply measures the distance between the two most confident predictions, entropy uncertainty utilizes all of the prediction weights. The uncertainty acquisition functions query 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 [70].

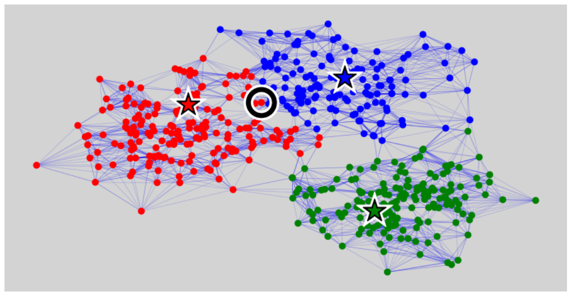
Another popular acquisition function is variance optimality (VOPT). VOPT is agnostic of the observed and predicted labels and essentially attempts to query the labels such that the unlabeled data suffers the least amount of variance in terms of its distribution [71]. An extension of VOPT that preserves submodularity guarantees and minimizes predictive covariance is σ optimality [72]. The last acquisition function to be mentioned is Model Change [73]. 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 [73]. There has been work done to combine the ideas of model change with variance optimality resulting in MCVOPT as another, derivative acquisition function [2]. 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, which is known as pool-based sampling. This is more common when datasets become very large, but introduces further considerations such determining the pool to sample from.

In some sense, these acquisition functions are motivated largely by heuristics. A common theme in the heuristics is the contrast between exploration and exploitation. Exploration in an active learning setting involves querying regions without much known information, which can provide a more complete understanding of the label distributions. Exploitation on the other hand involves querying samples near decision boundaries where other known labeled information may be near by. This allows for a refinement of the decision boundary. Both exploration and exploitation are necessary for accurately exploring a dataset.

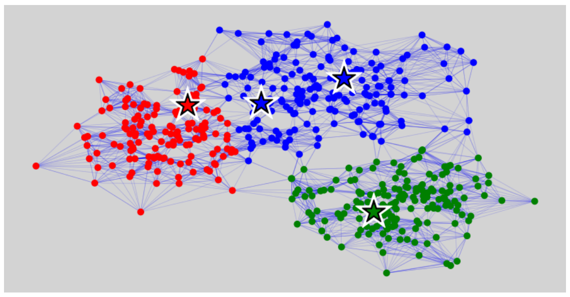
Figure 3.3 shows a cycle of active learning using a dataset of three Gaussian clusters with various noise. In Figure 3.3(a), the data with graph structure is shown. The graph is built using a k nearest neighbors search, for $k = 15$, and the similarity function in Equation 3.4 with the Euclidean distance [40, 65]. Three nodes are designated as initial labeled data with the stars. Using the margin uncertainty acquisition function, Figure 3.3(b) shows the most uncertain node being selected with the black circle. This node is queried, the label is recovered, and the graph learning process is retrained, producing the predictions in Figure 3.3(c). Repeating this process would further refine the predictions.



(a)



(b)



(c)

Figure 3.3: Active learning process. (a) Initial graph with three labeled nodes (stars) and starting predictions indicated by the colors red, blue and green. (b) The queried node is highlighted by the black circle. The acquisition function is margin uncertainty. (c) The queried node is labeled and now represented as a blue star. The resulting predictions are updated.

CHAPTER 4

Graph Active Learning with Applications to Synthetic Aperture Radar Data

4.1 Introduction

In this section, we explore the applications of neural network encoders with graph based semi-supervised learning to classify synthetic aperture radar (SAR) images. First, we provide a background on what SAR images are and how they are collected. Then, we examine various SAR datasets and evaluate the effectiveness of our semi-supervised learning methods on these datasets. The work in this section is largely adapted from the SPIE publication ”Utilizing contrastive learning for graph-based active learning of SAR data” in collaboration with Riley O’Neil, Jeff Calder, and Andrea Bertozzi [4]. Notably, the material regarding the background on SAR images and work on the Caesar dataset are new in this thesis and not present in the mentioned work. Jeff Calder and Andrea Bertozzi supervised the project, and provided feedback and direction. I was responsible for all of the data preprocessing on the Caesar dataset as well as training and running the neural network encoders and active learning models on the Caesar and MSTAR datasets. Riley’s contributions were in evaluating the quality of the SimCLR embeddings on the MSTAR dataset.

4.1.1 Synthetic Aperture Radar

Synthetic Aperture Radar (SAR) imaging is a remote sensing technique for collecting high resolution reconstructions from resolution-limited apertures mounted on moving objects, such as planes and satellites [74, 75]. 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 Earth monitoring, climate change research, change detection, and the detection and identification of specific targets [74].

In order to collect SAR data, a SAR satellite will emit energy towards a target and then measure the amount of energy returned from the target [75]. This is in comparison to a standard high resolution camera which relies on natural light to interact with the target and then measures specific light frequencies in response. A domain specific consideration of SAR data is choosing the appropriate frequency of light to emit towards the target. Certain energy frequencies will have stronger penetration power than others, indicating that specific frequency ranges will be ideal for certain targets. For example, the frequency range between $18 - 27$GHz is a rarely used frequency range that is ideal for measuring H_2O absorption whereas frequencies in the range of $4-8$GHz (known as C Band) are very popular frequencies used for global mapping and change detection among other applications [75].

SAR satellites also need to take into consideration the polarization of light for emitting and receiving. Most SAR sensors use linear polarization, meaning SAR satellites can emit either or both vertically and horizontally polarized light and also detect either or both horizontally and vertically polarized light. To keep track of the polarization, the data is labeled as some combination of V and H where the first term indicates the emitted polarization and the second is the received. So VH would indicate that vertically polarized light is emitted and horizontally polarized light is received by the sensor [75]. Linearly oriented surfaces tend to preserve the polarity whereas randomly oriented structures can scatter and depolarize the signal as it bounces. Using the different polarizations in conjunction can recover more information about of the image. A "single pol" system is one that only transmits and receives one polarization (typically HH or VV), while a "dual pol" system can typically transmit in one polarization but receive in two. A "quad-pol" or full system can transmit both H and V waves and receive both H and V waves. Each combination of polarizations can be treated as a different channel, not unlike the three channels in RGB imagery. Thus, satellites with limited polarization options can produce single-channel data, whereas others can generate up to four channels of data, although the information across these channels may not differ significantly.

4.2 Caesar Dataset

The Caesar dataset is a SAR dataset for ship detection with complex backgrounds [1]. It was constructed using 102 images from the Gaofen-3 satellite and 108 images from the Sentinel-1 satellite. The dataset consists of 43,819 ship chips (images) containing 59,535 ships. There are numerous imaging modes and resolutions in the original images that were cropped and labeled to create the dataset, providing a large variety of image types. Furthermore, the Caesar Dataset is based on images from sensors that use different polarization systems, resulting in essentially two channels per image, seen in Figure 4.1. It is important to note that these ship chips were extracted from images with various resolutions and swaths. This variability results in multiple ship scales, ranging from very small to those occupying nearly the entire chip. Furthermore, some of the images include land or islands which can lower accuracy.

As it is, the dataset is to serve as training and testing for object detection. This means that the chips are broad in scope and can contain multiple ships within a chip. In addition to the images, there are separate files pertaining to the location of the ships as well as bounding boxes surrounding them. This is meant to serve as the ground truth for a image detection algorithm that locates the ships in the chips and creates a bounding box around it, not unlike work done in Section 2.3.

In order to convert this object detection dataset to a classification dataset, we decided to

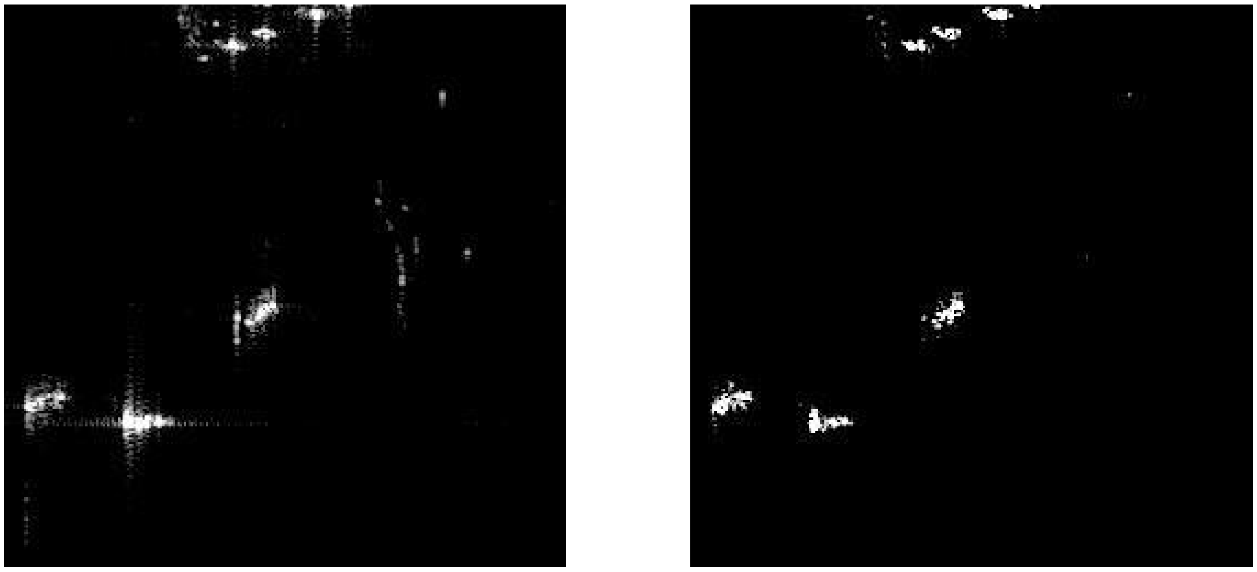

(a) (b)

Figure 4.1: Different polarizations of SAR images taken from the Gaofen satellite [1]. Three ships are seen in the lower left of the image in which (a) represents the hh channel and (b) The hv channel.

subsample regions in each of the chips. The goal is to create a binary classification problem, so we deliberately subsample regions that overlap with the ship bounding boxes to create the 'ship' class and we subsample regions that don't contain ships to create the 'non ship' class. Now a classifier can be trained to determine whether or not the subsampled region contains a ship or not. For example, Figure 4.2 shows a sample chip with the corresponding bounding box overlaid on each of the ships present in the chip. From this, we extract at least three subsamples centered around each of these bounding boxes with the corresponding 'ship' label associated.

In order to be used in a neural network pipeline, we require that all images in the dataset have the same dimensions or size. This means a uniform choice of the subsample window size is required (without re-scaling). Provided in [1] is a histogram plot demonstrating the relative size of the bounding boxes with respect to the chips. In the plot, the vast majority

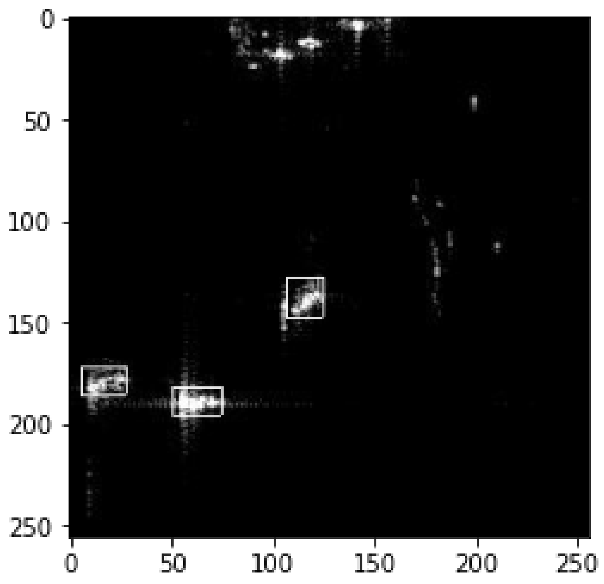


Figure 4.2: Sample chip from Figure 4.1(a) with bounding boxes overlaid around the three ships.

of the bounding boxes have size less than 30% of the chip size. Consequently, we chose to make the subsample bounding boxes have a relative size of 30% of the chip size, or in other words 76x76 pixels. This is larger than most of the ships' bounding boxes, so there may be overlap with other ships and blank space, but the additional variability should make for a more diverse dataset.

To make this problem straightforward, we assume, wherever possible, that the ships are centered exactly in the middle of the chip. However, this is not feasible for ships located near the image boundaries. Figure 4.3 shows an example of sampling the ships from a chip. It is worth noting that this could potentially create a bias in a classification pipeline that indicates learning when bright regions are centered in the chip. A way to fix this could include adding uniform or Gaussian noise to the center pixel location prior to cropping. Uniform noise would indicate that the ship could reasonably appear anywhere within the subsampled region whereas Gaussian noise would have a bias towards the ship being centered.

Naturally, in order to construct this binary classification dataset, we would also need to have samples that do not feature ships. In order to do this, we could naively randomly

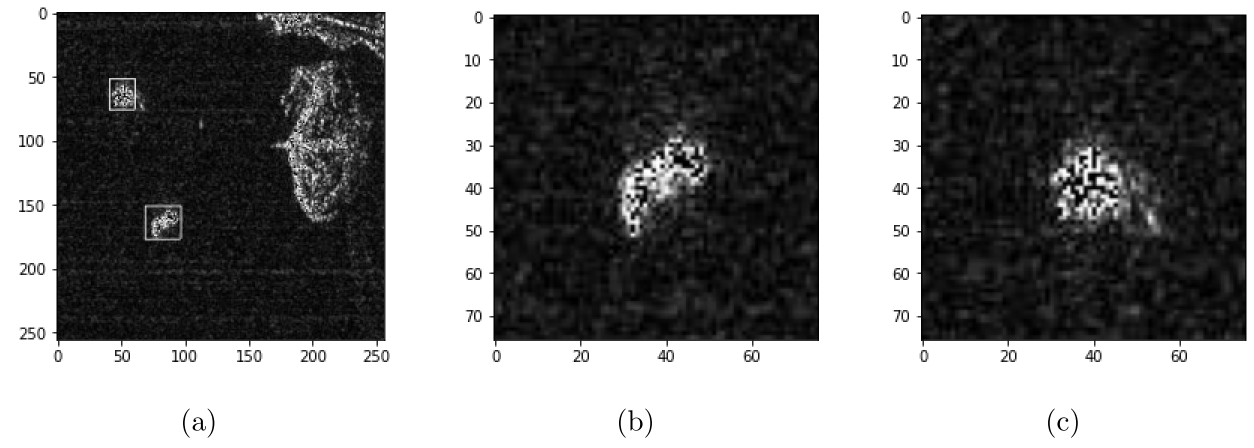


Figure 4.3: Ship images sampled from a larger chip. (a) The original chip with the bounding boxes for ships shown. (b) A cropped region surrounding the lower bounding box. (c) A cropped region surrounding the upper bounding box.

sample regions of the appropriate size and ensure they do not contain any ships by using the bounding box information. In order to determine if the randomly sampled window contained a ship, we used an intersection over union metric and arbitrarily decided that if the randomly chosen window contained 10% of the bounding box of any ship, then we would discard it. Another concern was that since many of the ships are simply over open ocean, the backdrop is often very dark or entirely black. In order to avoid superfluous black images, we added a condition that ensured a non-negligible L_2 variation (the square of the total variation semi-norm). Figure 4.4 shows two samples taken from Figure 4.2 that would be used as the 'non-ship' label. Notably, these samples do not intersect the ships in the image and also contain sufficient amounts of noise.

In summary, this dataset has 9779 images with dimensions of 76x76. There are 5335 samples that include a ship and 4444 samples that do not contain a ship, which together create the binary classification dataset.

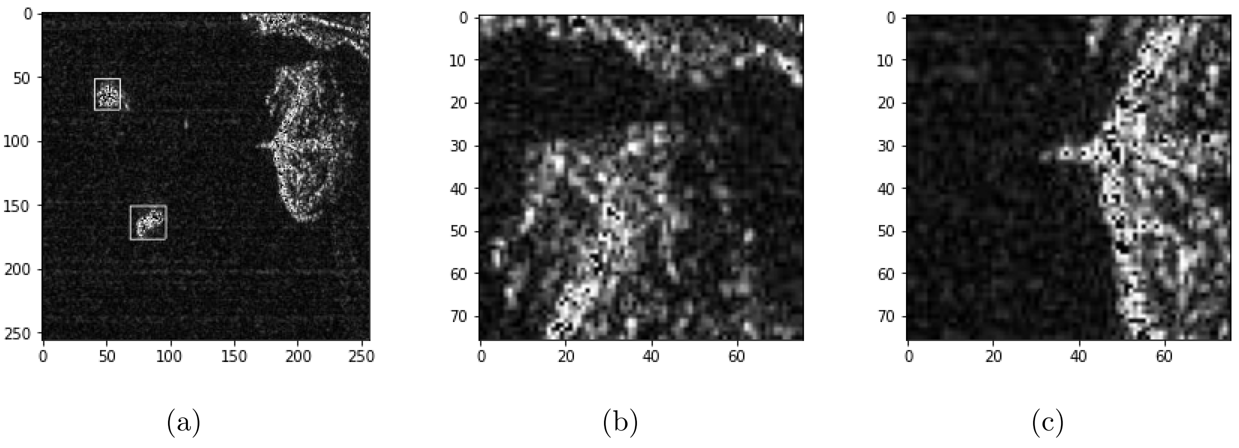


Figure 4.4: Non-ship images samples from a larger chip. (a) The original chip with bounding boxes for ships. (b) and (c) Cropped regions of land for the non-ship class that have sufficient variation.

4.3 MSTAR

The Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset [76] was published in 1998 by Sandia National Laboratories with funding from the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL). The MSTAR dataset consists of 6,874 SAR chip images, collected by a Sandia X-band radar operating at 9.60 GHz with a bandwidth of 0.591 GHz. Designed for automatic target recognition, the dataset features ten 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 angle of 17° corresponds to the testing data [76].

We preprocess the data following the method described in Miller et al. [2]. Specifically, 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. Additionally, the images are forced into a 3-channel format; the first channel is the magnitude of the image and the second and third channels are the real and complex

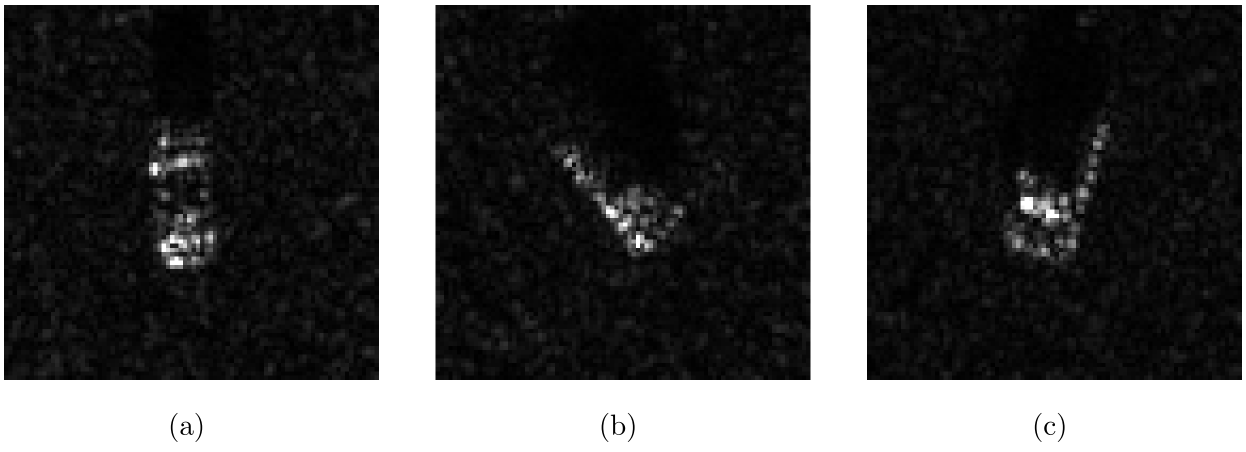


Figure 4.5: The magnitude (first channel) from three SAR images corresponding to different classes.

phases, respectively. Letting M be the magnitude of the image and P be the phase of the image, the three-channel format is given as

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

Examples of MSTAR images are shown in Figure 4.5. A notable characteristic is that the target is always directly centered in the image. Additionally, a shadow appears behind the target, which is a consequence of SAR imaging where the sensor's energy is blocked by the target. The shadow effect is more specific to MSTAR and is not prevalent in the Caesar dataset.

4.3.1 Previous and Related Work

Work on Automatic Target Recognition for SAR data predominantly focuses on the MSTAR dataset. Most of the work directly applied to MSTAR can be separated into pre-deep learning approaches, such as support vector machines (SVM), and modern deep learning approaches, such as convolutional neural networks (CNNs). Pre-deep learning methods are varied in their approaches. Researchers have used feature extraction combined with scattering models

[77, 78] as one approach. Many approaches utilize SVMs for their classification; such as in a straightforward manner in [79], paired with Adaboost [80], or in parallel with a hand-tuned covariance embedding scheme [81].

Convolutional neural networks are central to some of the most successful state-of-the-art machine learning methods, particularly 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 on the order of 4000 training chips, but, realistically, large amounts of labeled training data are not often available in practice. 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 [82]. Another approach, the All-Convolutional Networks, replaced fully connected layers with additional convolutional layers [83, 84]. Other researchers have simply tried to minimize the number of parameters in the CNN [85]. Additional approaches include unsupervised methods, such as variational autoencoders. The Euclidean Distance Restricted Autoencoder [86] method uses an autoencoder designed to embed images from the same class nearby each other, 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. For MSTAR, researchers have attempted to replicate SAR images using computer software. In the SAMPLE dataset, researchers present a dataset of real and synthetic data based on the MSTAR dataset [87]. By using Computer Aided Design (CAD) models for the vehicles, researchers have reproduced synthetic SAR images of these same vehicles under the same conditions. This raises a compelling challenge in determining how synthetic data, implicitly labeled by its creation, could assist in making predictions about unlabeled real-world data. It became apparent that the synthetic data was quite distinct from the real world data, leading to extra difficulties [87]. Some efforts, however, include training fully supervised CNNs on just the synthetic data and then using that to label the unlabeled data [88, 89]. Other efforts include using generative adversarial neural networks (GANs) to augment the synthetic data to make it more similar to real world data, hopefully boosting its usefulness towards classifying real world data [90].

4.4 SAR Image Classification

For typical image classification problems, data is often abundant and images are easily interpretable by humans, enabling quick and accurate annotation for training datasets. SAR images, by contrast, may be easy to collect given the proper sensor arrays, but the signals received can be very difficult for humans to interpret. The returned SAR images will show signs of energy scattering [75] and various reflectances that leave most objects appearing as bright blobs. However, despite human difficulty in interpretation, there is meaningful information in the way the signals are reflected and received from the target. This leads to an interesting problem where machine learning methods may be able to succeed and outperform humans. The caveat is that it is difficult to provide the models with sufficient amounts of well labeled training data, indicating that the ideal model will be in the low label rate regime. This is the prime motivation for active learning being applied to the datasets. With active learning, we can attempt to maximize the classification accuracy of our models with minimal reliance on human experts to label information.

The components of the classification pipeline and active learning regime were discussed in Chapter 3 and here we utilize and combine the established methods to great effect. The method is described as follows:

1. **Train Feature Extraction Neural Network:** Choose an appropriate neural network

architecture and train an encoder to encode the images in a latent space that preserves image similarity.

2. Construct a Graph: Pass all of the data samples through the encoder network to obtain latent space representations. Use a sparse k nearest neighbors search in the latent space to construct a similarity graph with cosine similarity.

3. Perform Laplace Learning: Using the known labeled information, propagate the labels across the graph to the unlabeled nodes, with Laplace learning, generating predictions.

4. Query a Label: Using the model's predictions, apply an acquisition function to select a label from the unlabeled samples. Once labeled by a human expert, add it to the labeled set.

5. Loop Until Completion: Repeat steps 3 and 4 until sufficient accuracy is achieved or enough energy has been spent on labeling samples.

Although active learning can technically query from the testing set, we restrict the queried labels to be from the training set, for MSTAR, so that comparisons to other techniques and methods in the literature remain cohesive.

In order for 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 semi-supervised learning (SSL) methods. We have introduced the variational autoencoder and SimCLR architectures in Chapter 3, which will be the principle feature extraction methods used in this work.

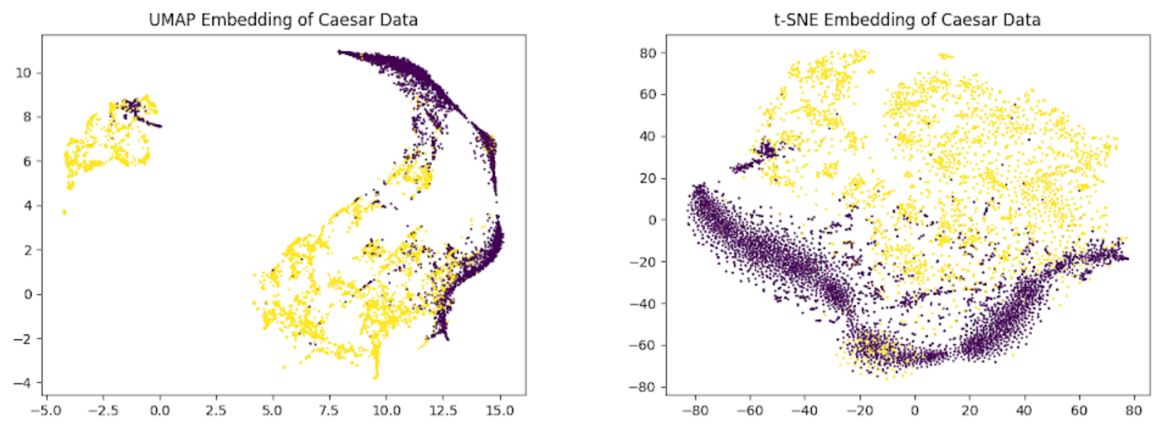
In this section, we build on previous work [2] in SAR classification by enhancing the feature extraction method. First, we begin with an application of the classification pipeline to the Caesar dataset using a variational autoencoder to showcase the general pipeline. Then, we develop a SimCLR framework for MSTAR images by introducing data augmentations that are specific to the dataset. 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 [2] and raw images. We evaluate the quality of the embeddings with support vector machines, spectral clustering, and graph cut energies. Secondly, we demonstrate 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. [2].

4.4.1 Caesar Results

In order to run image classification on the newly generated dataset, we train a variational autoencoder to encode the data, build a similarity graph on the encoded data, and then perform graph based semi-supervised learning within an active learning loop. For this example, we opt for the more established variational autoencoders over a SimCLR architecture, as this is a new dataset and we aim to demonstrate proven effectiveness [2]. For the autoencoder, we use an encoder that has four convolutional layers alongside max pooling and ReLU layers. Then, we apply the reparameterization trick, representing the embeddings as a mean and standard deviation. Finally, the decoder also has four convolutional transpose layers, mirroring the encoder architecture.

After training the variational autoencoder, latent space representations of the Caesar images are generated. Figure 4.6 shows the two dimensional visualizations of the learned latent space embeddings with Figure 4.6(a) showing a UMAP visualization and Figure 4.6(b) showing the t-SNE visualization. The purple samples represent ship samples and the yellow samples represent non-ship samples. We note that the data is well separated, which is likely indicative of distinct data classes and implies that a semi-supervised learning model should be able to distinguish between the classes with relatively high accuracy.

We construct a graph using the latent features with similarity function 3.4 using the cosine



(a) (b)

Figure 4.6: Two low dimensional embeddings of the VAE embedded Caesar dataset. (a) The UMAP visualization. (b) The t-SNE visualiation. The purple samples correspond to the class of ships and the yellow samples correspond to the class of non-ships.

distance metric, and use $k = 50$ as the number of nearest neighbors. With the constructed graph, we randomly acquire one label from each class. Then, we begin the active learning training loop with Laplace learning as the graph-based semi-supervised learning method. We compare various acquisition functions, many of which are discussed in Section 3.6. The results of running active learning are shown here in Figure 4.7. We observe that model change with variance optimality performs the best, but all of the acquisition functions show strong performance over the random acquisition method, reaching around 90% accuracy very quickly.

4.4.2 MSTAR Results

For the MSTAR dataset, previous work [2] showed very strong classification results using a variational autoencoder and an active Laplace learning loop. To improve upon this work, we sought to utilize more modern, state-of-the-art feature extraction techniques to enhance the quality of latent embeddings. In this section, we compare embeddings from the varia-

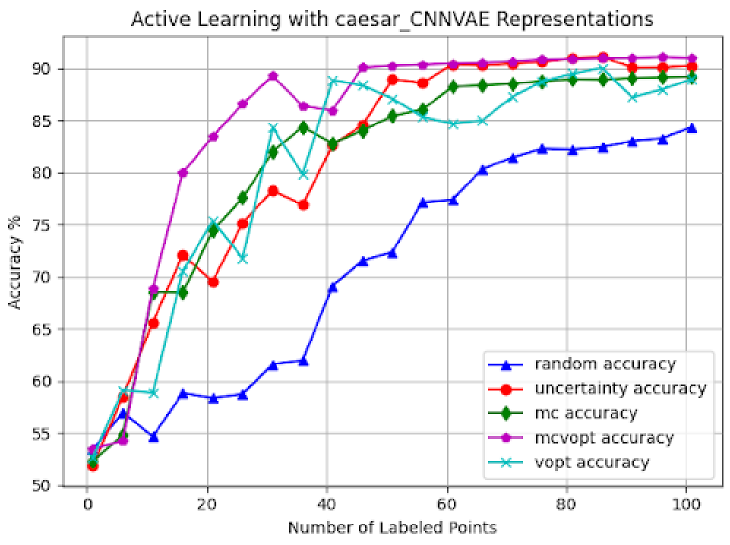


Figure 4.7: Results of running the active learning pipeline on the processed Caesar dataset. Multiple acquisition functions are shown in the legend.

tional autoencoder used in [2] with a SimCLR framework by examining various properties, visualizations, and classification results from the embeddings.

SimCLR [41] frameworks largely depend on the augmentations employed in them. If the augmentations are too harsh (e.g. total image corruption), then the network is forced to learn from noise and undesirable artefacts of the transformation. If the augmentations are too minor, then the network does not generalize well. Care must be taken to ensure that the augmentations used for a given dataset or task are meaningful and mimic unseen data. SimCLR frameworks for standard image classification (e.g. ImageNet, Cifar10) typically use color jitter, random cropping, random horizontal or vertical flips, and random blur. Here, we consider the peculiarities of SAR images and the MSTAR dataset, assess which augmentations are suitable and unsuitable, and propose custom augmentations which we ultimately use for training an encoder.

First, 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 [41] 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 standard color 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 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, which would affect the symmetry in the chips. This led to the implementation of 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 two allows for several max-pooling CNN layers, granting the encoder greater capacity for generalization 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 with a 7×7 kernel and a random sigma value randomly selected between 0.1 and 2.0 for each augmentation.

The seminal SimCLR paper [41] demonstrated that a standard ResNet [91] architecture is well-suited for the encoder network and recommends using a 2-layer perceptron for the projection head. However, Chen's later work [92] recommends using a slightly deeper projection head. Accordingly, we use a 3-layer perceptron instead of a 2-layer perceptron. SimCLR [41] uses a deep ResNet50 for training on ImageNet data. In order to have large batch sizes under memory constraints and prevent overfitting, we opted for the lighter ResNet18 model on MSTAR as well as the aforementioned downsampling to 32×32 resolution.

All code was implemented in Python. The source code to replicate our experiments and evaluation may be found on our GitHub repository[1]. The models were implemented using the PyTorch package, the graph learning and active learning methods were implemented from the GraphLearning Python package [65], and the SimCLR PyTorch implementation was adapted from SupContrast (Supervised Contrastive) GitHub[2] [93]. 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. [2], which is available on their GitHub[3], and the specifics of which can be seen in their paper.

We assess embedding quality of our SimCLR ResNet18, Miller et al. [2]'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.

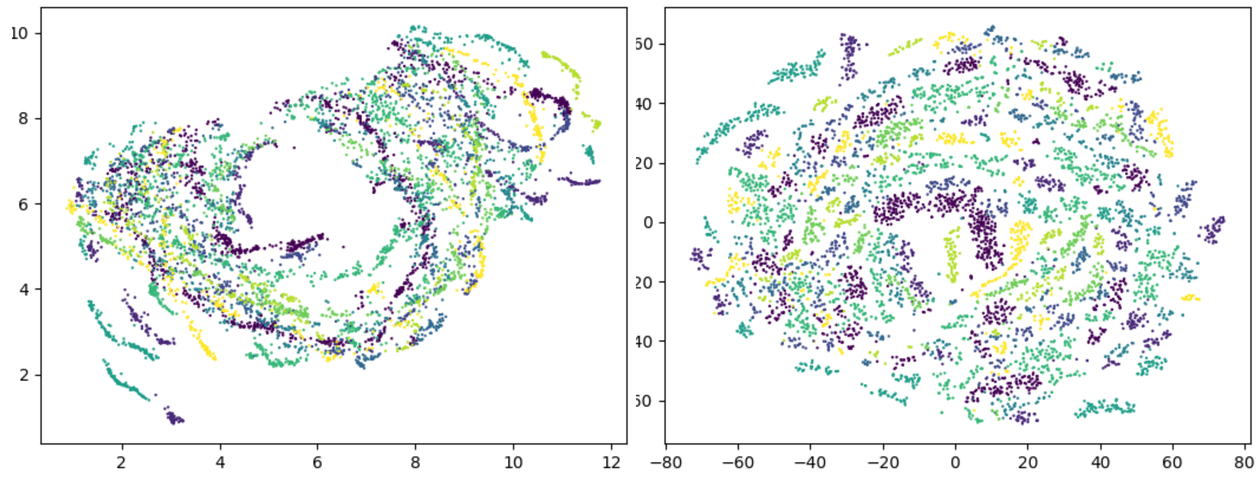
First, we compare UMAP [49] and t-SNE [47] visualizations of the raw image, VAE, and SimCLR embeddings shown in Figures 4.8, 4.9, and 4.10, respectively. The embeddings from the raw, unprocessed images show some class structure but are very noisy. Notably in the VAE embeddings, the t-SNE embedding shows the classes are generally well separated with little mixture between labels, but most of the labeled clusters appear disjointed from their respective class. In contrast, the SimCLR t-SNE and UMAP visualizations in Figure 4.10 reflect much more cohesion within the labeled clusters. There is very little mixing and the labeled clusters are very well connected with their respective classes. We would expect the SimCLR embeddings to be much better for graph construction.

We now consider the graph cut energy and spectral clustering accuracy. We take the

[1]https://github.com/jasbrown96/Contrastive-Active-Learning

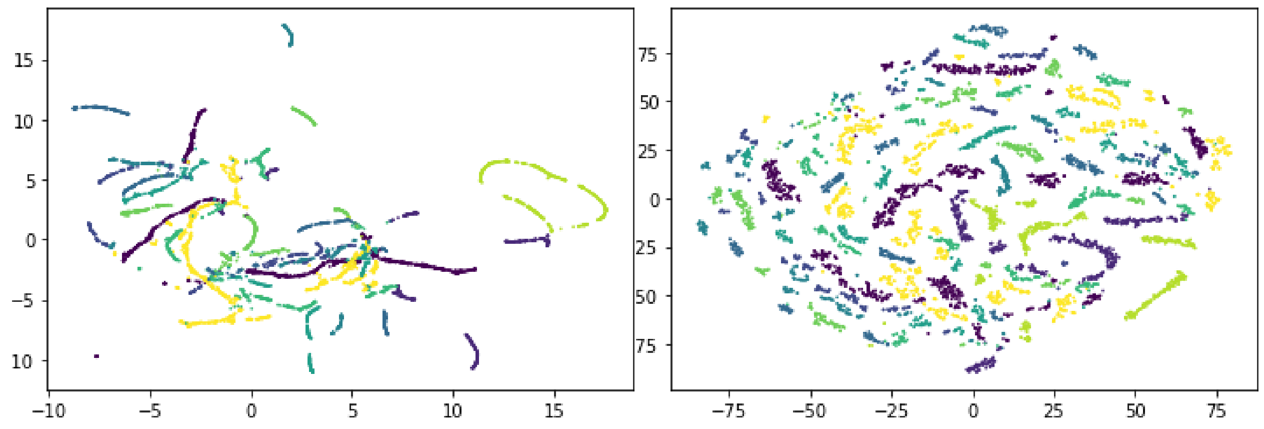
[2]https://github.com/HobbitLong/SupContrast

[3]https://github.com/jwcalder/MSTAR-Active-Learning



(a) (b)

Figure 4.8: Visualizations for the raw, un-embedded images. (a) The UMAP visualization of the data. (b) The t-SNE visualization of the data.



(a) (b)

Figure 4.9: Visualizations for the VAE encoded data. (a) The UMAP visualization of the data. (b) The t-SNE visualization of the data.

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 $k = 20$:

$$W_{ij} = e^{\frac{-4\|x_i - x_j\|}{d_k(x_i)}}$$

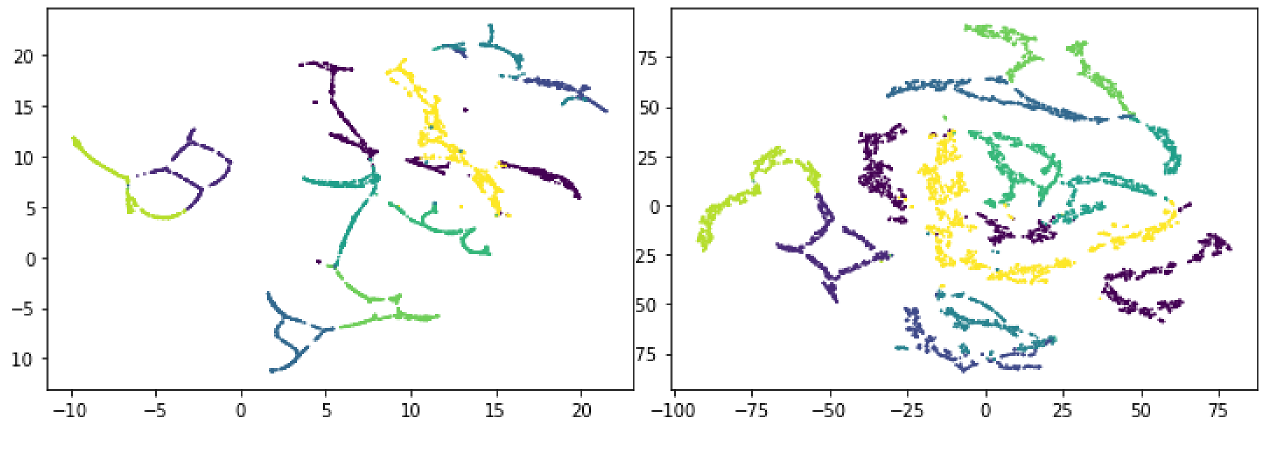

(a) (b)

Figure 4.10: Visualizations for the SimCLR encoded data. (a) The UMAP visualization of the data. (b) The t-SNE visualization of the data.

where $d_k(x_i)$ 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 captures clustering. The GCE can be computed as

$$GCE(W, U) \equiv \mathrm{tr}(U^T LU)$$

where L is the graph Laplacian built according to Section 3.5 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, while 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 [50]. Numerous methods of normalization exist; we opt to use Ng-Jordan-Weiss normalization [94, 50]. Let D be the diagonal matrix with $D_{ii} = \sum_{j\neq i} W_{ij}$. The algorithm uses the symmetrized Laplacian:

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

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

Table 4.1: 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 $k = 20$.

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 for each $\tilde{U}[i, :]$ to determine the cluster of the corresponding sample x_i [50]. 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 4.1, 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 support vector machines (SVM). The SVM fitting was performed across a range of train/test split ratios, from 10 to 3600 labeled points. 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 partitions were consistently applied to the SimCLR embeddings, VAE embeddings, and raw images in each instance. Figure 4.11 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 embeddings and raw images. The SimCLR curves represent the average SVM performance across 21 distinctly trained models (the 50 partitions are applied to each, averaged for each model, then averaged overall). Clearly, the SimCLR embeddings outperform both 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 outperforms the 500 epochs SimCLR models, more notably at low label rates, but the performance is quite similar overall.

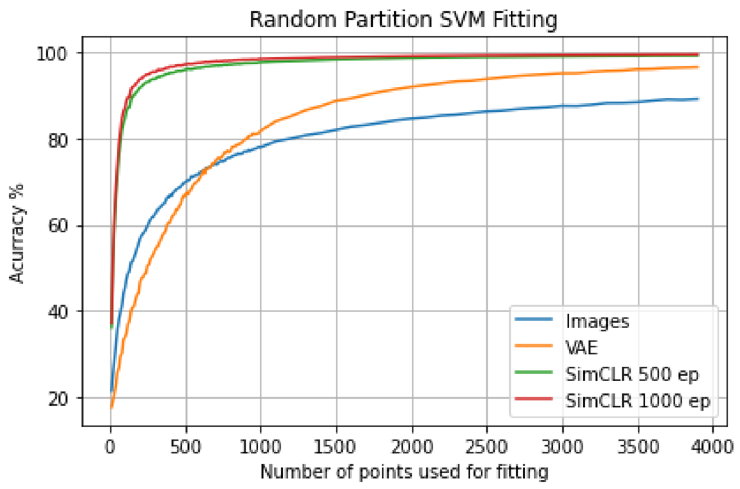


Figure 4.11: Different training/testing splits for fitting SVM classifiers on SimCLR, VAE, and raw image embeddings, averaged over 50 random partitions for each split, partitions randomly generated (susceptible to large class imbalance/underepresentation at low label rates).

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 4.12, which compares the VAE embeddings, 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.

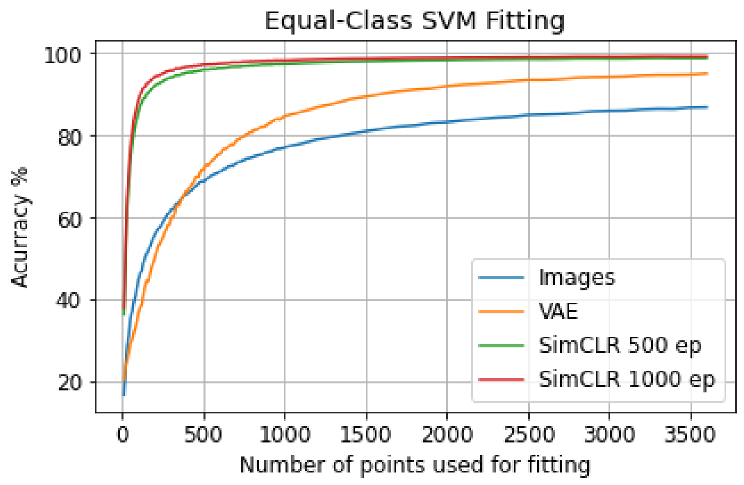


Figure 4.12: Equal class SVM fitting (same number of points used for each class in fitting) across different training/testing splits on SimCLR, VAE, and raw image embeddings, averaged over 50 random partitions for each split.

The t-SNE and UMAP visualizations of the embeddings, shown in Figures 4.8, 4.9, and 4.10, 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 verify, 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 4.13 and the results with the VAE embeddings are shown in 4.14. 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. 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), which is 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 4.15 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.

4.5 Conclusion

As demonstrated in Section 4.4.2, 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 embeddings, with greater local homogeny and connectedness. Combined with graph-based active learning, very few labels are necessary to achieve remarkable accuracy using the SimCLR embeddings; state of the art classification accuracies happen with far less labeled data required, compared to the VAE embeddings. This classification pipeline is able to succeed and excel at learning SAR data, which is a notable accomplishment. One particular interest for future work is that the SimCLR framework is amenable

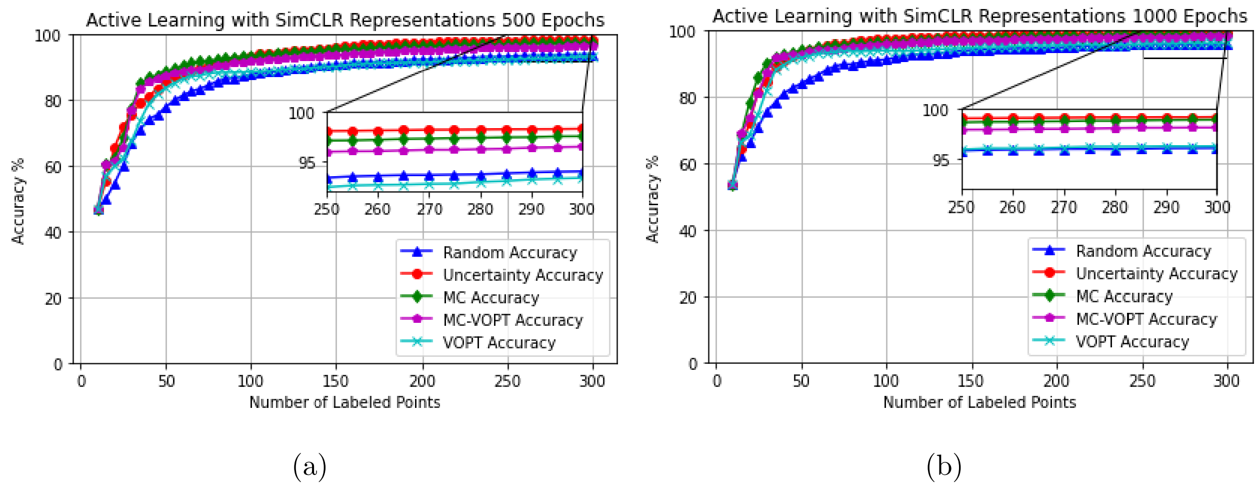


Figure 4.13: Accuracy of active learning with Laplace semi-supervised learning on SimCLR embeddings. (a) The encoder trained to 500 Epochs. (b) The encoder trained to 1000 Epochs. The results displayed are averaged across 21 distinctly trained models with active learning applied to each model individually. 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%.

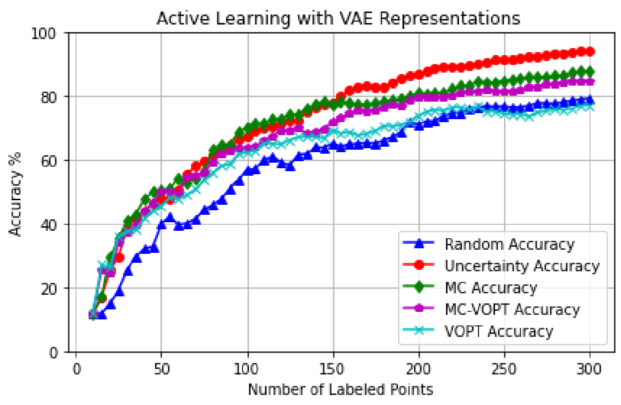


Figure 4.14: Accuracy of active learning with Laplace semi-supervised learning on the VAE embeddings with the pretrained weights from Miller et al. [2]. With 300 labels, the highest accuracy achieved is 94.2%.

to fine-tuning the encoder network over labeled data [92] in such a way that additional labeled data could lead to stronger embeddings, as well as strong machine learning models.

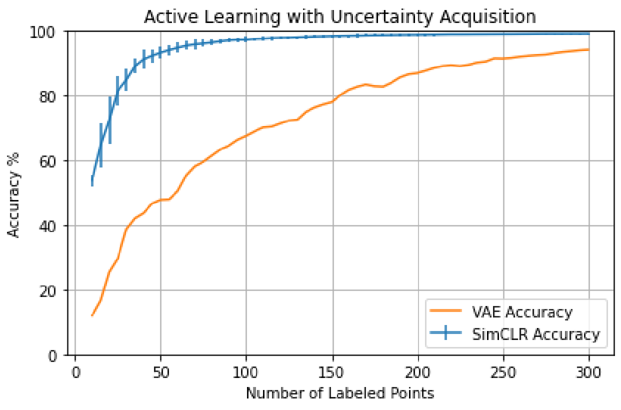


Figure 4.15: Direct comparison between the graph based active learning performance with the SimCLR embeddings against the VAE embeddings. The SimCLR embeddings are trained to 1000 epochs and averaged over 21 distinctly trained models, with the vertical bands corresponding to one standard deviation in accuracy.

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.

CHAPTER 5

Deep Learning Techniques and Applications to Hyperspectral Imagery

5.1 Introduction

In this section, we explore Hyperspectral pixel classification for identifying recyclable material within waste. This work is largely adapted from ”Material Identification in Complex Environments: Neural Network Approaches to Hyperspectral Image Analysis” published in the 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS) in Athens, Greece © 2023 IEEE [5]. Additional content is provided in this thesis regarding the background of hyperspectral imaging and processing of the data. The work was done in collaboration with Adrien Weihs, Bohan Chen, Harris Hardiman-Mostow, Jocelyn Chanussot, and Andrea Bertozzi. Several approaches to data exploration were considered. Bohan Chen utilized hyperspectral unmixing methods to separate the data while Adrien and Harris used PCA and statistical approaches. I manually sampled pixels from the training data by hand and used empirical features of the data to separate and pre-process the data into classes. For methodology, all methods utilized some form of feature extraction with a classification head. I explored contrastive learning methods such as SimCLR while Harris and Bohan used autoencoders. This work fits with the theme of data classification with limited labels due to the fact that the dataset is comprised of only three images. Our work is pixel based, and while there are a relatively large amount of pixels in each image, the limited variation reduces their viability for training large models.

5.1.1 Hyperspectral Data Review

Hyperspectral imaging utilizes specialized sensors to detect frequencies beyond the standard visible spectrum. Whereas the visible light spectrum wavelengths range from approximately 380nm to 750nm, hyperspectral sensors are able to detect light wavelengths up to 1700nm. For normal images captures taken by standard, color cameras, there are three channels to each image: red, green, and blue. For hyperspectral cameras, there are significantly more channels in each image, often including hundreds of frequencies that the camera detects. This leads to very high dimensional data and very smooth spectral signals [95]. A hyperspectral image is often compared to a hyperspectral cube where the first two dimensions correspond to the width and height of the image and the third dimension corresponds to the spectral signature of each pixel. Examining a single pixel in such a hypercube would result in a spectral band, showing the amount of absorbance and reflectance of the respective wavelengths.

The interactions with certain wavelengths of light has a strong correlation to the chemical composition of the region being imaged. This leads to a very wide range of applications, including agricultural purposes such as detecting grape cultures in a vineyard [96] or assessing fruit and vegetable quality [97], and also even biomedical imaging applications [98]. In this section, we consider the problem of identifying recyclable plastic material within waste, since the spectral information could meaningfully discern between desirable and undesirable waste.

Recycling of waste material is an important component of the United Nations' Sustainable Development Goals [99]. Chemometric study of spectroscopy data obtained through hyperspectral imaging (HSI) has proven to be a popular framework in quality sorting and recycling tasks, such as classification of polymers [100, 101, 102, 103], classification of food samples [104], contamination identification for plastic recycling [101], and waste identification in copper ore processing [105]. Similarly, we are interested in identifying plastic samples in a tray of waste materials of various chemical compositions. In particular, we want to perform pixel-wise classification to detect plastic on the HSI data of Figure 5.3(b), which contains plastic among other materials.

The novelties of this work are three-fold. First, while most of the mentioned papers consider spectrum data in the near infrared (NIR) wavelengths (900nm to 1700nm), we aim to compare classification results obtained from the NIR and visible (VIS) wavelengths (400nm to 1000nm). To this end, we present results on images collected from two different hyperspectral cameras, which image in the NIR and VIS wavelengths respectively. From an industrial point of view, such results can assist in the choice of the image acquisition setup.

Second, the spectroscopy data in the referenced papers is acquired in a very controlled environment: the samples are well-separated and placed on a uniform white background, allowing for straight-forward analysis of the spectral data. Furthermore, the typical framework assumes expert knowledge of the materials and thus precise definition of the classes in the classification task. However, in our dataset, samples are cluttered on a piece of cardboard with various interfering, partly overlapping elements and more realistic, varied lighting. The dataset is also only vaguely labelled, which leads to a broad definition of the plastic class (see Section 5.2 for details). The resulting intraclass variability presents a challenge to traditional chemometric classification approaches. This also reflects the more realistic industrial setting where recycling material is placed directly on a conveyer belt without pre-arrangement, manual or chemical sorting, and tedious pixel-wise labelling.

Third, chemometrics in most of the spectroscopy literature are based on linear methods (see Section 5.3). Recently, there has been more deep machine learning methods used in these applications [102, 103, 106]. We continue this trend by presenting novel results using contrastive learning, as well as autoencoders combined with graph learning. To the best of our knowledge, this has not previously appeared in the literature for this task.

The rest of the chapter is structured as follows: in Section 5.2, we describe our dataset, data calibration and data exploration; in Section 5.3 we survey our methods for the classification task; in Section 5.4 we discuss our results; in Section 5.5 we summarize our findings and offer future avenues of work.

5.2 Dataset

The dataset consists of six images derived from three scenes imaged by two cameras: Specim FX10 and Specim FX17. The imaging setup is shown in Figure 5.1. A rail moves the camera while two sources of light illuminate the scene to minimize shadows. The study by Morales [107] appears to describe very similar imaging conditions and also uses the same camera models, so we refer the reader to this source for further experimental setup details regarding the cameras.

The Specim FX10 camera operates in the visible wavelengths (400nm to 1000nm) and the Specim FX17 camera operates in the near infrared wavelengths (900nm to 1700nm). FX10 captures 448 evenly spaced wavelengths and FX17 captures 224 evenly spaced wavelengths. The cameras both use a push broom action and are set up as a row of sensors that can be scanned across the materials. The two cameras imaging the same scenes allows us to compare the classification results between the different wavelengths.

The three scenes contain two training images and one testing image. The different materials within the image have been very roughly labelled. The first training image, Figure 5.2, contains plastic samples on a white, paper background. For FX10, this image is 700×400; for FX17, this image is 730×320. The second training image, Figure 5.3(a), contains cluttered scraps of non-plastic material in a cardboard box. The non-plastic materials include copper, fabrics, stones, paper, and metal. For FX10, this image is 1120×570; for FX17, this image is 1125×480. The testing image, Figure 5.3(b), contains the same cluttered scraps as the training image, Figure 5.3(a), but with the addition of plastics. For FX10, this image is 1135×580; for FX17, this image is 1130×480. Despite the images sharing some samples, the different imaging conditions help our methods to prevent significant overfitting and allow our results to still reveal meaningful insights about the data and optimal classi-

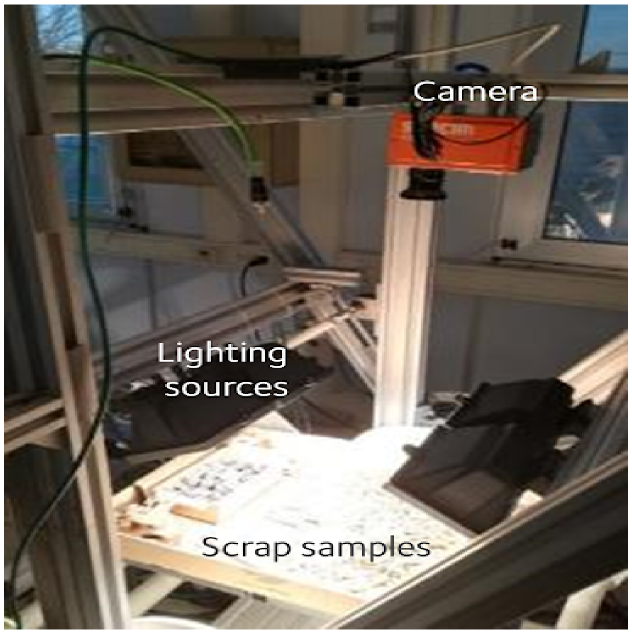


Figure 5.1: The imaging apparatus. Two light sources are used for illumination to minimize shadows on the scrap samples and the camera is attached to a rail. © 2023 IEEE

fication methods. As we intend to work with pixel-level data, these large images supply a very large amount of pixel information. If we are able to successfully separate the pixels by class, then supervised learning methods will become viable. However, as many of the pixels are expected to have extremely similar spectral information, we effectively consider this to be a limited information case and consider methods that are robust to overfitting.

5.2.1 Calibration

To calibrate the Specim cameras, we are provided with a dark image and a reference image. The dark image is taken when the lens is closed and captures the noise inherent to the sensors in the camera. The reference image contains materials with high reflectance which are used to normalize the measurements with respect to the lighting source. Using the dark and reference images, we calibrate our images via the industry standard equation, consistent with Arnold's work on hyperspectral imagery for industrial sorting applications [100].

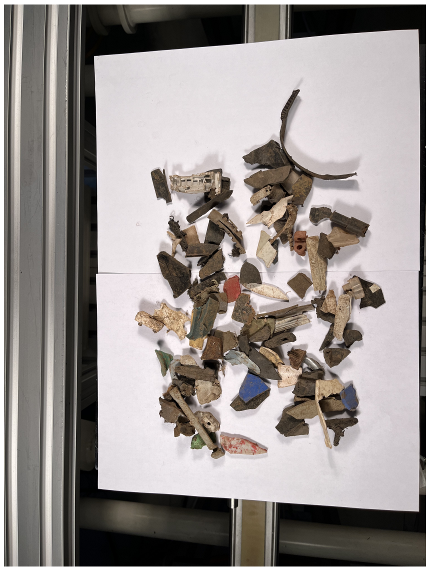


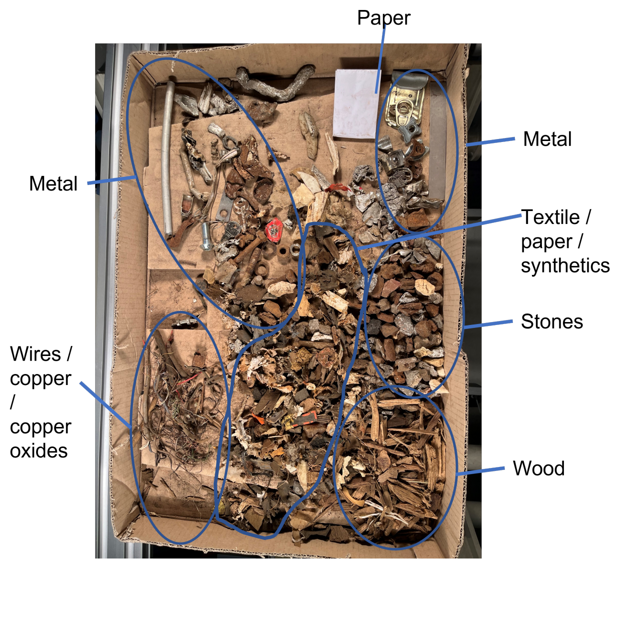
Figure 5.2: A controlled environment of various types of plastic on a sheet of white paper. The plastics have various colors and spectral signatures. © 2023 IEEE

First, we average the spectral intensity across each pixel in the dark image and will subtract that value from all of the camera's measurements. This removes biases in the instrument measurements, normalizing the data for further processing. The reference image contains a piece of paper with two materials on it, denoted as Ref 50 and Ref 99. The Ref 50 material and Ref 99 material reflect 50% and 99% of the light that is incident on the material, respectively. After subtracting the dark intensity from the images (including the reference), we then divide by the average spectrum from the Ref 99 material, which normalizes the observed spectrum in the data by the lighting used in the laboratory. We do not have information about the Ref 99 reflectance for all of the sensors, as only a patch is imaged, and so we must extrapolate and will take a mean over the Ref 99 pixels we do have.

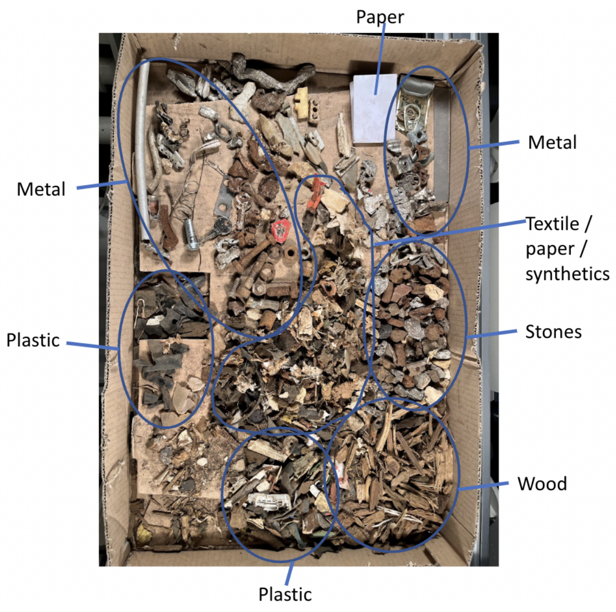
We define the spectrum for Ref by the following

$$Ref(\lambda) = \frac{\sum_{(i,j)\in A} I_{Ref}(i,j,\lambda) - I_{Dark}(i,\lambda)}{|A|}$$

where I_{Ref} is the image containing the reference materials, I_{Dark} is the image from covering the lens, the first coordinate (i) corresponds to the sensor axis and A is the set of x,y



(a)



(b)

Figure 5.3: Two scenes of mixed clutter. The materials present are annotated on the figures directly. (a) Contains no plastic and is used for sampling the clutter materials. (b) Contains plastic and is used for testing. © 2023 IEEE

coordinates corresponding to the desired reference material in the image. Simply put, this process subtracts the dark measurements from the reference image and then averages bands across the desired reference spectrum to acquire a single band. In order to calibrate one of the images in the dataset, we perform the following

$$I(x,y,\lambda) = \frac{I_{RAW}(x,y,\lambda) - I_{Dark}(x,\lambda)}{Ref(\lambda)}$$

where I_{Raw} is the uncalibrated image. Dividing by the reference value for the images helps to normalize for the intensity and presence of the lighting used in the laboratory environment.

5.2.2 Data Preprocessing

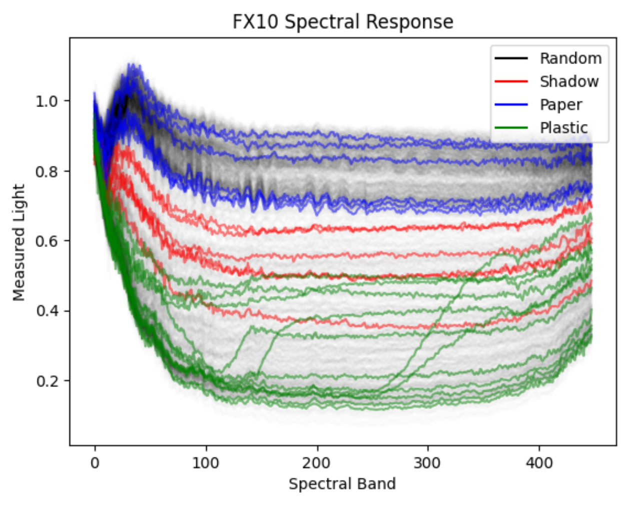
Light scattering can cause significant variability in the captured spectra both in diffuse reflectance and transmittance spectroscopy. For our application, this is particularly relevant: diffusively reflected light not only contains information about the chemical content of our samples but also about its micro-structure which causes the scattering (e.g. surface roughness, density fluctuations) [108]. The latter can be modeled using physical models and produces both multiplicative and additive interference in the spectra [109].

While deep learning methods are capable of dealing with this variability, for classical linear baseline methods and data exploration (refer to Sections 5.2.3 and 5.3), preprocessing of the spectra is necessary. Preprocessing is highly specific to the dataset [101, 110]. After considering multiplicative scattering correction, standard normal variate and spectral derivative methods (see [108] for a review), we decided to use the Savitzky-Golay (SG) filter [111]. Using a moving window, the filter applies smoothing to the spectra by performing polynomial approximation. This approximation allows one to take derivatives of the spectra easily. We write SG-0 for the smoothed spectra and SG-1 for the first derivative of SG-0. Due to the moving window, both the first and last bands of the spectrum can be discarded.

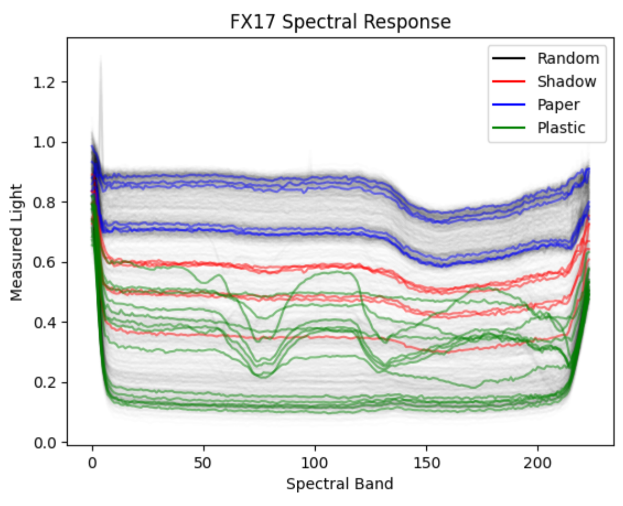
5.2.3 Data Exploration

In order to use supervision for pixel classification of the testing image, Figure 5.3(b), we need to define regions of interest (ROIs) to train and evaluate our algorithm on. To obtain pure plastic samples, we remove the background of the plastic image, Figure 5.2, by thresholding on specific bands [104]. For non-plastic samples, we use all of the pixels in Figure 5.3(a) as negative training samples, as the scene does not contain any plastic. We also note that removing the background in Figure 5.3(b) and defining a ROI is significantly harder than in some of the datasets considered in [100, 101, 104] due to its complexity. Using only an unmixing method on materials with high intraclass variability - such as in this recycling setting - may fail to identify ROIs with plastic. To avoid excluding plastic pieces from the ROI, we consider the whole scene as our testing ROI. This also makes our methods applicable to real-world settings where defining an ROI may not be realistic.

By manually sampling the plastic ROI in Figure 5.2, we observe that the plastic pieces



(a)



(b)

Figure 5.4: Spectral responses of manually selected pixels from Figure 5.2. The black spectra are randomly selected pixels that are used to reflect pixel density. The red spectra are shadows of the plastic on paper. The blue spectra corresponds to the background paper. The green spectra corresponds to plastic pixels. (a) FX10. (b) FX17.

may be composed of different polymers. This implies large intraclass variabilty in the plastic and is similar to practical applications of recycling, where one might be interested in various plastics types (and potentially in plastics with chemical compositions not seen in the training dataset). This motivates deep learning methods, whose larger capacity can capture intraclass variability more accurately than classical methods. The large variability in our classification task is in contrast to the usual setting explored in [100, 101, 104] where the target classes are known in advance and well-defined.

In Figure 5.4, we see respective spectral bands for pixels selected from Figure 5.2 for the FX10 camera, Figure 5.4(a), and FX17 camera, Figure 5.4(b). The colors distinguish between the material being imaged. Notably we see that paper reflects the most amount of light and has the largest signature whereas plastic will absorb more light and thus appear darker. The shadows have similar spectral bands to the paper but are less bright. Amongst the plastic spectral lines, we have sampled plastic of various colors which accounts for the significantly varied spectral lines, referred to previously as the intraclass variability. The black, mostly transparent bands correspond to randomly selected pixels in the image. This is intended to give a reference for the density of certain materials in the image. For example, the darkest regions fall within the blue bands and suggest that paper is the most prominent material in the image. We do still see black bands in the regions corresponding to plastic indicating more variability in the class. In order to isolate the plastic pixels from Figure 5.2, we use the information from these spectral bands to derive thresholds. More specifically, for FX10 pixels we threshold along the 40^{th} band where we empirically witness separation from the plastic and shadow spectra. For the FX17 camera, we see that the shadow spectra is more difficult to discern from the plastic so we instead use the standard deviation of the spectra alongside some additional thresholding to isolate those pixels.

5.3 Methods

In this section, we survey the different methods used for the detection of plastic in the mixed image. The methods vary from linear to highly nonlinear. We first describe Partial Least Squares Discriminant Analysis (PLS-DA) which we use as our baseline. Following the motivations detailed in Sections 5.2.2 and 5.2.3, we also explore deep learning methods. In particular, we perform dimensionality reduction through Autoencoders (AEs) and Contrastive Learning (CL); our classifiers include Multilayer Perceptrons (MLP), k-Nearest Neighbors (k-NN) [21], and Graph Learning (GL) [18]. Unsupervised autoencoders with semi-supervised graph learning classifiers were chosen as the low-label rate model while the supervised contrastive learning with k-NN and separate MLP classifier were chosen as the supervised methods.

5.3.1 Partial Least Squares Discriminant Analysis

We refer to [112] for a review of the methods discussed in this section. A standard assumption in multivariate linear regression is that the variables are uncorrelated. This is not satisfied when using spectra and we therefore aim to transform our data into a small number of orthogonal vectors. Unsupervised principal component analysis achieves this by projecting data onto principal components which point in relevant directions based on the variance of the spectra. Performing regression using the principal components as variables is called principal component regression (PCR). The supervised counterpart to PCR is PLS-DA: the directions of the orthogonal vectors used in the regression problem now maximize covariance with the response variable. Regression yields a continuous response prediction and we threshold at 0.5 to obtain labels in $\{0, 1\}$ (for binary classification). PLS-DA is the standard classification method used in spectroscopy [104, 101, 100] and in our experiments, we use three orthogonal vectors for PLS-DA.

5.3.2 Nonlinear Dimensionality Reduction and Classifiers

As explained in Section 5.3.1, dimensionality reduction is essential when dealing with spectral data. We now detail two deep learning approaches for this task.

We utilize an autoencoder with an input dimension of 448 or 224 (for FX10 or FX17, respectively), followed by fully connected layers with output dimension 100, 25, and 5, respectively, before fully connected layers with output dimension 25, 100, 448, respectively. Rectified Linear Unit (ReLU) activation functions are used after each layer except at the bottleneck. We train the network for 50 epochs. We use the Adam optimizer [113] with a learning rate of 0.001, and β values of 0.9 and 0.999. The loss is mean squared error. To classify the pixels, we perform graph learning (see Section 3.5) on the embeddings learned by the AE.

Additionally, we use SupCon [46], a fully supervised contrastive learning with the goal of learning latent representations for samples so that classes are well clustered. Typically when using contrastive learning, many data augmentations are used to create more varied data and teach the neural network to be invariant to certain effects. Due to the spectral nature of the data, it is less clear which augmentations are natural to use. Moreover, we have an abundance of pixels, so we instead do not use any augmentations and rely on the noise in the pixels to provide natural variation.

We train a simple multilayer perceptron encoder network with input dimension of 448 or 224 (for FX10 or FX17 respectively) followed by two linear layers with sizes 256 or 128 (for FX10 and FX17 respectively) and 64 with ReLU activation functions. We then have a dropout layer with dropout probability of 50% and then one more linear layer of size 16 with a sigmoid activation function. The projection head is a linear layer of size 4 with a normalization activation function. We train the models for 500 epochs using the Adam optimizer with a learning rate of 0.001, beta values of 0.9 and 0.999, and a gamma value of 0.99 for the scheduler. For the SupCon loss function, we use a temperature of 0.1.

Many of the non-plastic samples have identical latent representations with this architecture, which makes graph learning ill-posed when the number of nearest neighbors is not large enough. For classification, we instead opt to use a k-NN classifier with $k = 5$ for the contrastive embeddings. For the autoencoder trained embeddings, the Laplace learning methodology described in Section 3.5 is used. For our experiments, we use the cosine distance, set $k = 10$, and we use 1000 non-plastic samples and 500 plastic samples as training points to classify the pixels. As a separate classifier, we train a 3-layer fully connected neural network for binary classification on the data. The network has hidden layers of size 50 and 10 with ReLU activation functions. We use cross entropy loss. The network uses the same hyperparameters as the AE network.

5.4 Results

We present classification results on both FX10 and FX17 data in Figure 5.5. Since we have no pixel-wise ground truth, our evaluation is qualitative. In Figure 5.5, the background displays the hyperspectral images captured by the respective cameras, with pixels classified as plastic highlighted in yellow. The right most figure showcases the ground truth regions of interest. As we were not provided pixel-based annotations for the testing environment, we simply must intuit the qualitative results by the amount of overlap with the three marked regions of interest at the top, left, and bottom of the image.

We see that the detection results from the spectra in the FX17 camera appears to be much more confined to the regions of interest and therefore stronger. For the FX17 data, all of the methods are able to detect the plastic at the top and bottom of the image frame, but the linear method notably misses the plastics on the left-most region. All of the deep learning methods successfully detect plastics on all three regions but have varying degrees of sensitivity. The AE with graph learning appears to have more noise than the other methods but does detect the most amount of plastic. The contrastive learning embeddings with k-nearest neighbors classifier and the multi-layer perceptron classifer have more balanced results.

For the FX10 images, the results are notably poorer. Only the AE with graph learning is able to detect the plastic on the left-most region of interest but also suffers from many false positives elsewhere. The contrastive learning with k nearest neighbors and the multi-layer perceptron have some ability to distinguish the plastic but also suffer from more false positives than the FX17 images.

Overall, it is clear that the deep methods outperform PLS-DA, the more classical method, in detecting additional plastic. As noted, some of the images suffer from noise in the results and the appearances of false positives, but many of these erroneous results could likely be filtered with post-processing techniques. For example, plastic-classified pixels should be

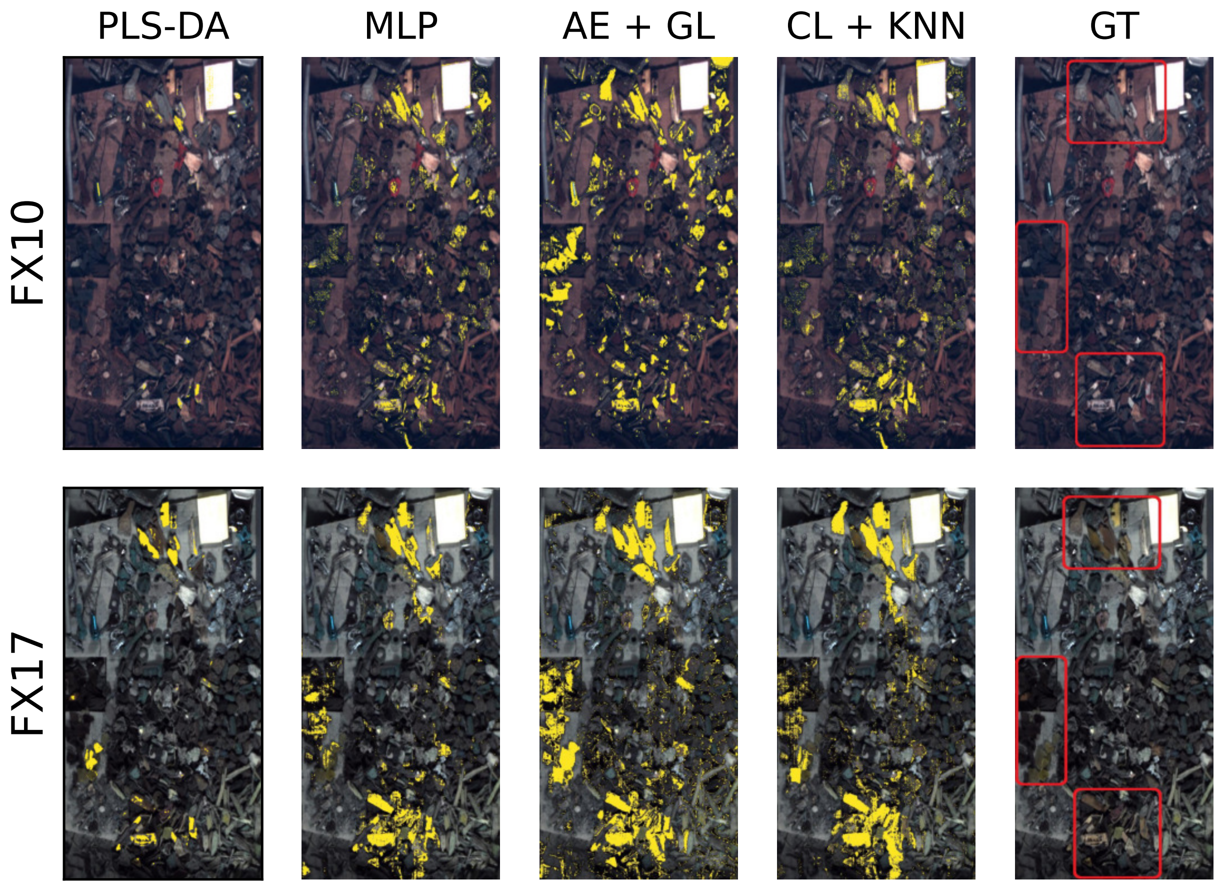


Figure 5.5: Binary classification on FX10 and FX17 images using four methods: PLS-DA (Partial Least Squares Discriminant Analysis), MLP (Multilayer Perceptron), AE + GL (Autoencoder and Graph Learning), and CL + k-NN (Contrastive Learning and k-Nearest Neighbors). In the first four columns, yellow pixels indicate a plastic classification. The last column shows the approximate "ground truth" areas where plastic was manually placed.© 2023 IEEE

densely grouped to correspond to a plastic object, so, reasonably, stray pixels should be filtered out.

5.5 Conclusion and Future Work

Using machine learning and artificial intelligence to automate the process of material sorting using hyperspectral cameras shows promise. We have demonstrated that, with relatively limited data, plastic can be detected readily with a variety of methods in both the visible and near infrared spectra. We find that classification results are stronger for images taken with the FX17 camera, indicating that the near infrared (900nm-1700nm) range is suitable for differentiating plastics from other non-plastic refuse, however the visible spectrum does appear to be viable as well.

It would be valuable to further explore the distinctions between the methods and light spectra with a meticulously curated, larger dataset containing expert knowledge of the types of plastics.

CHAPTER 6

Conclusion

In this thesis, we reviewed many different techniques for low label rate image learning in a variety of application environments. Many classical image processing techniques were introduced in Chapter 1, including linear, signal processing, and variational methods. In Chapter 2, we were able to utilize some of these methods. For example, the Canny edge detector is the first step in the detection of the largely transparent particles. We also introduced additional, classical techniques. The Hough Transforms that follow the edge detector are well-known methods for template detection that have lower requirements than neural networks and snake active contours are a variational, adaptive method for segmenting the particles that is sensitive to local, fine-grain features. With these methods we were able to leverage the specific geometry of the particles to facilitate an automated detection algorithm. Coupled with snake active contours and the resulting theory for for the convergence of the snakes, we established a viable method for the segmentation of the particles as well. The empirical results of the hand tuned algorithm on the small dataset show strong support for the methodology and have very good precision, which is ideal for the medical testing environment. Without relying on any deep learning methodology, this work can generalize well and run on limited hardware. Future work could involve utilizing more particle geometries, using a more efficient implementation of the generalized Hough Transform, or deriving more specific conditions on the particles to ensure convergence of the snake active contours.

Beyond the classical techniques, we also utilized many modern, more state-of-the-art neural network methods that worked very well with our classical pipeline. For example, the variational autoencoders and SimCLR architectures serve as a stronger, nonlinear upgrades over principal component analysis or non-negative matrix factorization for learning features in images. These methods couple very well with the transductive graph-based semi supervised learning, Laplace learning. The mathematical background of graph learning allows us to leverage an abundance of unlabeled information while using minimal labels. Further integrating these methods into an active learning pipeline can result in remarkable performance. All of these methods are detailed in Chapter 3.

With these established methods, we were able to consider more practical applications. To start, the work in Chapter 4 explores applications of the methodology to synthetic aperture radar data. We firstly introduced SAR data and discussed many of the domain relevant features of the data. Then we introduced two datasets: the Caesar dataset which required significant pre-processing to craft into a classification dataset, and the MSTAR dataset which is a classical benchmark dataset. Since both datasets consist of SAR images, they require expert knowledge to label, but can benefit by often having a large amount of data, much more so than what is considered in the particle detection problem. We show that the graph-based active learning pipeline from 3 achieves very strong results on both of these datasets and utilizing SimCLR for learning features on MSTAR improved over previous state-of-the-art results.

In the final chapter, Chapter 5, we move away from remote sensing SAR data to a close-range hyperspectral dataset. Similar to Chapter 4, we discuss many domain specific features of the dataset before processing the data. The dataset consisted of only six hyperspectral images across two cameras. This supplies a large amount of labeled pixel data relative to the other datasets, but the difficulty here stems from the lack of images provided. Despite there being a large number of varied pixel spectra to work with, all of the spectra comes from three images, indicating very similar imaging conditions. This can be acceptable for controlled environments where conditions are consistent, but may not be ideal in practical scenarios. However, we still use many of the established techniques to develop a variety of methods for pixel-based classification to segment the image. We show that the near infrared spectrum is well suited to the task and that deep learning tools are viable for the problem.

By combining classical and modern imaging techniques into a single pipeline, we are able to utilize as much information as possible in a given dataset, learn features, and propagate labels. The methods have seen great success and further work is being done to combine the methods to a higher degree.

REFERENCES

[1] Y. Wang, C. Wang, H. Zhang, Y. Dong, and S. Wei, "A sar dataset of ship detection for deep learning under complex backgrounds," *remote sensing*, vol. 11, no. 7, p. 765, 2019.

[2] K. Miller, J. Mauro, J. Setiadi, X. Baca, Z. Shi, J. Calder, and A. L. Bertozzi, "Graph-based active learning for semi-supervised classification of SAR data," in *Algorithms for Synthetic Aperture Radar Imagery XXIX*, E. Zelnio and F. D. Garber, Eds., vol. 12095, International Society for Optics and Photonics. SPIE, 2022, p. 120950C. [Online]. Available: https://doi.org/10.1117/12.2618847

[3] J. Brown, A. Arnheim, A. L. Bertozzi, and D. D. Carlo, "Detection and segmentation of shape-coded particles via hough transforms and snake active contours," in *2024 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI).* IEEE, 2024, pp. 85–88, © 2024 IEEE. Reprinted, with permission.

[4] J. Brown, R. O'Neill, J. Calder, and A. L. Bertozzi, "Utilizing contrastive learning for graph-based active learning of sar data," in *Algorithms for Synthetic Aperture Radar Imagery XXX*, vol. 12520. SPIE, 2023, pp. 181–195.

[5] J. Brown, B. Chen, H. Hardiman-Mostow, A. Weihs, A. L. Bertozzi, and J. Chanussot, "Material identification in complex environments: Neural network approaches to hyperspectral image analysis," in *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS).* IEEE, 2023, pp. 1–5, © 2023 IEEE. Reprinted, with permission.

[6] S. P. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[7] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition.* IEEE Computer Society, 1991, pp. 586–587.

[8] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.

[9] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[10] C. Van Loan, *Computational frameworks for the fast Fourier transform.* SIAM, 1992.

[11] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[12] J.-L. Starck, E. Pantin, and F. Murtagh, "Deconvolution in astronomy: A review," *Publications of the Astronomical Society of the Pacific*, vol. 114, no. 800, p. 1051, 2002.

[13] R. C. Gonzalez, *Digital image processing.* Pearson education india, 2009.

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[15] A. Rosenfeld, "Picture processing by computer," *ACM Computing Surveys (CSUR)*, vol. 1, no. 3, pp. 147–176, 1969.

[16] J. Calder, "The calculus of variations," *University of Minnesota*, vol. 40, 2020.

[17] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

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

[19] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[22] B. C. Csáji *et al.*, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, no. 48, p. 7, 2001.

[23] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5070–5079.

[24] G. Destgeer, M. Ouyang, C.-Y. Wu, and D. Di Carlo, "Fabrication of 3d concentric amphiphilic microparticles to form uniform nanoliter reaction volumes for amplified affinity assays," *Lab Chip*, vol. 20, pp. 3503–3514, 2020.

[25] G. Destgeer, M. Ouyang, and D. Di Carlo, "Engineering design of concentric amphiphilic microparticles for spontaneous formation of picoliter to nanoliter droplet volumes," *Analytical Chemistry*, vol. 93, no. 4, pp. 2317–2326, 2021.

[26] V. Shah, X. Yang, A. Arnheim, S. Udani, Y. Luo, M. Ouyang, G. Destgeer, O. Garner, H. Koydemir, A. Ozcan, and D. Di Carlo, "Amphiphilic particle-stabilized nanoliter droplet reactors with a multimodal portable reader for distributive biomarker quantification," *ACS Nano*, vol. 17, pp. 19 952–19 960, 2023.

[27] G. Destgeer, M. A. Sahin, L. van den Eijnden, and C. Bhiri, "Deep learning based recognition of shape-coded microparticles," *Frontiers in Lab on a Chip Technologies*, vol. 2, p. 1248265, 2023.

[28] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[29] S. Savkare and S. Narote, "Blood cell segmentation from microscopic blood images," in *2015 International conference on information processing (ICIP)*. IEEE, 2015, pp. 502–505.

[30] C. Zhang, X. Xiao, X. Li, Y.-J. Chen, W. Zhen, J. Chang, C. Zheng, and Z. Liu, "White blood cell segmentation by color-space-based k-means clustering," *Sensors*, vol. 14, no. 9, pp. 16 128–16 147, 2014.

[31] F. Al-Hafiz, S. Al-Megren, and H. Kurdi, "Red blood cell segmentation by thresholding and canny detector," *Procedia Computer Science*, vol. 141, pp. 327–334, 2018.

[32] P. V. Gulyaev, "Application of the hough transform to dispersion control of overlapping particles and their agglomerates," *Devices and Methods of Measurements*, vol. 14, pp. 199–206, 2023.

[33] V. Pătrăucean, P. Gurdjos, and R. G. Von Gioi, "A parameterless line segment and elliptical arc detector with enhanced ellipse fitting," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*. Springer, 2012, pp. 572–585.

[34] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[35] P. M. Merlin and D. J. Farber, "A parallel mechanism for detecting curves in pictures," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 96–98, 1975.

[36] L. D. Cohen, "On active contour models and balloons," *CVGIP: Image understanding*, vol. 53, no. 2, pp. 211–218, 1991.

[37] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[38] J. Ivins and J. Porrill, "Everything you always wanted to know about snakes (but were afraid to ask)," *Artificial Intelligence*, vol. 2000, 1995.

[39] B. Braden, "The surveyor's area formula," *The College Mathematics Journal*, vol. 17, no. 4, pp. 326–337, 1986.

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

[41] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations." PMLR, 2020, pp. 1597–1607.

[42] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

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

[44] ——, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, Nov. 2019, publisher: Now Publishers, Inc. [Online]. Available: https://www.nowpublishers.com/article/Details/MAL-056

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

[46] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *CoRR*, vol. abs/2004.11362, 2020. [Online]. Available: https://arxiv.org/abs/2004.11362

[47] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[48] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

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

[50] U. von Luxburg, "A tutorial on spectral clustering." *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/sac/sac17.html#Luxburg07

[51] B. Chen, Y. Lou, A. L. Bertozzi, and J. Chanussot, "Graph-based active learning for nearly blind hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

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

[53] J. Qin, H. Lee, J. T. Chi, L. Drumetz, J. Chanussot, Y. Lou, and A. L. Bertozzi, "Blind hyperspectral unmixing based on graph total variation regularization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 4, pp. 3338–3351, 2020.

[54] "ANNOY library," https://github.com/spotify/annoy, accessed: 2024-01-01.

[55] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[56] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nystrom method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[57] A. L. Bertozzi and A. Flenner, "Diffuse interface models on graphs for classification of high dimensional data," *Multiscale Modeling & Simulation*, vol. 10, no. 3, pp. 1090–1118, 2012.

[58] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. G. Percus, "Multiclass data segmentation using diffuse interface methods on graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1600–1613, 2014.

[59] J. Woodworth, G. Mohler, A. L. Bertozzi, and P. Brantingham, "Non-local crime density estimation incorporating housing information," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2028, p. 20130403, 2014.

[60] J. Calder, B. Cook, M. Thorpe, and D. Slepčev, "Poisson Learning: Graph based semi-supervised learning at very low label rates," *Proceedings of the 37th International Conference on Machine Learning, PMLR*, vol. 119, pp. 1306–1316, 2020. [Online]. Available: http://proceedings.mlr.press/v119/calder20a.html

[61] K. Miller and J. Calder, “Poisson reweighted Laplacian uncertainty sampling for graph-based active learning,” *arxiv:2210.15786*, 2022.

[62] M. Flores, J. Calder, and G. Lerman, “Analysis and algorithms for Lp-based semi-supervised learning on graphs,” *Applied and Computational Harmonic Analysis*, vol. 60, pp. 77–122, 2022. [Online]. Available: https://doi.org/10.1016/j.acha.2022.01.004

[63] E. Merkurjev, A. L. Bertozzi, and F. Chung, “A semi-supervised heat kernel pagerank MBO algorithm for data classification,” *Communications in Mathematical Sciences*, vol. 16, no. 5, pp. 1241–1265, 2018.

[64] D. G. Zill, M. R. Cullen, and W. S. Wright, *Differential equations with boundary-value problems.* Brooks/Cole Publishing Company, 1997.

[65] J. Calder, “GraphLearning Python Package,” `doi:10.5281/zenodo.5850940`, 2022.

[66] B. Settles, *Active Learning.* Morgan & Claypool Publishers LLC, Jun. 2012, vol. 6, no. 1. [Online]. Available: https://doi.org/10.2200/s00429ed1v01y201207aim018

[67] A. Kirsch and Y. Gal, “Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities,” *arXiv preprint arXiv:2208.00549*, 2022.

[68] J. Chapman, B. Chen, Z. Tan, J. Calder, K. Miller, and A. L. Bertozzi, “Novel batch active learning approach and its application on the synthetic aperture radar datasets,” in *Algorithms for Synthetic Aperture Radar Imagery XXX*, vol. 12520. SPIE, 2023, pp. 96–111.

[69] C. E. Shannon, “A mathematical theory of communication,” *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[70] D. D. Lewis, “A sequential algorithm for training text classifiers: Corrigendum and additional data,” in *Acm Sigir Forum*, vol. 29, no. 2. ACM New York, NY, USA, 1995, pp. 13–19.

[71] M. Ji and J. Han, “A variance minimization criterion to active learning on graphs,” in *Artificial Intelligence and Statistics*, Mar. 2012, pp. 556–564. [Online]. Available: http://proceedings.mlr.press/v22/ji12.html

[72] Y. Ma, R. Garnett, and J. Schneider, “σ-optimality for active learning on gaussian random fields,” *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[73] K. Miller and A. L. Bertozzi, “Model-change active learning in graph-based semi-supervised learning,” Oct. 2021, arXiv: 2110.07739. [Online]. Available: http://arxiv.org/abs/2110.07739

[74] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, “A tutorial on synthetic aperture radar,” *IEEE Geoscience and remote sensing magazine*, vol. 1, no. 1, pp. 6–43, 2013.

[75] A. I. Flores-Anderson, K. E. Herndon, R. B. Thapa, and E. Cherrington, “The sar handbook: comprehensive methodologies for forest monitoring and biomass estimation,” Tech. Rep., 2019.

[76] 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.

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

[78] L. C. Potter and R. L. Moses, “Attributed scattering centers for SAR ATR,” *IEEE Transactions on image processing*, vol. 6, no. 1, pp. 79–91, 1997.

[79] Q. Zhao and J. C. Principe, “Support vector machines for SAR automatic target recognition,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 2, pp. 643–654, 2001.

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

[81] G. Dong and G. Kuang, “Target recognition in SAR images via classification on riemannian manifolds,” *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 1, pp. 199–203, 2014.

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

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

[84] S. Chen, H. Wang, F. Xu, and Y.-Q. Jin, “Target classification using the deep convolutional networks for SAR images,” *IEEE transactions on geoscience and remote sensing*, vol. 54, no. 8, pp. 4806–4817, 2016.

[85] C. Coman and R. Thaens, “A deep learning SAR target classification experiment on MSTAR dataset,” in *2018 19th International Radar Symposium (IRS)*, 2018, pp. 1–6.

[86] S. Deng, L. Du, C. Li, J. Ding, and H. Liu, “SAR automatic target recognition based on euclidean distance restricted autoencoder,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 7, pp. 3323–3333, 2017.

[87] B. Lewis, T. Scarnati, E. Sudkamp, J. Nehrbass, S. Rosencrantz, and E. Zelnio, “A SAR dataset for ATR development: the synthetic and measured paired labeled experiment (SAMPLE),” in *Algorithms for Synthetic Aperture Radar Imagery XXVI*, E. Zelnio and F. D. Garber, Eds., vol. 10987, International Society for Optics and Photonics. SPIE, 2019, p. 109870H. [Online]. Available: https://doi.org/10.1117/12.2523460

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

[89] N. Inkawhich, M. J. Inkawhich, E. K. Davis, U. K. Majumder, E. Tripp, C. Capraro, and Y. Chen, “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*, vol. 14, pp. 2942–2955, 2021.

[90] M. Swan, A. Major, J. Lear, C. G. Parks, and J. Zhan, “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*, vol. 12095. SPIE, 2022, pp. 115–125.

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

[92] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[93] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, “Supervised contrastive learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[94] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Avances in Neural Information Processing Systems.* MIT Press, 2001, pp. 849–856. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8100

[95] N. Hagen and M. W. Kudenov, “Review of snapshot spectral imaging technologies,” *Optical Engineering*, vol. 52, no. 9, pp. 090 901–090 901, 2013.

[96] F. Lacar, M. Lewis, and I. Grierson, "Use of hyperspectral imagery for mapping grape varieties in the barossa valley, south australia," in *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, vol. 6. IEEE, 2001, pp. 2875–2877.

[97] D. Lorente, N. Aleixos, J. Gómez-Sanchis, S. Cubero, O. L. García-Navarrete, and J. Blasco, "Recent advances and applications of hyperspectral imaging for fruit and vegetable quality assessment," *Food and Bioprocess Technology*, vol. 5, pp. 1121–1142, 2012.

[98] G. Lu and B. Fei, "Medical hyperspectral imaging: a review," *Journal of biomedical optics*, vol. 19, no. 1, pp. 010 901–010 901, 2014.

[99] B. X. Lee, F. Kjaerulf, S. Turner, L. Cohen, P. D. Donnelly, R. Muggah, R. Davis, A. Realini, B. Kieselbach, L. S. MacGregor *et al.*, "Transforming our world: implementing the 2030 agenda through sustainable development goal indicators," *Journal of public health policy*, vol. 37, pp. 13–31, 2016.

[100] T. Arnold, M. De Biasio, R. Kammari, and K. Sayar-Chand, "Development of vis/nir hyperspectral imaging system for industrial sorting applications," in *Algorithms, Technologies, and Applications for Multispectral and Hyperspectral Imaging XXVII*, vol. 11727. SPIE, 2021, pp. 298–304.

[101] G. Bonifazi, L. Fiore, R. Gasbarrone, R. Palmieri, and S. Serranti, "Hyperspectral imaging applied to weee plastic recycling: A methodological approach," *Sustainability*, vol. 15, no. 14, p. 11345, 2023.

[102] A. Sbrana, A. G. de Almeida, A. M. de Oliveira, H. S. Neto, J. P. C. Rimes, and M. C. Belli, "Plastic classification with nir hyperspectral images and deep learning," *IEEE Sensors Letters*, vol. 7, no. 1, pp. 1–4, 2023.

[103] B. Delaporte, T. van Gelder, and K. van der Sluis, "Polymer flake detection through hyperspectral imaging."

[104] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1923–1938, 2018.

[105] M. Dalm, M. Buxton, and F. van Ruitenbeek, "Discriminating ore and waste in a porphyry copper deposit using short-wavelength infrared (swir) hyperspectral imagery," *Minerals Engineering*, vol. 105, pp. 10–18, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0892687516304228

[106] K. Moirogiorgou, F. Raptopoulos, G. Livanos, S. Orfanoudakis, M. Papadogiorgaki, M. Zervakis, and M. Maniadakis, "Intelligent robotic system for urban waste recycling," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2022, pp. 1–6.

[107] A. Morales, P. Horstrand, R. Guerra, R. Leon, S. Ortega, M. Díaz, J. M. Melián, S. López, J. F. López, G. M. Callico *et al.*, "Laboratory hyperspectral image acquisition system setup and validation," *Sensors*, vol. 22, no. 6, p. 2159, 2022.

[108] Å. Rinnan, F. Van Den Berg, and S. B. Engelsen, "Review of the most common pre-processing techniques for near-infrared spectra," *TrAC Trends in Analytical Chemistry*, vol. 28, no. 10, pp. 1201–1222, 2009.

[109] H. Martens, J. P. Nielsen, and S. B. Engelsen, "Light scattering and light absorbance separated by extended multiplicative signal correction. application to near-infrared transmission analysis of powder mixtures," *Analytical Chemistry*, vol. 75, no. 3, pp. 394–404, 02 2003. [Online]. Available: https://doi.org/10.1021/ac020194w

[110] Å. Rinnan, "Pre-processing in vibrational spectroscopy–when, why and how," *Analytical Methods*, vol. 6, no. 18, pp. 7124–7129, 2014.

[111] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 07 1964. [Online]. Available: https://doi.org/10.1021/ac60214a047

[112] A. Biancolillo and F. Marini, "Chemometric methods for spectroscopy-based pharmaceutical analysis," *Frontiers in Chemistry*, vol. 6, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fchem.2018.00576

[113] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.