MACHINE LEARNING PIPELINE FOR RARE-EVENT DETECTION IN SYNTHETIC-APERTURE RADAR AND LIDAR DATA

by

Trey Palmer Scofield

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2021

ACKNOWLEDGEMENTS

I would like to acknowledge my family, friends, Willa, Mary, graduate committee, and Dr. Bradley Whitaker for the motivation and support through my years in graduate school. Earning my masters degree has been a rewarding experience full of knowledge, personal growth, and boxed mac n' cheese. It would not have been able to be completed without the funding and support of the Montana Space Grant Consortium, NASA grant number NNX15AJ19H and the U.S. Air Force Research Laboratory through a subcontract with S2 Corp.

VITA

Trey Palmer Scofield was born in Havre, MT in 1997 to his father Vincent Norman Scofield and his mother Kimberly Compton Scofield. Trey attended Havre High School and graduated in 2015 as a valedictorian. In 2019, Trey graduated cum laude from Carroll College in Helena, MT with a Bachelors of Arts in Applied Mathematics with an Emphasis in Engineering and a minor in Physics. Trey plans to receive his Masters of Science degree from Montana State University in Bozeman, MT in May 2021 and seek a career in the electrical engineering or data science industry.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

Table Page

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

ABSTRACT

In this work, we develop a machine learning pipeline to autonomously classify synthetic aperture radar (SAR) and lidar data in rare-event, remote sensing applications. Here, we are predicting the presence of volcanoes on the surface of Venus, fish in Yellowstone Lake, and select marine-life in the Gulf of Mexico. Given the efficiency of collecting SAR images in space and airborne lidar geographical surveys, the size of the datasets are immense. Immense training data is desirable for machine learning models; however, a large majority of the data we are using do not contain volcanoes or fish, respectively. Thus, the machine learning models must be formulated in such a way to place a high emphasis on the minority, target classes. The developed pipeline includes data preprocessing, unsupervised clustering, feature extraction, and classification. For each collection of data, sub-images are initially fed through the pipeline to capture fine detail characteristics until they are mapped back to their original image to identify overall region behavior and the location of the target class(es). For both sub-images and original images, results were quantified and the most effective algorithm combinations and parameters were assigned. In this analysis, we determined the classification results are not sufficient enough to propel a completely autonomous system, rather, some manual observing of the data will need to be performed. Nonetheless, the pipeline serves as an effective tool to reduce costs associated with electronic storage and transmission of the data, as well as human labor in manually inspecting the data. It does this by removing a majority of the unimportant, non-target data in some cases while successfully retaining a high percentage of the important images.

1

INTRODUCTION

Machine learning is a vital component to problem solving and productivity in several industries today [1]. Scientists are able to analyze data using machine learning algorithms and processing power that we could only dream of in the past. However, an abundance of data is required for effective machine learning and statistical analysis [2]. Therefore, the methods in which data is collected have evolved over time in being able to generate more meaningful data. One area of study that has boosted our ability to collect image data more quickly and efficiently is through optics [3]. Moreover, the precision and power of optics instruments have allowed for a plethora of data collected from aircraft, spacecraft, and other modes of transportation [4].

Synthetic aperture radar (SAR) and lidar are two varieties of optics sensors that are commonly used to collect images of the environment [5; 6]. Each project has unique characteristics in environment, budget, test subject, required accuracy, and other variables that dictate which type of optics sensor is the best option for the application [6]. Therefore, each type of optics sensor has applications where it is superior, proving that choosing the optimum optics sensor is case specific. Nonetheless, research in areas including SAR and lidar have greatly increased the efficiency of collecting data of conventional subjects as well as subjects that we can not physically reach (e.g. space applications), which has begun a large amount of research in the area of remote sensing [7].

Remote sensing refers to detecting, monitoring, or collecting information about an object from a distance without physically touching the object [7]. Furthermore, this detecting, monitoring, and data collection can only be done using technology such as SAR or lidar. Research in remote sensing provides invaluable information in space-imaging of other

planets, stars, and monitoring large-scale events on Earth such as forest fires, tracking cloud movement, changes in farm landscape, and mapping areas of the ocean floor [8].

Once sufficient data is collected, scientists must clean, filter and manipulate the data to perform a thorough analysis [9]. Also, given that SAR and lidar sensors collect an abundance of data, machine learning is often used as a classification method to autonomously identify the target(s) in the collected images [10]. Using tools such as machine learning, or similarly, autonomous learning, produces a more thorough and efficient analysis using less human interaction than traditional data collection and analysis methods [10].

Autonomous learning is known as the physical embodiment of machine learning intelligence [11]. Furthermore, through machine learning, a system is able to examine, understand, and adapt to the target in focus and make decisions accordingly during an autonomous learning step [11]. Not only will further research in autonomous learning result in more algorithms being developed and additional applications using intelligent computing, but it will also increase our understanding of the world at a quicker pace than conventional methods [2].

Oftentimes in real-world applications, the subject of interest is not abundant in the environment [12]. Therefore, we must be patient and wait till the subject emerges and can be studied [12]. Analogously, wildlife photographers do not go outside and immediately photograph the most interesting animals; they must search and be patient to find the rare and breathtaking images they have been searching for. Studying real world phenomena is no different where lots of meaningless data must be collected before finding the important data. In the applications of remote sensing and autonomous learning, this occurrence is called rare-event detection [12].

Rare-event detection refers to identifying target(s) in a collection of data that are sparsely available and identifiable [13]. Furthermore, detecting rare target(s) requires sifting through many non-targets in a high class imbalanced collection of data. When using machine

learning to identify rare-events, precautions must be taken when designing the model not to simply optimize the model's accuracy, but to emphasize the importance of correctly identifying the target class(es) when they are scattered sparsely throughout the data [14].

The research completed in this work involves applying a machine learning pipeline to analyze and autonomously predict the behavior in three datasets with a high class imbalance. Through the funding and support of the Montana Space Grant Consortium and their collaboration with NASA, we identify volcanoes in SAR images on the surface of Venus [15]. In addition, with funding and support from the U.S. Air Force Research Laboratory, we identify invasive lake trout in Yellowstone lake and various types of marine life in the Gulf of Mexico using two different airborne lidar datasets [3; 16].

4

BACKGROUND

The motivation and call for research is to advance the study of machine learning in space-based applications and water-based applications with NASA and the Air Force Research Laboratory, respectively. Both areas of research involve remote sensing and have a highly weighted non-target class which makes the target class unique and rare. The hypothesis for this thesis are as follows:

> Data preprocessing techniques, data clustering, and a variety of feature extraction and classification algorithms were utilized to identify the foremost machine learning pipeline to autonomously classify synthetic aperture radar and lidar images in rare-event, remote sensing applications.

Data Preprocessing Techniques

When obtaining new sets of raw data, it is often important to perform data preprocessing to ensure effectiveness later in the data analysis steps. The goals of this step are to eliminate any uninformative data and transform the data into a more usable form. Examples of data preprocessing measures could include removing invalid numbers from the data, manually creating label files to be used in classification, transforming the data into smaller and more-realistic sub-images for volcano analysis, or performing surface detection and correction in lidar applications. Choosing which data preprocessing methods to apply to a dataset requires a case-by-case analysis as one mold does not fit all datasets.

Data Clustering

Data clustering is the gathering or grouping of data into groups or clusters based on the observations, data items, or feature vectors present in the data [17]. By nature, clustering

is an unsupervised technique, meaning that it does not require labeled data to sort the datapoints. This fact makes clustering attractive for many applications such as pattern analysis, grouping, decision-making, and machine learning situations, including data mining, document retrieval, image segmentation, and pattern classification [17].

Clustering has been studied heavily throughout the late 20th and early 21st centuries but began its work in other disciplines such as biology, psychiatry, psychology, archaeology, geology, geography, and marketing [18]. Today, clustering continues to be used as a functional tool for analyzing data.

Within the discipline of clustering, there are several different approaches that are used to group the data. The first major split in clustering approaches occurs with hierarchical methods and partitional methods [17]. Hierarchical methods produce dendrograms which represent the nested grouping of patterns and similarity levels at which groupings change. Once the dendrogram is created, it can be broken at different levels to obtain different clusterings of the data. On the other hand, partitional approaches obtain a single partition of the data instead of a clustering structure (i.e. dendrogram). A majority of the customary clustering algorithms fall under partitional approaches. See Figure 2.1 for a visual example of the different types of clustering approaches.

Within the two main areas of clustering, there are several different algorithms and optimization methods used to identify the clusters for a given dataset. Although we will not describe these algorithms in detail here, common methods used are least square error, mixture resolving models, k-means models, nearest neighbor models, expectation-maximization (EM) models, and algorithms utilizing graph theory such as single link and complete link models [17]. In this work, the iterative EM algorithm is the chosen optimization method while using the Gaussian Mixture Model (GMM) algorithm.

A GMM is a function that is a superposition of k Gaussian distributions, where k is the number of clusters identified in the data [19]. Within each Gaussian distribution, a mean

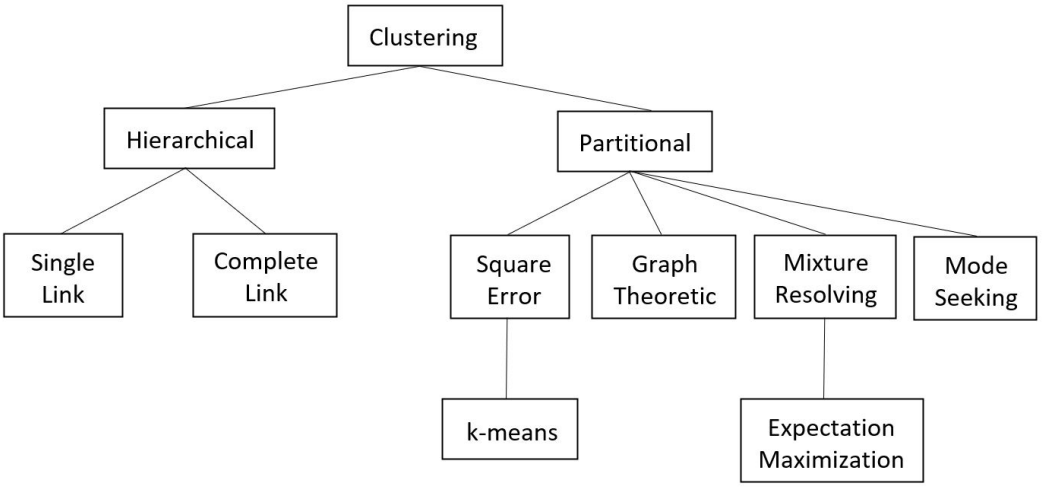


Figure 2.1: Visual display of the different taxonomy of clustering approaches. Image generated after [17].

μ defines the center, a covariance Σ defines the width, and a mixing probability π defines the size of the Gaussian function. Since GMMs are defined over a Gaussian distribution, by nature they are a soft clustering algorithm that assigns cluster probabilities rather than strict assignments. Using the EM algorithm, the parameters $\theta = \{\mu, \pi, \Sigma\}$ can be optimized. The following describes the general steps of the EM algorithm abstractly [20]:

1. Initialize $\theta = \{\mu, \pi, \Sigma\}$ values and label them θ_{old}.

2. Expectation (E) step. Let Z represent latent variables that can take on the values 0 or 1. Let X represent the set of datapoints from our training set. Then, evaluate $p(X, Z|\theta_{old})$ using maximum likelihood estimation.

3. Maximization (M) step. Let

$$Q(\theta|\theta_{old}) = \sum_{Z} p(Z|X, \theta_{old}) \ln p(X, Z|\theta).$$

   Then, find $\theta = \arg\max_{\theta} Q(\theta|\theta_{old})$.

4. Denote $\theta_{old} = \theta$ and repeat steps $2 - 4$ until the log-likelihoods after an iteration are less than some designated threshold value.

Then, once the EM algorithm is complete, datapoints are assigned probability values of being in each Gaussian cluster using the trained θ values, meaning that a GMM model does not need to be a hard-clustering algorithm like a majority of the clustering algorithms [19]. Moreover, each datapoint is assigned probabilities of being in each cluster and not a simple assignment. Datapoints can be assigned a cluster in a GMM by simply choosing the Gaussian with the highest probability value.

Feature Extraction Algorithms

Through extensive research in statistical learning and machine learning, many feature extraction algorithms have been developed and all have separate benefits. For example, some feature extraction algorithms are intended for data dimension reduction while some are intended to boost the dimensions of the data by adding more meaningful information. Nonetheless, feature extraction algorithms are designed to pull, stretch, and manipulate the data in different ways to *extract* meaningful information that will lead to more accurate classification results.

In this work, principal component analysis (PCA) is utilized and applied to the data to determine its effectiveness as a feature extraction algorithm. The goal of PCA is to project the data onto a lower dimensional subspace while minimizing the average error or maximizing the preserved statistical variance in the data. Thus in a machine learning context, it is primarily used as a method of dimensionality reduction [21]. In this work, the components kept following the feature extraction step explain 99% of the variance in the data.

Another feature extraction algorithm that is applied to the data is the fast Fourier transform (FFT) [22]. The Fourier transform of a signal converts it from the temporal or

spatial domain into the spectral or frequency domain. It is well known that spectral features play an important role in image processing [23]. Therefore, our second feature extraction algorithm involved taking the 2-D FFT of each image and using the frequency domain representation of the signal as a datapoint for machine learning classification.

Lastly, two different methods of dictionary learning using sparse coding are applied. In general, dictionary learning aims to generate a sparse representation of the input data in the form of a linear combination of dictionary atoms that form the dictionary itself. The trained dictionary can then be applied to the testing data to create a sparse testing dataset that reflects the features and behavior of the input data. The first method used in this work is a standard dictionary learning approach and the second method is frozen dictionary learning. Sparse coding has a long history of being used in image processing and classification tasks [24; 25; 26; 27]. The goal of sparse coding is to represent a vector in the form

$$x = \mathbf{D}\alpha, \tag{2.1}$$

where $x \in \mathbb{R}^{\mathrm{n}}$ is the original vector, $\mathbf{D}$ is a dictionary matrix whose columns are feature vectors, and $\alpha \in \mathbb{R}^{\mathrm{m}}$ is a sparse coefficient vector, meaning most of the entries of α are zero. The goal of dictionary learning is to use a collection of many input vectors to determine an appropriate dictionary matrix as well as sparse coefficient vectors corresponding to each input. In this work, we use the K-SVD algorithm [28] for feature extraction.

Frozen dictionary learning is a modification of traditional sparse coding dictionary learning that attempts to learn a dictionary that can effectively model imbalanced datasets [29]. The following is an outline for how we use frozen dictionary learning in this work:

1. Separate the sub-image patches into two groups: one known to be target-free (coming from mother images without the target feature), and one known to have some target features (coming from mother images containing target feature).

2. Learn a standard sparse coding dictionary on the group without target features, thus learning a model for 'normal' images.

3. Freeze the dictionary and augment the dictionary with new, randomly-generated columns.

4. Using the group of sub-images containing the target features, use sparse coding to update only the augmented portion of the dictionary. The algorithm is free to use the entire dictionary to model an image patch, but can only update the new, unfrozen dictionary columns. Thus the non-target portions of the dataset will be modeled by the original dictionary, and the augmented columns will update to learn the features associated with the target class.

5. Once the augmented portion dictionary has been learned, use the full dictionary to generate sparse coding vectors for both groups of image patches, as well as new, test data points.

Machine Learning Classification

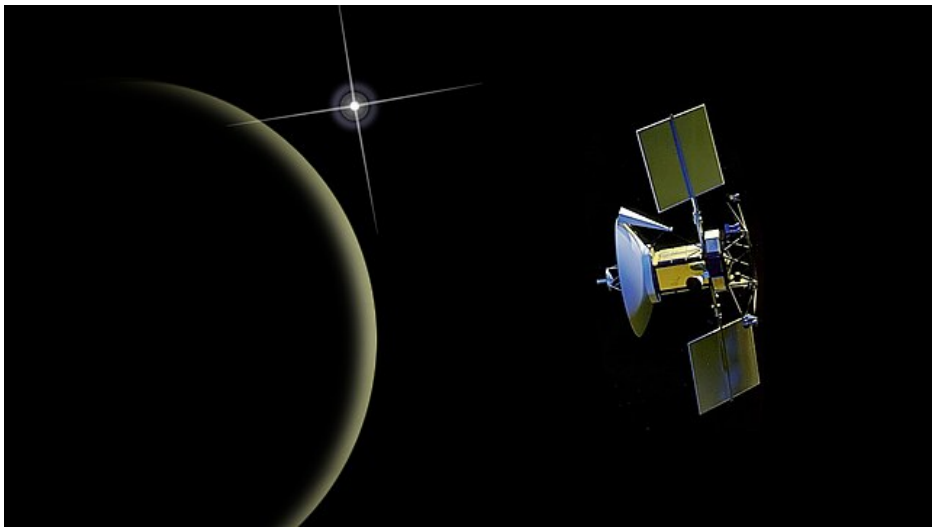
The term *machine learning* was first introduced and discussed in the early 1960s, initially based on a model correlating to human brain cell interaction created by Donald Hebb in 1949 [30]. In the first years of research, nearest neighbor algorithms and the Perceptron algorithm, an early development of neural networks, were the first methods developed. Although there was research support and algorithm framework in place in the mid-to-late 1900s, the field did not start flourishing until the 1990s when computer performance was improved enough to aid in the calculations [31]. Now, machine learning classification is the basis of autonomous learning and artificial intelligence. Machine learning encapsulates the collection of algorithms and methods that make decisions and assign class labels following the

training and testing steps of the pipeline [1]. This pseudo-process is what drives the results of AI models to make real-time decisions and meaningful outcomes in many industries.

There are two main categories of machine learning that dictate the framework of the algorithms: supervised learning and unsupervised learning [1]. Supervised learning algorithms require the presence of class labels in the data. In turn, this step usually requires manual human interaction to label the data prior to any machine learning algorithms being utilized. On the other hand, unsupervised learning algorithms do not require class labels in the data. Hence, the algorithms can predict the class labels on their own through training and testing with only the information available from the data. Therefore, choosing to use supervised or unsupervised learning algorithms often depends on the data in hand and whether labeled data is available or not.

In general, classification algorithms are trained with a sufficiently large dataset containing information the model requires to make accurate class decisions for a particular scenario. Then, once the model is established, new datapoints or an entire testing dataset are given to the classification algorithm to make a prediction and assign class labels to new data [1]. Once class labels are assigned, the effectiveness of the classifier must be quantified with accuracy metrics.

Under the umbrella of supervised and unsupervised learning, there are many different categories of machine learning algorithms. Some examples of algorithm categories include regression algorithms, instance-based algorithms, decision tree algorithms, Bayesian algorithms, and deep learning algorithms [1]. In this work, we chose to implement support vector machines (SVM), an instance-based algorithm; decision tree algorithms; Naive Bayes, a Bayesian algorithm; and neural networks and convolutional neural networks, both deep learning algorithms.



(a) Synthetic aperture radar sensor

(b) Lidar sensor

Figure 2.2: Visual examples of SAR and lidar sensors. (a) the SAR sensor onboard the Magellan spacecraft to collect images of the surface of Venus. (b) an example of a lidar sensor that could be used on the ground (as shown here) or in airborne applications.

Radar and Lidar Imaging

Synthetic aperture radar (SAR) and lidar are two types of optics sensors commonly used to collect images of the environment. Figure 2.2 shows two examples of these sensors. Both types of sensors use electromagnetic radiation to record data of the desired subject, but use different wavelengths which allow them to shoot different subjects more effectively. Furthermore, lidar sensors utilize wavelengths within the visual light spectrum, specifically as lasers, to capture data [6]. On the other hand, SAR use radio waves to record data which are undetectable by the naked eye [32].

In airborne applications, SAR is a type of active data collection where the optics sensor produces its own energy and then records the amount of energy that is reflected back to the aircraft after making contact with the surface of interest [32]. The spatial resolution of recorded radar data is directly related to the ratio of the sensor's wavelength and sensor antenna's effective aperture dimension. Because it is unreasonable to have a very long antenna to achieve high resolution, scientists have created a workaround that defines the "synthetic" in SAR. Furthermore, SAR sensors utilize a sequence of acquisitions from a shorter antenna that is combined to simulate a larger antenna [32].

Lidar, which is an abbreviation for *light detection and ranging*, is another method of data collection that utilizes light in the form of a pulsed laser to measure ranges to the surface of interest [4]. Using the recorded light pulses, scientists are able to generate precise, three-dimensional information about the surface characteristics of the Earth or other surfaces of interest. Topographic and bathymetric lidar systems are two types of sensors that are often used in geographical imaging [4]. Topographic lidar often uses a near-infrared laser that is typically used for geographic data collection and bathymetric lidar uses a green or blue-green laser that is able to penetrate water for seafloor and riverbed applications.

There are several factors that determine which method, SAR or lidar, is preferable for

a specific application. These factors may include cost, sensor accuracy, and environmental conditions. Furthermore, with the costs of aircraft aside, SAR is more cost-effective, it is less sensitive to different environments and conditions, and is more functional for scanning large areas of the environment [5]. On the other hand, lidar sensors are more suited for imaging smaller areas and targets but perform better with fair weather conditions and nighttime use [6].

Remote Sensing

Remote sensing is the studying and gathering of information from a subject from a distance [7]. It is a common term when discussing space exploration and also projects involving SAR or lidar sensors. The advancements in the remote sensing industry have helped scientists further understand the earth and other planets in the galaxy without physically visiting the areas.

Motivation and Preliminary Work

The motivation behind the research in this manuscript is to assist in creating a streamlined procedure for processing remote sensing, SAR or lidar data. The streamlined procedure includes various methods of processing the raw data, removing unimportant images, extracting features, and classifying the images to identify the target and non-target classes.

Given that airborne SAR and lidar sensors are so efficient, an abundance of data is collected in a short period of time. Therefore, an emphasis is going to be placed on removing as many unimportant images as possible to minimize the amount of manual sorting by scientists to reduce human error, time, and payroll costs.

In future work, designing a device onboard the aircraft to filter SAR or lidar images in real-time would be a major benefit for remote sensing applications. A device as described

would help save an abundance of data memory, time, and energy when transmitting data back to earth in spacecraft-based remote sensing projects.

The results of applying the machine learning pipeline to detecting volcanoes from SAR images taken from Venus were published at the i-ETC conference in 2020, "Machine Learning Pipeline for Shift-Invariant Detection of Volcanoes on Venus" [33]. In addition, the work describing the use of the pipeline to identify invasive lake trout in Yellowstone Lake and various marine life in the Gulf of Mexico has been submitted for publication in [34].

MACHINE LEARNING PIPELINE DESCRIPTION

Overview

The primary steps utilized in the pipeline consist of data preprocessing, clustering, feature extraction, and classification. This pipeline displays the procedure of training and testing data points. A visual example of the pipeline can be found in Figure 3.1.

Data Preprocessing

The tasks executed in the data preprocessing step differed depending on which dataset we were focusing on. Furthermore, this step included objectives such as removing NaNs from the data, altering the size of the images themselves, and eliminating any other uninformative data. Further descriptions of the data preprocessing steps that are specific to each dataset will be provided later in Chapters 4 and 5.

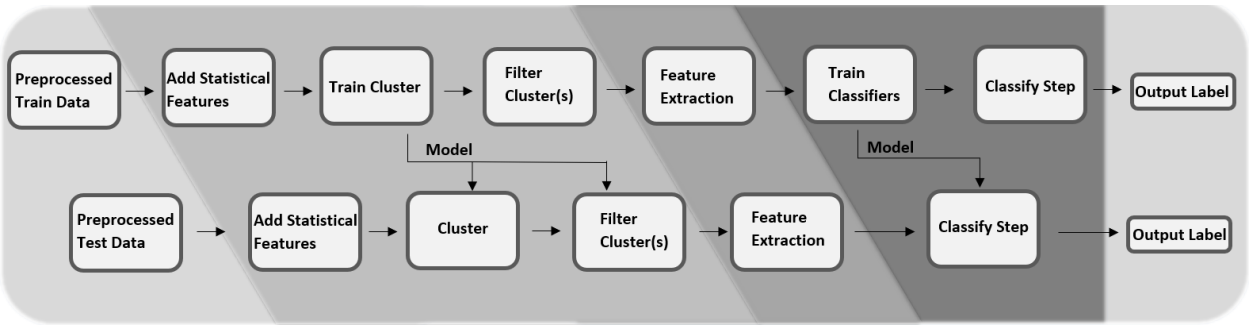


Figure 3.1: Flow chart diagram displaying the training process (top row) and testing process (bottom row) of the pipeline including clustering, feature extraction, classification, and each step's respective outputs.

Clustering

In the clustering step of the pipeline, we train Gaussian Mixture Models using statistical data as the input and filter unwanted clusters from the data. This step is achieved to remove as many non-target class images as possible prior to feature extraction and classification steps.

Statistical Feature Extraction

Statistical measurements were taken from each of the images in both the time domain and the frequency domain following a discrete cosine transform (DCT) computation. The statistical measurements recorded are minimum, maximum, variance, range, standard deviation, mean, median, skewness, and kurtosis. Then, histogram distributions comparing the feature class(es) and non-feature class are examined for each statistical measurement to determine which measurements best distinguish the class differences. These chosen statistical measurements are then set aside while discarding the other measurements.

Discrete Cosine Transform

Discrete cosine transform (DCT), is a common tool used for image compression applications. Furthermore, the DCT helps separate the data into spectral sub-bands of differing importance in respect to the features in the data [35]. Similarly to the discrete Fourier transform (DFT), the DCT transforms the data from the time domain into the frequency domain. However, the DCT uses only real numbers rather than the complex numbers used in a DFT transformation [36].

Gaussian Mixture Model

Gaussian Mixture Models (GMMs) are an unsupervised clustering algorithm that presents k Gaussian distributions (i.e. normal distributions) centered at a mean, μ, in

which datapoints are given a probability of being in cluster k, where k is the number of specified clusters [37]. Assigning datapoints a probability of being in each cluster is called soft-clustering given that a datapoint could be assigned to multiple clusters with the same probability.

We considered using a k-means model which also uses k clusters centered at a μ value, but defines a circular decision boundary to assign datapoints [38]. Ultimately, we ended up using GMMs as our clustering method for a variety of reasons. The difference between a GMM and a common k-means model is that a GMM model also accounts for the variance in the data [39]. In turn, each cluster's boundary shape is defined by the variance. In a k-means model, when classifying a new datapoint to a cluster, we simply use the minimum Euclidean distance to each centroid to assign each cluster label, hence, k-means being a hard-clustering algorithm by nature. However, accounting for the variance in a GMM allows for the boundary shapes to be more flexible and the behavior of the data to be more accurately captured. Although a k-means model is a suitable starting place when applying clustering algorithms to new data, a GMM model was more appropriate for our application. Figure 3.2 shows a side-by-side comparison of k-means and GMM models clustering a set of sample data. It is apparent in this example that the GMM model's flexible decision boundary is useful in capturing the behavior of the data. These same concepts also transpire in other sets of data including the datasets used in this research leading to us choosing the algorithm.

In our GMM model, we assigned each datapoint to one cluster by using the maximum likelihood estimation and coin-flip method if multiple clusters have equal probability [37]. This ensures that cluster boundaries do not overlap and the cluster analysis will be more comprehensible.

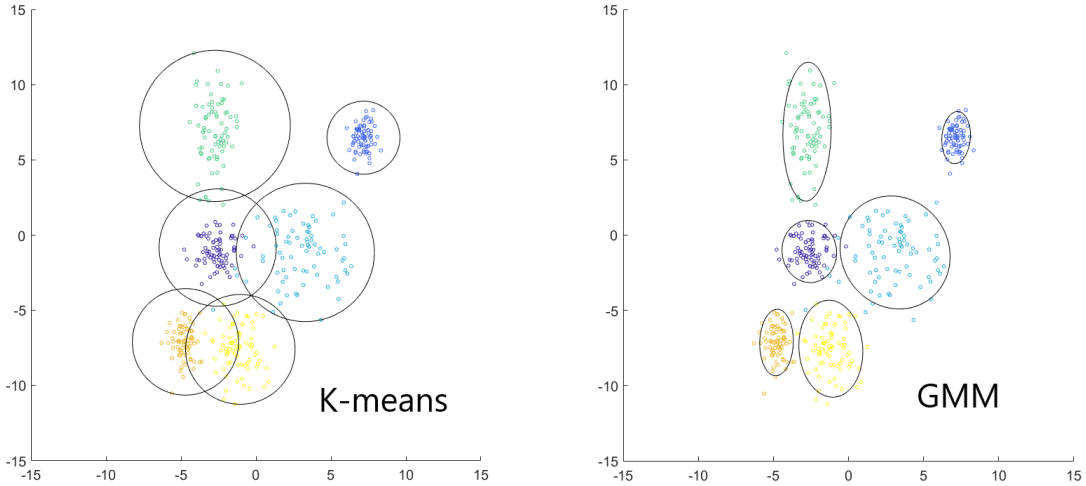


Figure 3.2: Example of k-means and GMM models clustering a set of sample data displaying the flexible decision boundary behavior of a GMM model.

Feature Extraction Algorithms

Several feature extraction algorithms were tried and implemented in the pipeline to improve the classification results. During this step, we searched for algorithms that were able to manipulate the data in ways that could distinguish the classes observed in the data. Furthermore, multiple feature extraction algorithms were tested in various combinations to produce additional sets of features. The algorithms chosen were Principal Component Analysis (PCA), Fourier Transform (using the two-dimensional FFT), Discrete Cosine Transform (DCT), sparse coding dictionary learning, frozen dictionary learning, and finally no feature extraction algorithm at all.

- **PCA**: The goal of PCA is to project the data onto a lower dimensional subspace while minimizing the average error or maximizing the preserved statistical variance in the data. Thus in a machine learning context, it is primarily used as a method of dimensionality reduction [21].

- **FFT**: The Fourier Transform of a signal transforms it from the temporal or spatial domain into a spectral or frequency domain. Here, we take the 2-D FFT of each image and use the frequency-domain representation of the signal as a datapoint for machine learning classification.

- **Dictionary Learning with Sparse Coding**: The goal of dictionary learning is to use a collection of many input vectors to determine an appropriate dictionary matrix as well as sparse coefficient vectors corresponding to each input. In this work, we use the K-SVD algorithm [28] for feature extraction.

- **Frozen Dictionary Learning with Sparse Coding**: Frozen dictionary learning is a modification of traditional sparse coding dictionary learning that attempts to learn a dictionary that can effectively model imbalanced datasets [29]. The data used to train the dictionary is split up into data containing target images and ones that are not. The beginning columns of the dictionary are trained on the images with the non-target class, then are frozen, augmented with new columns, and the new columns are trained using the target data.

Classification Algorithms

The classification algorithms we have chosen to research are support vector machine (SVM), Naïve Bayes Classifier, Decision Trees, feed-forward neural networks, and convolutional neural networks.

Support Vector Machine

The basis behind SVM [40] is to find an N-dimensional hyperplane that classifies the datapoints, where N represents the number of features in the data. Fig. 3.3 illustrates the intuition behind the SVM algorithm. Both figures show data from two classes (represented

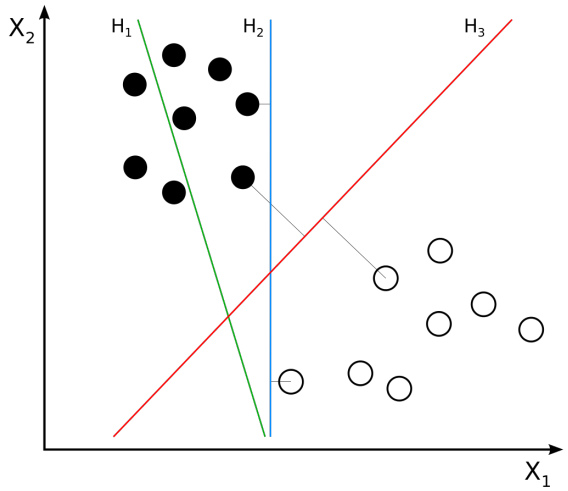


Figure 3.3: Illustration of a 2-dimensional, 2-class support vector machine. This example displays 3 different hyperplanes, H_1, H_2, and H_3. Here, H_1 does not separate the two classes, H_2 does suboptimally, and H_3 does optimally.

as circles and squares). The goal of a support vector machine is to optimally separate the classes, so new data points can accurately be predicted by the separating line or hyperplane. Fig. 3.3(a) shows that many such separating hyperplanes can exist, however, they are not all optimal. The optimal hyperplane for this particular example is shown in Fig 3.3(b). The mathematical principles behind SVMs extend to larger dimensions ($N > 2$) and inseparable datasets. In addition, kernelized SVM models are not restricted to separating data with hyperplanes but allow for a variety of decision boundaries. In this work, non-kernelized, linear SVM models are utilized.

Naïve Bayes Classifier

The Naïve Bayes Classifier is a probabilistic classifier that utilizes Bayes' Rule to calculate the probability that features will be in each class [41]. Bayes' Rule is represented as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{3.1}$$

where A can be replaced with our target and non-target classes and B can be replaced with our set of features. The classifier is coined with the term *naïve* because each feature is assumed to be linearly independent. While this assumption is rarely met, it makes the calculations of the probabilities possible.

Once the probabilities for each data point are calculated, a metric needs to be identified to classify the data point. Here, Naïve Bayes algorithms simply choose the class that has the highest probability. This metric is known as the Maximum A Posteriori decision rule.

The training of a Naïve Bayes Classifier is deterministic and rather quick given that it must only find the individual feature's probabilities and weigh the probabilities. Therefore, even with a high feature count, the training of a Naïve Bayes model is relatively fast.

Decision Trees

Decision tree algorithms are a common method used for many classification applications. In general, the algorithms are optimized quickly, simple to update when more training data becomes available, and generalize well beyond the training set to prevent overfitting when testing on new data [42]. Decision trees were given their name from the unique anatomy of the algorithm structure. The structure begins at the *root* where a decision between two selected predictors is made. Then, the next stage is at the *leaf* of the structure in which contains the binary response. This process is repeated continuously by making a decision for each predictor and then choosing which path or *branch* to follow. Branches can also be referred to as sub-trees [42]. A visual example can be found at Figure 3.4.

There are various tree structures and decision rules that are utilized within the general decision tree anatomy. Typically, decision trees work in a top-down fashion and continue splitting into more branches until splitting no longer adds value to the predictions [42]. Other methods include but are not limited to, bottom-up method, hybrid (top-down and bottom-up blend), and growing-pruning approaches [42]. Additionally, the user can define

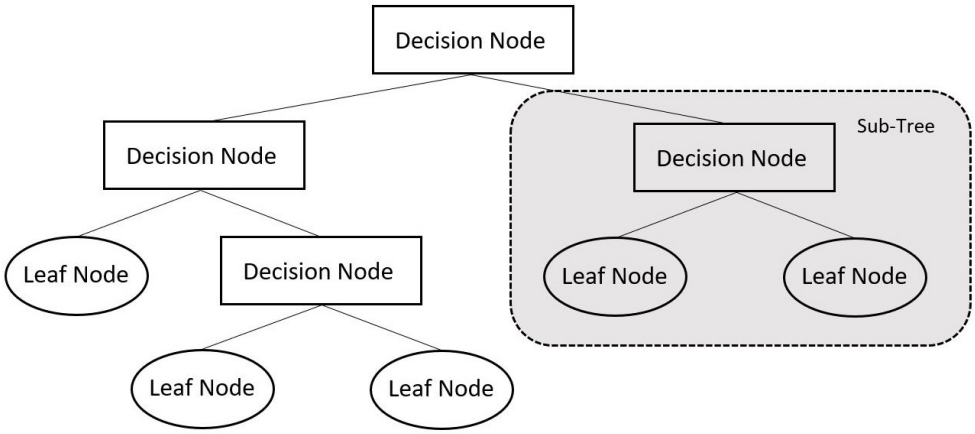


Figure 3.4: Illustration of simple top-down decision tree algorithm displaying the decision nodes, leaf nodes, and sub-trees. Image generated after [43].

the maximum amount of tree splits to improve efficiency, reproducibility, and to have greater control over the model. Oftentimes, tree structures are too large to exhaustively search for the optimum decision rules in the tree, therefore, heuristic search algorithms are utilized to optimize the tree without searching comprehensively [44].

Feed Forward Neural Network

Neural networks have been known to be very accurate in many modern machine learning problems. They are designed to have flexible architectures in that the number of nodes and layers in the hidden layer(s) are chosen by the user. These chosen architectures can have a large impact on the accuracy of the classifier, however, there is no clear-cut method of choosing the architecture. There has been extensive research conducted on the behavior of neural networks, but the current optimization method for choosing a neural network's architecture is simply guess and check.

A visual example of a simple dual-class feed-forward neural network can be found at Fig 3.5. In this example, there is one hidden layer with four nodes and two output nodes. Neural networks have an infinite number of hidden layer architecture possibilities, but the

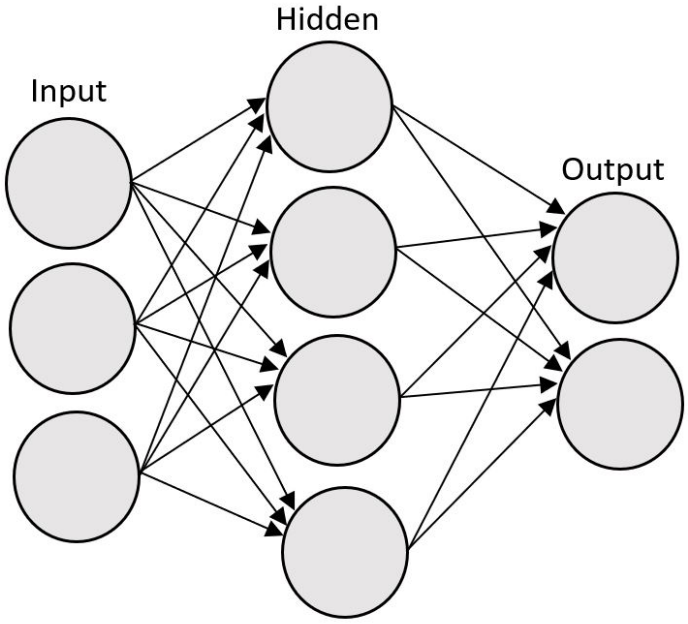


Figure 3.5: A visual example of a simple dual-class feed-forward neural network with one, four node hidden layer. Image generated after [45].

number of input nodes is determined by the dimension of the input data, and the number of output nodes is determined by the number of classes.

Convolutional Neural Network

Similar to a feed-forward neural network, convolutional neural networks (CNN) function in one direction from left to right. However, in a CNN, one or more of the hidden layers also contains a convolutional operator. Within the convolutional layer, the network also contains a kernel and a pooling layer [46; 47; 48]. The kernel is a smaller square matrix than the input image that shadows the original image and records cropped copies. In addition, the kernel also contains padding, meaning that it also searches outside of the image and has properties of edge detection. Once the entire image has been scanned by the kernel, the sub-images are augmented into a new matrix. Depending on the amount of padding added to the kernel, this step often reduces the dimension of the matrix and will increase the performance of the

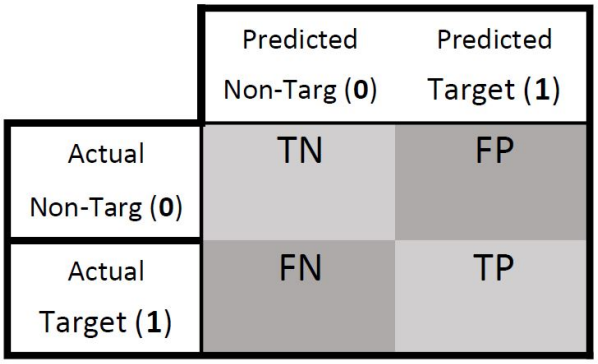


Figure 3.6: A blank dual-class confusion matrix.

network as a result. Also, the pooling layer acts as an additional tool for reducing the spatial size of the matrix by extracting the dominant features from the sub-images. In other words, the pooling layer gathers the pixels with the maximum values from each kernel sub-image and groups them together in a new matrix.

Convolutional neural networks assume two-dimensional input data. Therefore, we can only use CNNs when our preprocessing output yields a matrix (e.g. an image).

Model Accuracy Metrics

In the datasets used in this work, a high class imbalance is present which causes the model's accuracy metric to be unhelpful for quantifying the model's overall effectiveness. Therefore, we must use other metrics that better encapsulate the model's effectiveness of identifying the target class(es). Here, classification performance is measured with recall, precision, and the F_3 score, which is a particular combination of recall and precision. All of these metrics range between 0 and 1, with 1 being the best. These metrics can be understood in terms of a dual class confusion matrix, shown in Figure 3.6.

Recall is a measure of how many images in the target class were correctly predicted

and is defined as

$$\text{recall} = \frac{TP}{TP + FN}. \tag{3.2}$$

A recall of 1 means that all objects of interest were identified.

Precision, on the other hand, is a measure of how many identified targets were actually objects of interest. Precision is defined as

$$\text{precision} = \frac{TP}{TP + FP}. \tag{3.3}$$

A high precision value means the classifier did not mislabel many images containing objects of interest, while a low precision value means that the classifier mislabeled many non-target class items as objects of interest.

The F_β score is the weighted harmonic mean of precision and recall, which weights recall β-times higher than precision. Since recall is more important than precision in our datasets, the F_3 score is used, which weights recall three times higher than precision. The F_β and F_3 scores are defined as

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{3.4}$$

$$F_3 = 10 \cdot \frac{\text{precision} \cdot \text{recall}}{9 \cdot \text{precision} + \text{recall}}. \tag{3.5}$$

The F_3 score provides a single metric that can be used to evaluate classification performance. We use this metric because we consider missing a target to be worse than misidentifying a non-target as a target.

IDENTIFYING VOLCANOES IN SAR IMAGES OF VENUS

Intelligent computing is essential for modern space flight and discovery. For example, researchers are currently investigating the ability of artificial intelligence to manage many space-related tasks, including obstacle avoidance for navigation [49], frequency hopping for communication [50], and prediction and control of life-support systems [51]. However, space-based instruments measure far more data than can feasibly be analyzed by human scientists, or even transmitted to Earth. Thus intelligent software has the potential to increase the science return of space-based missions by locating, recording, and transmitting only the most important data related to the mission. This is especially true when data of interest are extremely rare events.

One major challenge with space-based intelligent computing, however, is that the hardware required for intensive machine learning calculations cannot be sent to outer space. The computational specifications of space-worthy hardware lag behind Earth-based machines by many years. This is due to a variety of reasons, including the radiation environment outside Earth's atmosphere, as well as the size, weight, and power limitations of sending computational equipment to space [52; 53; 54].

In this work, we apply the presented pipeline to a dataset provided by NASA. We strive to address two important challenges of space-based intelligent computing: limited transmission capabilities (with respect to sensing capabilities) and limited on-board computational resources. The most effective classifier that meets the on-board computational constraints can then be selected and optimized for space-based implementation. We demonstrate a proof-of-concept of the process by attempting to detect volcanoes in a dataset of synthetic aperture radar (SAR) images of the surface of the planet Venus. This process can be adapted to other mission-based sensing problems, allowing for on-board processing to effectively prioritize the transmission of data collected in space.

Volcanoes on Venus Dataset

The featured dataset is an open-source dataset of SAR images taken of Venus' surface in the 1990s by NASA's Magellan spacecraft. The "Volcanoes on Venus" dataset will provide a robust proof of concept that will test the effectiveness of intelligent algorithms in identifying meaningful features in images [15]. The algorithms can then be applied to new datasets and even real-time situations to identify important features. The original dataset contains $7,000$ training images and $2,734$ testing images, all of which are 110×110 pixels. Each pixel is represented as a value between 0 and 255 and represents an elevation value on the image surface. Along with each of the image sets, there are two additional sets that contain labels for the images in the training and testing sets. Each image contains a binary value whether there is a volcano or not in that image, as well as a confidence level between 1 and 4. See Fig 4.1 for an example of arbitrary images from the original training set, where the first-row images have no volcanoes, and the second-row images all have volcanoes.

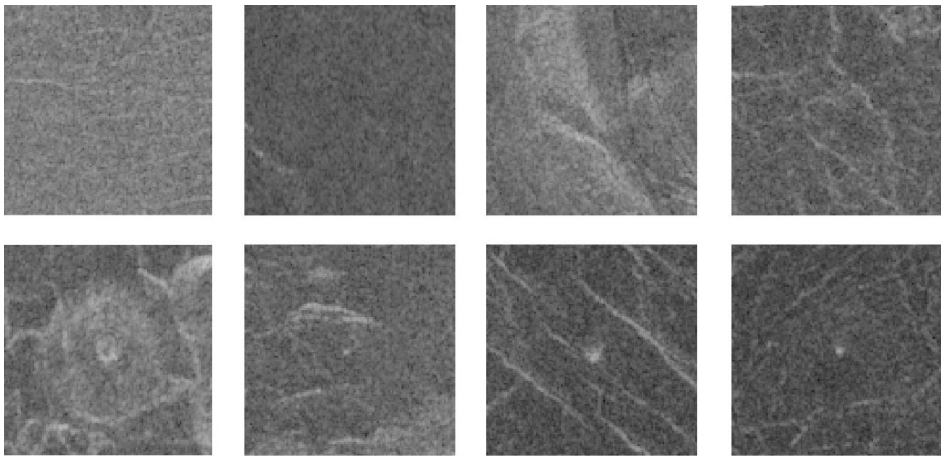


Figure 4.1: Examples of eight arbitrary images from the "Volcanoes on Venus" dataset [15]. Four images without volcanoes are in the top row, and four images with volcanoes are in the bottom row. Note that in the original dataset all volcanoes are centered in the example images.

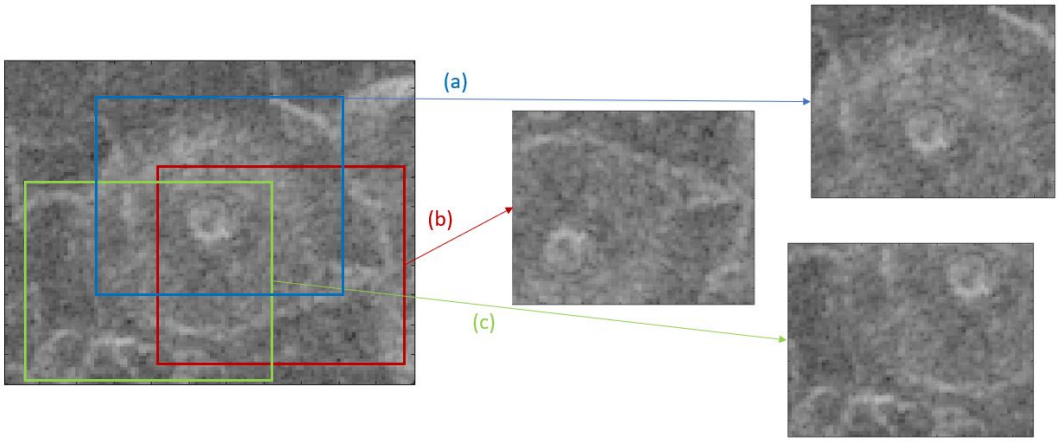


Figure 4.2: Visual description of generating 65×65 pixel dataset from 110×110 pixel dataset. This example shows three 65×65 pixel images being created where (a) and (c) have no transformation and (b) has a flip around the horizontal axis.

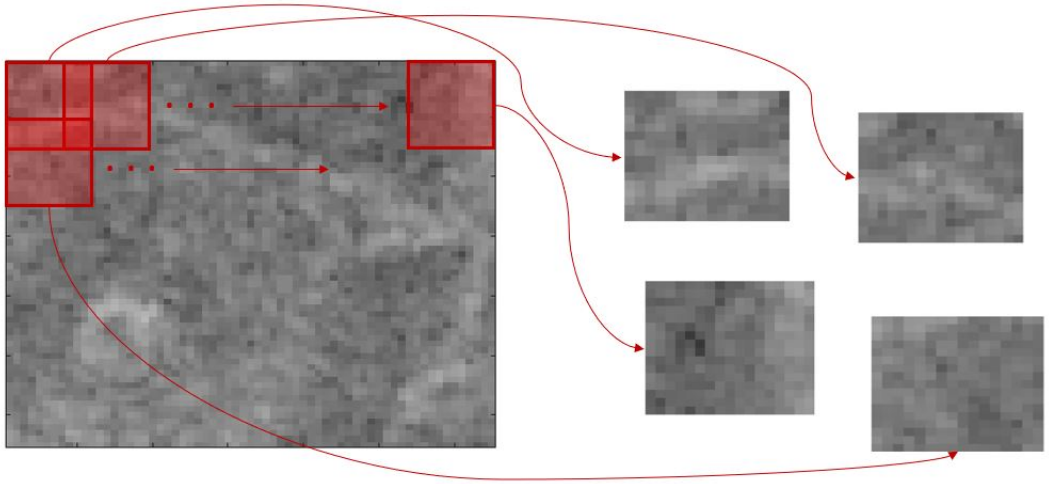


Figure 4.3: Visual description of generating 16×16 (or similarly, 8×8) pixel dataset using the sliding window function from the 65×65 pixel data using an overlap of 4 pixels.

To transform the data into a more realistic form, we created additional datasets consisting of smaller images. The first dataset was created to overcome the impracticability of the volcanoes being located in the middle of the 110×110 pixel images. The transformation for this dataset involved randomly cropping each image from 110×110 to 65×65 pixels and executing an image transformation by either transposing, flipping around the horizontal axis,

both, or none. We repeated the procedure five times for each image, increasing the size of the training dataset from $7,000$ to $35,000$ images and testing dataset from $2,734$ to $13,670$ images. In turn, this data preprocessing creates a more realistic scenario by increasing the dataset size and forcing the algorithms to search in all areas of the images for volcanoes.

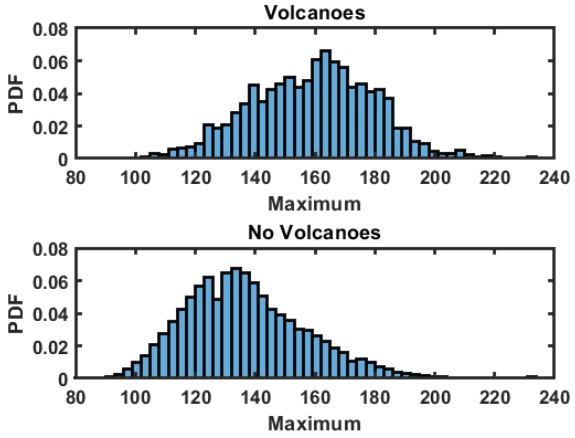
In a real-world context, we may be interested in determining the location and not just the presence of a volcano in an image. With this in mind, we created two additional datasets of smaller image patches. In this data preprocessing step, we implemented a 16×16 pixel sliding window with an overlap of 4 pixels to transform each 65×65 image into a collection of 25 16×16 image patches and an 8×8 pixel sliding window with an overlap of 2 pixels to transform each 65×65 pixel image into a collection of 100 8×8 image patches. Figure 4.3 shows an example of how the sliding window functions visually. Generating these datasets greatly increases the number of training samples (from $35,000$ to $875,000$ and $3,500,000$, respectively) and testing samples (from $13,670$ to $341,750$ and $1,367,000$, respectively), but adds additional complexity by dramatically increasing the class imbalance. Moreover, the percentage of volcano images in the datasets decrease from the 65×65 to 16×16 pixel training datasets from 14.29% to 0.79% and from 14.29% to 0.14% from the 65×65 to 8×8 pixel training datasets. Therefore, while the two new datasets generated by the sliding box function are more difficult to analyze, they represent a more realistic setting for space-based classification algorithms. The analysis of these datasets allows this work to be straightforwardly applied to other remote-sensing applications due to its complexity and large class-imbalance.

Clustering

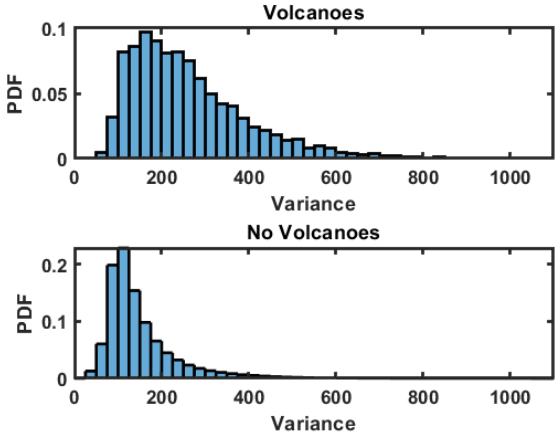
Statistical data was recorded for each image in the training and testing sets in both the time and frequency domain. Histograms of all eighteen statistical measurements of the 16×16 pixel training set were examined to determine the most effective measurements

k	Number of Filtered Clusters	Non-Volcano Images Removed	Volcano Images Removed	Percent of Target Class Removed	Percent of Total Data Removed
2	1	722855	2928	42.53%	82.95%
3	1	492552	1046	15.19%	56.53%
4	1	254475	304	4.42%	29.12%
5	1	163707	150	2.18%	18.73%
6	2	325063	498	7.23%	37.21%
7	4	3832	71	1.03%	0.44%
8	4	3337	33	0.48%	0.38%
9	5	19870	132	1.92%	2.29%
10	5	16776	157	2.28%	1.94%
11	6	19266	145	2.10%	2.22%
12	6	19243	139	2.02%	2.22%
13	7	19803	136	1.98%	2.28%
14	8	20075	150	2.18%	2.31%
15	8	42473	134	1.95%	4.87%
Corresponding Test Cluster Summary Using Selected k Value					
3	1	191865	990	14.38%	56.14%
5	1	64679	106	3.45%	18.96%

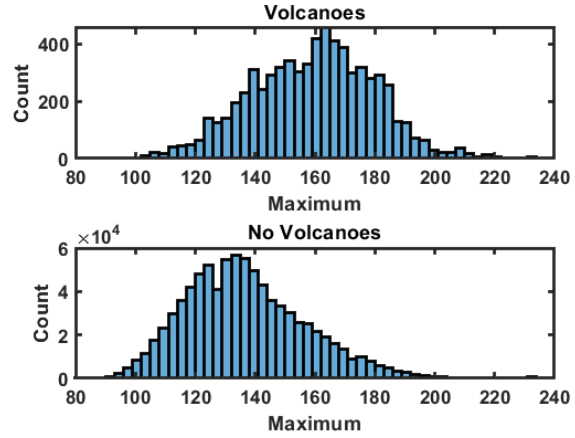
Table 4.1: Clustering Summary for 16×16 Pixel Data



(a) Maximum elevation distribution



(b) Elevation variance distribution



(c) Non-normalized maximum elevation distribution

Figure 4.4: Histograms of the maximum and variance distribution in the time domain of the 16×16 pixel dataset comparing the volcano and non-volcano classes.

to separate the volcano class and non-volcano class. The chosen statistical measurements are maximum, variance, range, standard deviation, and skewness in the time domain and minimum and kurtosis in the frequency domain. Given that the images are generated from SAR, the statistical measurements are based on the elevations present in each respective sub-image. See Figure 4.4 for example distributions of the chosen statistical measurements from the 16×16 pixel dataset and Appendix A for the entire collection of histograms. Figures 4.4a

and 4.4b show the maximum elevation and elevation variance distributions comparing the volcano and non-volcano classes. From these distributions, we can predict that if an image has a large maximum elevation and large elevation variance, it is more likely to contain a volcano. Identifying measurements that separate the two classes distinctly is important not only in the clustering step but also the classification step to intelligently predict whether an image contains a volcano or not. It is important to note that the histogram distributions are normalized, meaning that the volcano histograms are in fact shorter in height than the non-volcano class given the class imbalance in the data. Figure 4.4c show the non-normalized maximum distribution to display the magnitude of difference in histogram height.

A parameter search for the optimum number of GMM clusters, k, was completed. In this search, we altered the value of k from 2 to 15. A full clustering summary for the 16×16 and 8×8 pixel datasets can be found at Tables 4.1 and 4.2.

In Table 4.1, there are two different values of k highlighted in the clustering summary for the 16×16 pixel dataset. Shown in gray are the training and testing clustering summaries for $k = 3$ and shown in blue are the training and testing clustering summaries for $k = 5$. The cells highlighted in blue are notable because they represent the value of k that filters out a suitable number of non-volcano images while removing only a small number of volcano images. Here, a total of 18.73% and 18.96% of the total data is removed while removing only 2.18% and 3.45% of the images containing volcanoes from the training and testing datasets, respectively. The cells highlighted in gray are also important because they exhibit the optimum k-value that can be trained unsupervised. Since only one cluster is filtered from the GMM model in this case, we can simply remove the cluster with a majority of the images rather than examining the labeled data to obtain the results shown in the table. Furthermore, this method removes 56.53% and 56.14% of the total data while removing 15.19% and 14.38% of the images containing volcanoes from the training and testing datasets, respectively. This method yields a less than optimal result for rare-event detection applications because it

k	Number of Filtered Clusters	Non-Volcano Images Removed	Volcano Images Removed	Percent of Target Class Removed	Percent of Total Data Removed
2	1	2858351	1477	29.54%	81.67%
3	1	1735689	549	10.98%	49.61%
4	1	1024509	267	5.34%	29.28%
5	1	666304	169	3.38%	19.04%
6	2	1234132	324	6.48%	35.27%
7	4	207950	213	4.26%	5.95%
8	3	1510615	405	8.10%	43.17%
9	5	98619	165	3.30%	2.82%
10	3	1235272	315	6.30%	35.30%
11	7	386885	351	7.02%	11.06%
12	6	84734	89	1.78%	2.42%
13	5	98968	93	1.86%	2.83%
14	6	1498479	455	9.10%	42.83%
15	7	103588	145	2.90%	2.96%
Corresponding Test Cluster Summary Using Selected k Value					
3	1	680440	581	26.77%	49.78%
5	1	264582	190	8.76%	19.35%

Table 4.2: Clustering Summary for 8×8 Pixel Data

removes a large amount of the volcano images. However, if losing a large amount of the target class was suitable for the problem, choosing this GMM model would filter out a majority of the non-target data. In our circumstances, we cannot afford to lose that much of our target-class data with the large class imbalance present. Therefore, we will move forward in this work with the GMM model where $k = 5$.

The clustering summary for the 8×8 pixel data can be found in Table 4.2. Here, there are two training k values and their corresponding test values highlighted. The cells highlighted in gray show the training and testing clustering summaries where $k = 3$, which represent the best k-value for clustering the data unsupervised. This method removes a large amount of the training and testing data, 49.61% and 49.78%, respectively. However, this method also removes a portion of the volcano data from the training and testing sets, 10.98% and 26.77%, respectively. Therefore, clustering with three clusters is effective at removing a large amount of non-volcano images, but should only be used if losing a large amount of the target class is sufficient. The cells highlighted in blue show the clustering summary for the training and testing sets for $k = 5$. These cells represent the best overall ratio of removing the most non-volcano images while keeping the lost target class images to a minimum. Furthermore, this method removes 19.04% and 19.35% of the total training and testing datasets, while only losing 3.38% and 8.76% of the volcano images, respectively. Given that retaining volcano images is vital to this remote sensing project, we will move forward with the filtered data clustered into 5 clusters.

Classification Results

Following the clustering and filtering steps, the datasets were moved to feature extraction and classification. Here, classification metrics for both datasets were calculated for every combination of feature extraction method and classification algorithm. Then, data storage conclusions can be generated by analyzing the percentage of data removed and the

percentage of volcano data lost if all data predicted 0 were thrown out.

Full classification results for the 16×16 pixel dataset can be found in Table 4.3. In this figure, the best algorithm combination for each machine learning classifier is highlighted. However, the cells highlighted in green represent the best overall algorithm combination for the 16×16 pixel dataset. This algorithm combination represents the results generated by extracting via sparse coding and classifying the data from a decision tree. The confusion matrices for this combination can be found at Figure 4.5. For the 16×16 sub-images, the algorithms were able to attain a recall of 0.7107 and a precision of 0.0318. From a data storage perspective, this is equivalent to removing 79.93% of the total data and retaining 71.07% of the volcano images. When the predicted sub-image labels are mapped back to the 65×65 pixel images, the algorithms generate a recall of 0.9069 and a precision of 0.2034. In other words, from these predictions we are able to remove 29.22% of the total data while retaining 90.69% of the volcano images.

The 8×8 pixel dataset classification results can be found in Table 4.4. The best results for each classification algorithm are highlighted in yellow and in green for the best algorithm combination in the dataset. Here, the best overall option for the 8×8 pixel dataset is with sparse coding and support vector machines. This combination achieved a sub-image recall of 0.5691 and precision of 0.0125 and a recall of 0.8060 and precision of 0.2015 for the 65×65 pixel images. The confusion matrices for this algorithm combination can be referenced at Figure 4.6. Again, analyzing these results from a storage perspective allows us to draw conclusions on the data efficiencies gained from the pipeline. For the 8×8 pixel sub-images, we are able to throw away 92.45% of the total data and retain 56.91% of the volcano data. Likewise, we can remove 36.51% of the total data and retain 80.60% of the volcano images for the corresponding 65×65 pixel data.

	Feature Extraction Method	Naive Bayes			
		Accuracy	Precision	Recall	F_3
16×16	Sparse Coding	0.8668	0.0385	0.4772	0.2230
	Frozen Dictionary	0.9052	0.0412	0.3519	0.2006
	Nothing	0.4167	0.0131	0.7189	0.1125
65×65	Sparse Coding	0.5015	0.2062	0.7507	0.5939
	Frozen Dictionary	0.4900	0.1892	0.6737	0.5364
	Nothing	0.3519	0.1756	0.8341	0.6066

	Feature Extraction Method	Decision Tree			
		Accuracy	Precision	Recall	F_3
16×16	Sparse Coding	0.8031	0.0318	0.7107	0.2267
	Frozen Dictionary	0.8483	0.0310	0.6570	0.2176
	Nothing	0.7539	0.0310	0.7273	0.2241
65×65	Sparse Coding	0.4214	0.2034	0.9069	0.6738
	Frozen Dictionary	0.4832	0.2127	0.8350	0.6460
	Nothing	0.4161	0.2007	0.8982	0.6666

	Feature Extraction Method	SVM			
		Accuracy	Precision	Recall	F_3
16×16	Sparse Coding	0.7005	0.0238	0.8083	0.1881
	Frozen Dictionary	0.8580	0.0418	0.6746	0.2684
	Nothing	0.6965	0.0233	0.8008	0.1846
65×65	Sparse Coding	0.4232	0.1883	0.7954	0.6015
	Frozen Dictionary	0.5098	0.2160	0.7940	0.6264
	Nothing	0.4119	0.1850	0.7945	0.5976

Feature Extraction Method	CNN			
	Accuracy	Precision	Recall	F_3
16×16 - Nothing	0.5576	0.0192	0.8052	0.1581
65×65 - Nothing	0.3404	0.1782	0.8733	0.6282

Table 4.3: Classification results for each feature extraction method and classification algorithm for the 16×16 pixel volcano data.

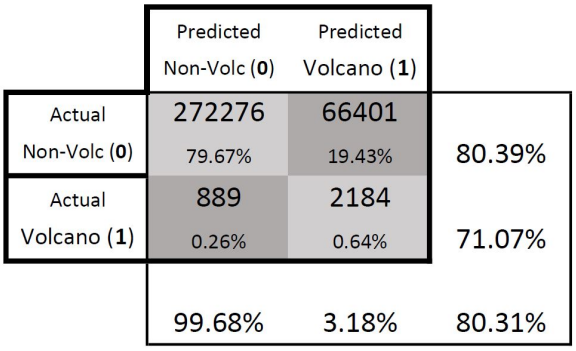
	Feature Extraction Method	Naive Bayes			
		Accuracy	Precision	Recall	F_3
8×8	Sparse Coding	0.9369	0.0125	0.4751	0.1011
	Frozen Dictionary	0.9514	0.0088	0.2539	0.0671
	Nothing	0.7679	0.0045	0.6371	0.0423
65×65	Sparse Coding	0.4467	0.1870	0.7424	0.5724
	Frozen Dictionary	0.3656	0.1654	0.7406	0.5495
	Nothing	0.4261	0.1912	0.8097	0.6118

	Feature Extraction Method	Decision Tree			
		Accuracy	Precision	Recall	F_3
8×8	Sparse Coding	0.9781	0.0231	0.2962	0.1357
	Frozen Dictionary	0.9719	0.0201	0.3355	0.1306
	Nothing	0.9721	0.0197	0.3236	0.1273
65×65	Sparse Coding	0.6570	0.2306	0.4968	0.4454
	Frozen Dictionary	0.6202	0.2165	0.5318	0.4642
	Nothing	0.6101	0.2113	0.5327	0.4624

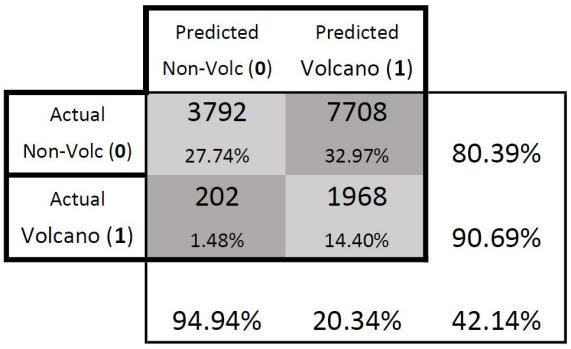
	Feature Extraction Method	SVM			
		Accuracy	Precision	Recall	F_3
8×8	Sparse Coding	0.9247	0.0125	0.5691	0.1044
	Frozen Dictionary	0.9870	0.0246	0.1762	0.1090
	Nothing	0.9685	0.0215	0.4049	0.1455
65×65	Sparse Coding	0.4623	0.2015	0.8060	0.6200
	Frozen Dictionary	0.7340	0.2176	0.2604	0.2554
	Nothing	0.6275	0.2321	0.5834	0.5067

Feature Extraction Method	CNN			
	Accuracy	Precision	Recall	F_3
8×8 - Nothing	0.9609	0.0191	0.4481	0.1380
65×65 - Nothing	0.5530	0.2099	0.6567	0.5414

Table 4.4: Classification results for each feature extraction method and classification algorithm for the 8×8 pixel volcano data.

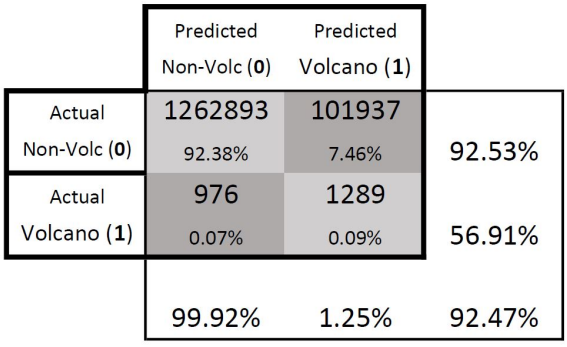


(a) 16×16 pixel dataset

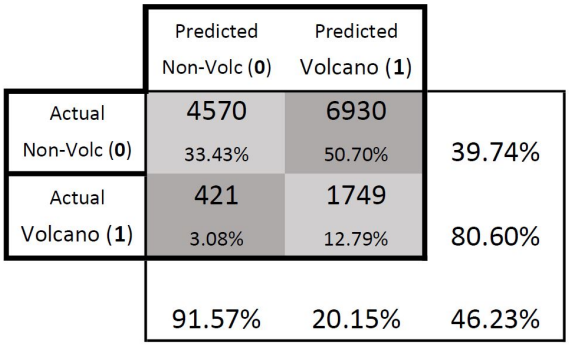


(b) 65×65 pixel dataset

Figure 4.5: Confusion matrix of the 16×16 and corresponding 65×65 pixel volcano datasets extracted via sparse coding and labeled by a decision tree classifier. Results of this algorithm combination can be found in Figure 4.3.



(a) 8×8 pixel dataset



(b) 65×65 pixel dataset

Figure 4.6: Confusion matrix of the 8×8 and corresponding 65×65 pixel volcano datasets extracted via sparse coding and labeled by a SVM classifier. Results of this algorithm combination can be found in Figure 4.4.

Discussion and Conclusion

From analyzing the results in the previous section, it is determined that the classification results are adequate enough to aid in storage efficiency; however, not sufficient to autonomously label the data without some manual sorting. Nonetheless, the number of images that the pipeline is able to effectively remove allows for countless hours of scientists manually sorting data to be spent elsewhere. For example, if the pipeline for the 16×16 pixel dataset was used, the algorithms would be able to filter $\approx 30\%$ of the data while keeping a large majority of the target images ($\approx 90\%$). Suppose each image requires 30 seconds for scientists to examine on average. If there are a $500,000$ images to examine (not out of reach for remote-sensing data collection), this pipeline would save scientists $75,000$ minutes of work or $1,250$ hours. Therefore, this pipeline provides real value to scientists in saving storage space, time, and confidence in the data by keeping a majority of the important data.

For the two sub-image datasets generated in this project, 16×16 pixel and 8×8 pixel datasets, the goal was to perform analysis on the sub-regions of the image and then map the results back to the 65×65 pixel images to identify if/where the volcanoes are located. We saw the classification results for the 65×65 pixel data after mapping back from the sub-image; however, we have not examined visual examples of how this looks. See Figure 4.7 for six examples of the 16×16 pixel sub-images mapped back to their corresponding 65×65 pixel mother images. Here, the red "$\times$" represents the true volcano location and the green "$\star$" represent the center pixels of sub-images that predict there is a volcano. These examples display a variety of scenarios of this process. Figure 4.7a shows a true positive example, but one that did not predict the volcano location very precisely. In this example, the prediction of several sub-images is enough to identify this image as being important, however, the location is inaccurate and will need further manual examination. Figures 4.7b and 4.7c show additional true positive examples that do an acceptable job at predicting the

correct location of the true volcano. However, there are still sub-images in these images that do not predict the correct region of the image. Figure 4.7d shows a true positive example that did not predict the correct region of the 65×65 pixel image. Again, it is still considered a true positive but will require a full manual examination to determine where the volcano is located from our prediction. Figure 4.7e shows a false negative example where there is a volcano in the mother image, but our model predicted no volcano present. Moreover, false negatives are very undesirable in data analysis such as this one with a high class imbalance. Luckily, the high recall rates display that the false negative instances are rare. Lastly, Figure 4.7f shows an example of a true positive that does a very good job of identifying the volcano and locating it correctly in the 65×65 pixel image. Ideally, all 65×65 pixel images with volcanoes would look like this example where a handful of sub-images clustered around the true volcano location predict there is a volcano. One thing to consider, the red "$\times$" identifies the center pixel of the volcano; however, many of the volcanoes are big and take up a large portion of the image. Therefore, the predictions not immediately next to the red marks in many cases are still identifying the correct location of a volcano.

As a sanity check for our pipeline, we tested the algorithm's effectiveness on perfect volcano images. Or in other words, we tested the algorithms on images that contained more meaningful and obvious information about where the volcano location is. To do this, we replaced the center pixel of the volcano locations with a black dot and re-ran the pipeline. Through this process, we achieved an accuracy of $\approx 100\%$. This fact is significant because it shows that the algorithms we used in the pipeline do indeed work; however, the data resolution is poor which makes the problem a difficult one. Therefore, perhaps with higher quality images of the same subject, the pipeline could achieve higher accuracy rates.

In conclusion, we recommend using a larger sub-image with a bigger overlap (i.e. 16×16 pixels with a 4 pixel overlap) to analyze and identify objects of interest in SAR, remote-sensing applications. In addition, choosing a number of clusters of approximately five will

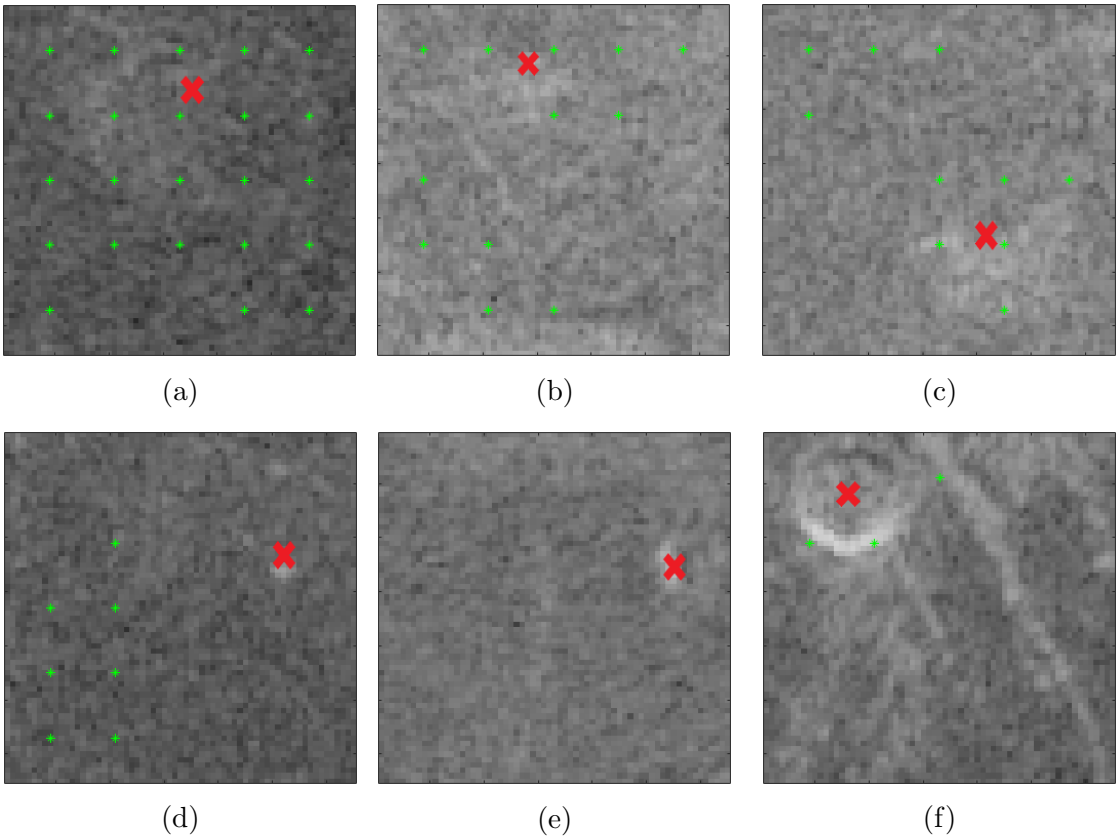


Figure 4.7: Examples of the 16×16 pixel dataset mapped back to its corresponding 65×65 pixel mother-image. The red "$\times$" represents the true volcano location and the green "$\star$" represent the center pixels of sub-images that predict a volcano is in the sub-image.

ensure that a portion of the non-target data is removed without removing many important datapoints. However, if the target class is abundant in the data and it is acceptable to throw away a portion of the target datapoints, consider clustering unsupervised with two clusters and throwing away more of the data. Using a method like this will lower the overall model recall, but may improve the overall classification results. Also, if losing target data is acceptable, a smaller sub-image size with a smaller overlap (i.e. 8×8 pixels with a 2 pixel overlap) will allow for more data to be thrown away. These observations are model specific and must be considered based on the dataset and parameters the project is constrained to.

IDENTIFYING FISH IN AIRBORNE LIDAR DATA

Lidar technologies are constantly being refined in the field of optics given their benefits in many industries. For example, airborne lidar is a common contemporary method for performing ecological surveys of different environments. Typical surveys may include mapping rugged mountainous landscapes, surveying mining areas, or inspecting power lines. In this application, airborne lidar is utilized to explore fisheries in oceans and lakes.

Due to the efficiency and ease in collecting airborne lidar data, large amounts of data are collected from each day of using the sensor. Therefore, traditional methods of data processing and labeling are laborious to complete. Most fish lidar data do not contain physics-based features that can be used to identify objects of interest. Consequently, machine learning and other automated methods prove to be valuable tools when analyzing airborne lidar data. Utilizing computers to help with lidar applications allows for less manual examination by scientists which allows for less preventative error and more lidar surveys to be completed in the end.

Lidar Datasets

In this work, we studied and tested the machine learning pipeline outlined in Chapter 3 on two separate lidar datasets: Yellowstone Lake and Gulf of Mexico. The goal of the Yellowstone Lake dataset is to identify invasive lake trout in Yellowstone Lake, Wyoming and to identify flying fish, schools of unidentified fish, jellyfish, and plankton in the Gulf of Mexico dataset.

Yellowstone Lake

The Yellowstone Lake survey was conducted to identify spawning locations of invasive lake trout (Salvelinus namaycush) in Yellowstone Lake, Yellowstone National Park [55]. The

survey was conducted with daytime flights during lake trout spawning season in 2015 and 2016 using airborne lidar. The data was examined manually by Roddewig et al. [16] and label files were created which we used as ground truth labels in our pipeline. The 2016 flight data was used in this work as the training data leaving the 2015 flight data for testing. A summary of the Yellowstone Lake datasets is shown in Table 5.1, and a representative example of lidar data is shown in Figure 5.1.

	Instances	**Dimensions**	**Fish Instances**
2015 Flight	108778	2048	364 (0.335%)
2016 Flight	192201	2048	37 (0.019%)

Table 5.1: Summary of data collected at Yellowstone Lake [16], showing the number of instances (lidar shots), dimensions (samples per shot), and the number of instances containing fish. The percentage of fish instances is shown in parentheses. A single fish often shows up in multiple adjacent instances.

Gulf of Mexico

The Gulf of Mexico survey was conducted by Churnside et al. over 12 days from September 20 to October 6, 2011 [3]. The data contains instances of flying fish, schools of fish, jellyfish, and plankton layers. All classes were examined and labeled manually by scientists. The data from October 1, 2011, was used as the training data and all the other days were used for testing. A summary of the dataset is shown in Table 5.2, and a representative example of lidar data is shown in Figure 5.2.

The original data was in the form of 1000×1000 pixel PNG images, with multiple images for each day; for each PNG image, object labels were denoted by start and end columns in the image. The original data were transformed such that each day became a separate MATLAB

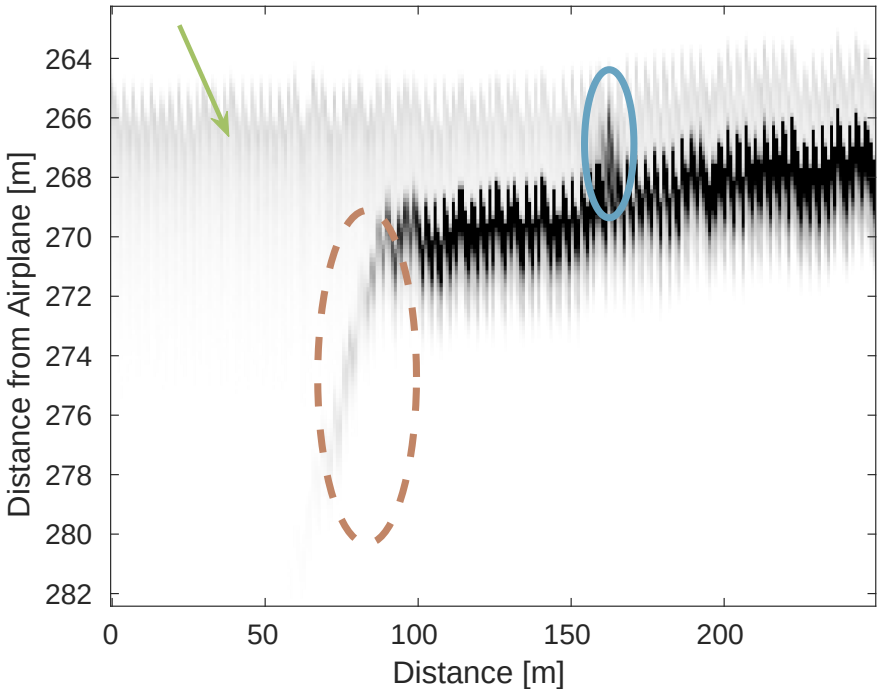


Figure 5.1: Example cross-polarized lidar data from Yellowstone Lake, shown as a grayscale image. Darker values indicate higher reflected radiance; radiance values were compressed to increase contrast. The light gray patch indicated by the green arrow is the surface of the water. The start of an underwater shelf is indicated by the brown dashed ellipse. The blue ellipse highlights a typical fish hit, which appears as a vertical spike above the shelf.

mat file. The labels were placed in a matrix, where each row corresponded to one of the four different objects of interest, and each column indicated whether an object was present.

Lidar Instrument Description

Both datasets were collected with similar dual-polarization lidar instruments using Q-switched Nd:Yag lasers operating at a wavelength of 532 nm. The Yellowstone Lake data were measured with a lidar system developed at Montana State University, employing a diode-pumped laser transmitter putting out 100 pulses/s at 26 mJ per pulse and 7.2 ns pulse width with separate receiver telescopes for co-polarized and cross-polarized signals digitized at 800 Msamples/s at 14-bit resolution [55]. The Gulf of Mexico data were measured with the

	Instances	**Dimensions**	**Fish Instances**
Day 1	600731	1000	113777 (18.94%)
Day 2	220434	1000	45311 (20.56%)
Day 3	418544	1000	75513 (18.04%)
Day 4	318984	1000	42306 (13.26%)
Day 5	116101	1000	9188 (7.91%)
Day 6	132867	1000	8850 (6.66%)
Day 7	183236	1000	21125 (11.53%)
Day 8	125061	1000	11129 (8.90%)
Day 9	401922	1000	103326 (25.71%)
Day 10	206708	1000	27982 (13.54%)
Day 11	205790	1000	6502 (3.16%)

Table 5.2: Summary of data collected over the Gulf of Mexico[56], showing the number of instances (lidar shots), dimensions (samples per shot), and the number of instances containing either a fish, schools of fish, jellyfish, or plankton layers. The percentages of each instance type are shown in parentheses. Objects of interest often occur across multiple adjacent instances. Some instances contain multiple types of objects of interest.

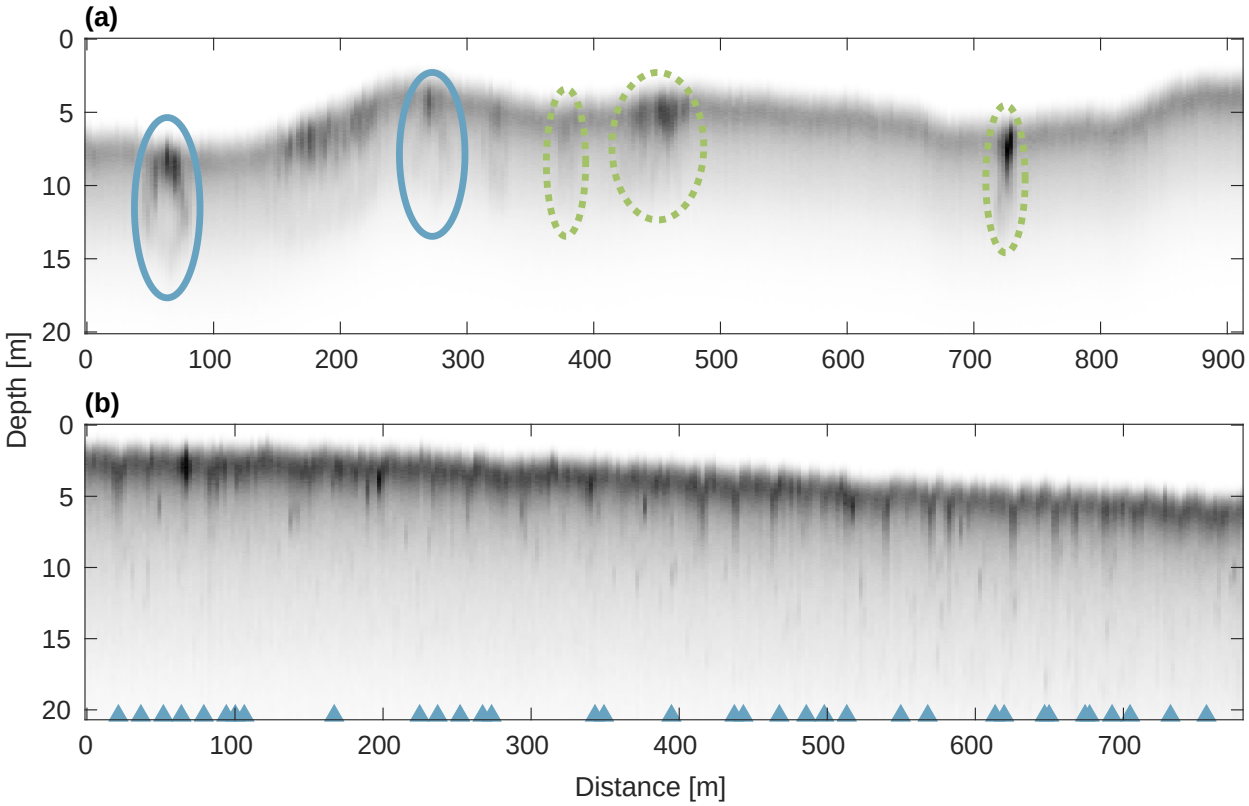


Figure 5.2: Example cross-polarized lidar data from the Gulf of Mexico. **(a)** Data from day 2. Hollow jellyfish aggregations [57] are annotated with blue ellipses, while fish schools are highlighted with green dotted ellipses. **(b)** Data from day 1. The blue triangles indicate lidar shots where a single fish was manually found and labeled.

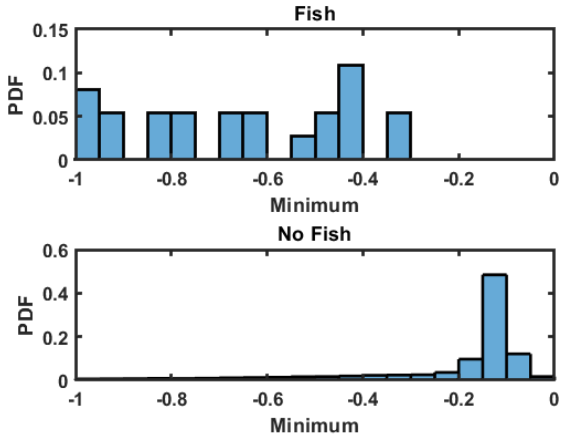
NOAA fish lidar, which used a flashlamp-pumped laser with 30 pulses/s at 100 mJ/pulse and 12 ns pulse width, also with separate co- and cross-polarized receivers with signals digitized at 1 Gsamples/s and 8-bit resolution with a logarithmic amplifier [56]. For both datasets, the cross-polarized signal was analyzed in this study because previous research has shown that the water volume scatter is predominantly in the co-polarized channel. Thus the cross-polarized signal has a higher contrast for fish detection [58].

Data Preprocessing

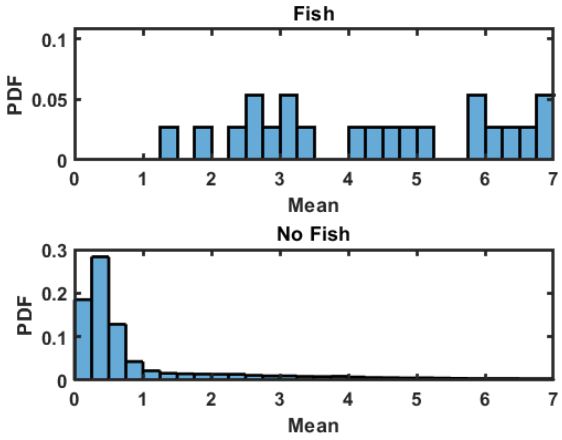
Data preprocessing is the first step when obtaining a new dataset. This step ensures that uninformative data is removed and in turn will improve the classification results. The same overall preprocessing procedures were completed on both lidar datasets.

The first two preprocessing steps used are surface detection and correction. These steps compensate for changes in airplane elevation to ensure that the rows of the lidar image correspond to the correct water depth. The surface of the water in each image was detected by finding the maximum return in the co-polarized channel, and then moving up toward the lidar until 25% of the maximum return was reached. Then, after the surface was detected in each shot, the surface was smoothed with a 10-tap moving average filter. To deal with noisy returns from the air in the Yellowstone data, the first 600 and 512 samples were skipped in the 2015 and 2016 data, respectively. Once the surface was found, each lidar shot was shifted up so the surface starts at the first row, then the lidar shot was padded with zeros at the bottom, maintaining the original number of rows.

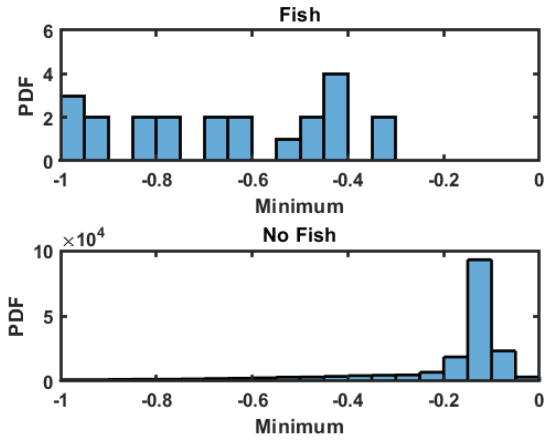
After surface correction, the height of the image was reduced based on the penetration depth of the lidar and the expected fish depth. In the Yellowstone Lake data, the image was reduced to 60 rows, which corresponds to a depth of approximately 8.65 meters. The Gulf of Mexico data was reduced to 150 rows, which corresponds to a depth of approximately 17.3 meters.



(a) Minimum depth distribution



(b) Depth mean distribution



(c) Non-normalized minimum depth distribution

Figure 5.3: Histograms of minimum and mean distributions in the time domain comparing the fish and non-fish classes from the Yellowstone Lake data

Statistical data was recorded for each lidar image in the training and testing sets in both the time and frequency domain following a discrete cosine transform. Histograms of all eighteen statistical measurements of the training set were examined to determine the most effective measurements to separate the non-fish class to one or several of the target

k	Number of Filtered Clusters	Non-Fish Images Removed	Fish Images Removed	Percent of Target Class Removed	Percent of Total Data Removed
2	1	139964	0	0%	72.82%
3	1	134870	0	0%	70.17%
4	2	133751	0	0%	69.59%
5	3	137585	0	0%	71.58%
6	4	140087	0	0%	72.89%
7	4	136729	0	0%	71.14%
8	5	141804	0	0%	73.78%
9	6	143776	0	0%	74.81%
10	7	147917	0	0%	76.96%
11	8	151076	0	0%	78.60%
12	8	144496	0	0%	75.18%
13	9	144390	0	0%	75.12%
14	10	145951	0	0%	75.94%
15	10	144922	0	0%	75.40%
Corresponding Test Cluster Summary Using Selected k Value					
2	1	87876	6	1.65%	80.78%
11	8	94000	31	8.52%	86.41%

Table 5.3: Clustering Summary for Yellowstone Lake Data

classes. The chosen statistical measurements for the Yellowstone Lake data are minimum, maximum, variance, range, standard deviation, and mean in the time domain and minimum, maximum, variance, range, standard deviation, mean, and kurtosis in the frequency domain. The chosen statistical measurements for the Gulf of Mexico data are maximum, range, standard deviation, and mean in the time domain and maximum, range, standard deviation, mean, and kurtosis in the frequency domain. Given that these measurements are from lidar data, the statistical measurements are based on the depths present in each individual shot. See Figure 5.3 for examples of distributions comparing the fish and non-fish class for the Yellowstone Lake data. The entire collection of histograms for the selected measurements for the Yellowstone Lake data and Gulf of Mexico data can be found in Appendix B. Figures 5.3a and 5.3b show the minimum and mean distributions comparing the fish and non-fish classes of the Yellowstone Lake data. From these distributions, we can predict that if a lidar shot has a small minimum and large mean, it is more likely to be contain a fish. It is important to note that these distributions are normalized, meaning that the fish histograms are shorter in height than the non-fish histogram given the class imbalance in the data. Figure 5.3c show the non-normalized minimum distribution to display the magnitude of difference in histogram height.

A parameter search for the optimum number of GMM clusters, k, was completed. In this search, the value of k was altered from 2 to 15. A full clustering summary for the Yellowstone Lake data can be found at Table 5.3 and the Gulf of Mexico data at Table 5.4.

In the Yellowstone Lake table, there are two different values of k that are highlighted. Shown in gray are the training and testing clustering summaries for $k = 2$ and shown in blue are the training and testing clustering summaries for $k = 11$. The cells highlighted in blue are notable because they represent the value of k that filters out the most non-target lidar images, meaning they do not contain fish. Here, a total of 78.60% and 86.41% of the total data is removed while removing 0% and 8.52% of the images containing fish from the training

k	Number of Filtered Clusters	Non-Fish Images Removed	Fish Images Removed	Percent of Target Class Removed	Percent of Total Data Removed
2	1	52136	124	1.40%	39.24%
3	2	56586	154	1.74%	42.59%
4	2	48760	100	1.13%	36.70%
5	3	59707	195	2.20%	44.94%
6	3	49528	94	1.06%	37.28%
7	5	59692	195	2.20%	44.93%
8	4	41856	60	0.68%	31.50%
9	5	56853	146	1.65%	42.79%
10	7	54528	105	1.19%	41.04%
11	8	57484	153	1.73%	43.26%
12	8	56899	147	1.66%	42.82%
13	9	56799	146	1.65%	42.75%
14	11	57063	144	1.63%	42.95%
15	11	58825	184	2.08%	44.27%
Corresponding Test Cluster Summary Using Selected k Value					
2	1	399340	8237	1.80%	14.27%
5	3	495385	15784	3.45%	17.71%

Table 5.4: Clustering Summary for Gulf of Mexico Data

and testing datasets, respectively. The cells highlighted in gray are also important because they represent the optimum k-value for a GMM model that can be trained unsupervised. Given that only one cluster is being filtered from the GMM model, we can simply remove the cluster containing the most images rather than examining the labeled data. This method yields a slightly less-optimal result of removing 72.82% and 80.78% of the total data, while removing 0% and 1.65% of the lidar images containing fish, respectively. Nonetheless, with a small increase in total removed data from $k = 2$ to $k = 11$ and the attractiveness of unsupervised learning, GMM models using 2 clusters are used for the remainder of the analysis.

In Table 5.4, the clustering summary for the Gulf of Mexico data is displayed. Similar to the Yellowstone Lake data, there are two different k-values for both the training and testing datasets highlighted. Likewise, the cells highlighted in gray represent the optimal k-value for unsupervised GMM clustering and the cells highlighted in blue represent the optimal k-value for supervised clustering when class labels are available. Here, when $k = 2$ the filtering step removes 39.24% and 14.27% of the images without fish while only removing 1.40% and 1.80% of the images containing fish of the training and testing sets, respectively. When $k = 5$, the filtering step removes 44.94% and 17.71% of the images without fish while only removing 2.20% and 3.45% of the images containing fish of the training and testing sets, respectively. When training the GMM model of the Gulf of Mexico data unsupervised, the only difference from the Yellowstone Lake data that needs to be noted is the cluster with the *fewest* lidar images will be the filtered cluster. Again, the unsupervised model is the parameter chosen moving forward in this research.

Classification Results

The selected statistical data recorded in the previous step was concatenated onto other predictors in the datasets prior to training and testing the classification algorithms. The

classifiers chosen for the fish lidar datasets are support vector machines (SVM), Naive Bayes, fine decision trees, and neural networks.

For each feature extraction method, classification models are generated and the model's accuracy, precision, recall, and F_3 score are used as metrics to quantify the model's effectiveness. Given the high class imbalance in the data, accuracy is not the most effective metric for estimating a model's effectiveness because the model's with the highest accuracy tend to predict only the non-target class. Furthermore, we are interested in models with high recall and precision to ensure that the models are indeed making accurate predictions on the target (fish) class. Also, the F_3 score is recorded which quantifies the precision and recall together in one, weighted-average value that places a higher weight on the recall than the precision.

In addition to the individual image, per shot, classification metrics, classification results were also calculated per-region. Here, labels were grouped into 1000-shot windows that were overlapped by 100 shots. A region was considered a region of interest (ROI) if at least one fish label was present for the true labels and at least ten fish were present for the predicted labels.

For the Gulf of Mexico dataset, there are eleven days of data collected in which one was used for training. To test the other ten days of data, the daily data was individually tested on each model and predicted label values were recorded. Then after all ten days were tested, the predicted labels and true labels were grouped and compared to generate accuracy metrics. The overall precision and recall values were calculated by taking a weighted sum of the individual classes' precision and recall values weighted by the number of fish in each class.

	Feature Extraction Method	SVM			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.9967	-	0	0
	Sparse Coding	0.9915	0.0610	0.1071	0.0996
	Frozen Dictionary	0.9957	0.2603	0.1566	0.1631
	Nothing	0.9929	0.1918	0.3462	0.3204
ROI	PCA	0.9732	-	0	0
	Sparse Coding	0.8926	0.4615	0.5000	0.4959
	Frozen Dictionary	0.9174	0.6250	0.4167	0.4311
	Nothing	0.8843	0.4500	0.7500	0.7031

	Feature Extraction Method	Naive Bayes			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.9967	-	0	0
	Sparse Coding	0.9046	0.0191	0.5467	0.1453
	Frozen Dictionary	0.8979	0.0155	0.4725	0.1197
	Nothing	0.9308	0.0249	0.5165	0.1737
ROI	PCA	0.9732	-	0	0
	Sparse Coding	0.6446	0.2182	1	0.7362
	Frozen Dictionary	0.7025	0.2500	1	0.7692
	Nothing	0.7521	0.2857	1	0.8000

	Feature Extraction Method	Decision Tree			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.9967	-	0	0
	Sparse Coding	0.9957	0.0093	0.0028	0.0030
	Frozen Dictionary	0.9963	0.2688	0.0687	0.0742
	Nothing	0.9963	0.1111	0.0165	0.0180
ROI	PCA	0.9732	-	0	0
	Sparse Coding	0.9008	0.5000	0.1667	0.1786
	Frozen Dictionary	0.9339	1	0.3333	0.3571
	Nothing	0.9091	1	0.0833	0.0917

	Feature Extraction Method	NN			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.9967	-	0	0
	Sparse Coding	0.9736	0.0124	0.0879	0.0546
	Frozen Dictionary	0.9728	0.0242	0.1813	0.1099
	Nothing	0.9784	0.0217	0.1236	0.0841
ROI	PCA	0.9732	-	0	0
	Sparse Coding	0.7769	0.2000	0.4167	0.3760
	Frozen Dictionary	0.7686	0.1923	0.4167	0.3732
	Nothing	0.8017	0.2000	0.3333	0.3125

Table 5.5: Classification test results for each feature extraction method and classification algorithm for the Yellowstone Lake lidar data.

Yellowstone Lake

Table 5.5 show the classification test results for SVM, Naive Bayes, decision trees, and neural networks for each feature extraction method. The highlighted cells represent the best algorithm combinations for each classifier. Highlighted in green shows the best results out of all of the classifiers for the Yellowstone Lake data. Furthermore, this combination is the individual and ROI classification results for the data concatenated with the selected statistical features labeled by a Naive Bayes classifier. The confusion matrices for this combination can be found at Figure 5.4. Here, the individual classifications had a recall of 51.65% and a precision of 2.49%. From a storage perspective, this means we would remove 93.07% of the total data while retaining 51.65% of the fish data if the predicted non-fish column was filtered. For the ROI classification results, a precision of 28.57% and recall of 100% were achieved. Meaning that 65.29% of the total data would be removed without losing any of the fish data.

False positive analysis: Again, the precision and recall for ROI classification results were 28.57% and 100%, respectively. Or in other words, there were 30 false-positive predictions and 0 false-negative predictions when using a Naive Bayes classifier. To further understand the behavior of our model, we analyzed the 30 false-positive predictions. In this analysis, Dr. Joseph A. Shaw, a professor at Montana State University and a participating researcher in the Yellowstone Lake data collection, assisted us [16]. We concluded that up to 11 of the 30 ROI shots labeled as false-positives, possibly contain fish. An updated confusion matrix for these results can be found in Figure 5.4c. With this update, the precision and accuracy are updated to 54.76% and 84.30%, respectively. This is a significant improvement over the prior analysis with the precision value nearly doubling. However, instances such as this do not arise often as we usually accept all labeled data as being labeled with 100% certainty. But this is usually not the case and there should be some labeling or human error to be expected.

	Predicted Non-Fish (**0**)	Predicted Fish (**1**)	
Actual Non-Fish (**0**)	101060 92.91%	7354 6.76%	93.22%
Actual Fish (**1**)	176 0.16%	188 0.17%	51.65%
	99.82%	2.49%	93.08%

(a) Individual

	Predicted Non-Fish (**0**)	Predicted Fish (**1**)	
Actual Non-Fish (**0**)	79 65.29%	30 24.79%	72.48%
Actual Fish (**1**)	0 0.00%	12 9.92%	100.00%
	100.00%	28.57%	75.21%

(b) ROI

	Predicted Non-Fish (**0**)	Predicted Fish (**1**)	
Actual Non-Fish (**0**)	79 65.29%	19 15.70%	80.61%
Actual Fish (**1**)	0 1.74%	23 19.01%	100.00%
	100.00%	54.76%	84.30%

(c) Updated ROI

Figure 5.4: Confusion matrix of the individual data, corresponding region of interest, and adjusted region of interest following further false positive analysis of the Yellowstone Lake data labeled by a Naive Bayes classifier. Results of this algorithm combination can be found in Figure 5.5.

Therefore, it is important to utilize manual human examination and computers together in an iterative fashion to achieve the best results when analyzing and labeling new data. Also, this analysis highlights the difficulty of quantifying the effectiveness of a model based on one number (e.g. F_3-score) or a collection of numbers (e.g. precision, recall, accuracy) when there is mislabeled data present. Although technically invalid because we altered the true label values, the updated ROI results are significantly better than the original ROI results. This fact is important to note and one to take in to consideration when analyzing data in the future. In conclusion, this analysis proved that the machine learning pipeline used in this work can overcome some level of human error present when labeling the data.

Gulf of Mexico

The classification results for the Gulf of Mexico data can be referenced in Table 5.6. Similarly, the highlighted cells represent the best feature extraction and classification method combinations for each classifier where the green cells are the best overall results for the dataset. Here, the best results are achieved by classifying the raw data with the statistical features using a support vector machine classifier. Individually, this combination is able to achieve an 18.19% precision and 73.47% recall. When the predicted labels are grouped together, the region of interest results improve to a precision of 53.69% and recall of 96.32%. A confusion matrix for the ROI results can be found at Figure 5.5. Here, 15.37% of the total data can be removed while retaining 96.32% of the fish data.

Discussion and Conclusion

From analyzing the results of the two datasets, it is determined that the pipeline is able to filter a large portion of the data effectively; however, the classification models are not accurate enough to be a completely autonomous system. Instead, this pipeline can help scientists with initial screenings of the data to reduce manual filtering time and energy. Also,

	Feature Extraction Method	SVM			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.8373	-	0	0
	Sparse Coding	0.5452	0.1933	0.5575	0.4691
	Frozen Dictionary	0.5458	0.1973	0.5890	0.4914
	Nothing	0.4351	0.1819	0.7347	0.5635
ROI	PCA	0.9153	-	0	0
	Sparse Coding	0.6193	0.5569	0.9441	0.8827
	Frozen Dictionary	0.6132	0.5537	0.9284	0.8696
	Nothing	0.5907	0.5369	0.9632	0.8923

	Feature Extraction Method	Naive Bayes			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.8373	-	0	0
	Sparse Coding	0.3958	0.0876	0.6057	0.3806
	Frozen Dictionary	0.4464	0.0775	0.6266	0.3668
	Nothing	0.4290	0.1340	0.4431	0.3600
ROI	PCA	0.9153	-	0	0
	Sparse Coding	0.6264	0.5633	0.9257	0.8697
	Frozen Dictionary	0.6331	0.5698	0.9066	0.8560
	Nothing	0.6273	0.5686	0.8698	0.8260

	Feature Extraction Method	Decision Tree			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.8373	-	0	0
	Sparse Coding	0.5305	0.1852	0.5449	0.4563
	Frozen Dictionary	0.5425	0.1748	0.5062	0.4255
	Nothing	0.5647	0.1605	0.3693	0.3268
ROI	PCA	0.9153	-	0	0
	Sparse Coding	0.6019	0.5450	0.9455	0.8808
	Frozen Dictionary	0.6090	0.5521	0.9059	0.8513
	Nothing	0.6392	0.5755	0.8964	0.8491

	Feature Extraction Method	NN			
		Accuracy	Precision	Recall	F_3
Individual	PCA	0.8373	-	0	0
	Sparse Coding	0.7042	0.2025	0.3120	0.2960
	Frozen Dictionary	0.6108	0.2037	0.5014	0.4375
	Nothing	0.5464	0.1686	0.4713	0.3996
ROI	PCA	0.9153	-	0	0
	Sparse Coding	0.6424	0.5753	0.9243	0.8714
	Frozen Dictionary	0.6045	0.5476	0.9298	0.8691
	Nothing	0.5929	0.5410	0.9046	0.8476

Table 5.6: Classification test results for each feature extraction method and classification algorithm for the Gulf of Mexico lidar data.

	Predicted Non-Fish (**0**)	Predicted Fish (**1**)	
Actual Non-Fish (**0**)	424 13.63%	1219 39.20%	25.81%
Actual Fish (**1**)	54 1.74%	1413 45.43%	96.32%
	88.70%	53.69%	59.07%

Figure 5.5: Confusion matrix of the region of interest for the Gulf of Mexico dataset labeled by an SVM classifier. Results of this algorithm combination can be found in Figure 5.6

with a high recall rate for the ROI results in both datasets, the scientists can be sure that the models will retain most of the target images while removing a portion of the unimportant images. This observation is especially true with the Yellowstone Lake dataset, where the model is able to remove a majority of the dataset and not lose any regions of interest at all. On the other hand, the Gulf of Mexico data does not remove a majority of the non-target area of interest data, but with a large testing set size of $2,796,238$ individual images and $3,090$ areas of interest, the time and storage savings are significant in filtering unimportant data.

In this project, the ROI analysis is very important given that it is more practical than individual shot analysis in identifying marine life in bodies of water. Therefore, the goal of this work was to achieve sufficient ROI classification results to make the data analysis of lidar datasets more efficient and effective. In Figure 5.6 we show examples of fish instances in Yellowstone Lake where true positive, false positive, and false negative labels were predicted in ROI analysis. In Figure 5.6a, an example of a true positive prediction is shown. Here, there are three fish in the lidar shot and our model, although not perfect, predicts fish close enough to the actual fish to identify this shot as an ROI. Figure 5.6b shows a false positive

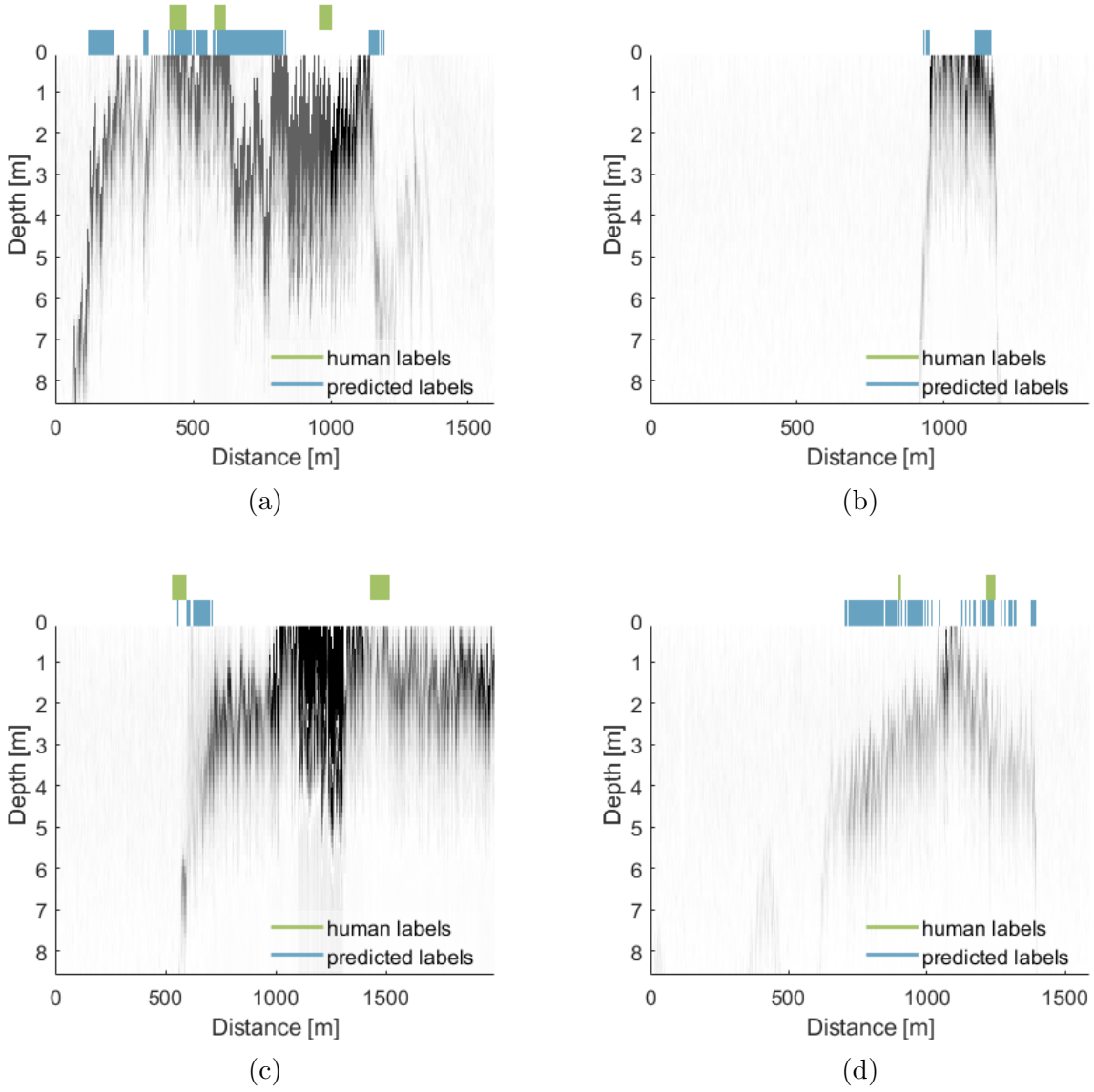


Figure 5.6: Shown here are examples of true positive, false positive, and false negative fish instances in the Yellowstone Lake data. Specifically, (a) is a true positive, (b) is a false positive, (c) is a true positive and false negative, and (d) is a true positive.

example where the model predicted there are two fish in the shot when there are in fact no fish at all. Figure 5.6c shows both a true positive and false negative example in one lidar shot. In this example, there are two fish in the shot, but our model was only able to correctly identify one of them. Given that our ROI model had 100% recall, shots like this with both a true positive and false negative instance are going to be the only examples of shots with false negatives. In our pipeline, false negatives are the worst case scenario given the rare target class, therefore we want to preserve this data. Lastly, Figure 5.6d shows another true positive example where the model was able to identify approximately where the two fish are in this lidar shot.

Similarly, Figure 5.7 shows examples of true positive, false positive, and false negative ROI predictions for day eight in the Gulf of Mexico testing set. Here, Figure 5.7a shows a false negative instance where there is some identified marine life around 300m in distance that our model missed. On the other hand, Figure 5.7b shows a false positive example where the model predicted several fish or other marine life where there were not any at all in this shot. Figures 5.7c and 5.7d show two true positive examples from the Gulf of Mexico data. Furthermore, Figure 5.7c does a sufficient job of identifying the important regions of the lidar shot. But, Figure 5.7d identifies one region correctly at distances less than 500m and then predicts several false positives in the rest of the lidar shot. With the high recall results here, we can confidently conclude there will be few false negative examples like in Figure 5.7a. But, with the precision being around 50%, there will be several cases of examples like Figure 5.7b and Figure 5.7d from distance 500m and greater.

In conclusion, the pipeline used for these two lidar datasets serves as a great starting step for filtering unwanted data and in turn, reduces the amount of manual filtering. Although the classification results are not sufficient to be completely autonomous, this model provides real value to individuals analyzing large airborne lidar datasets with a high class imbalance.

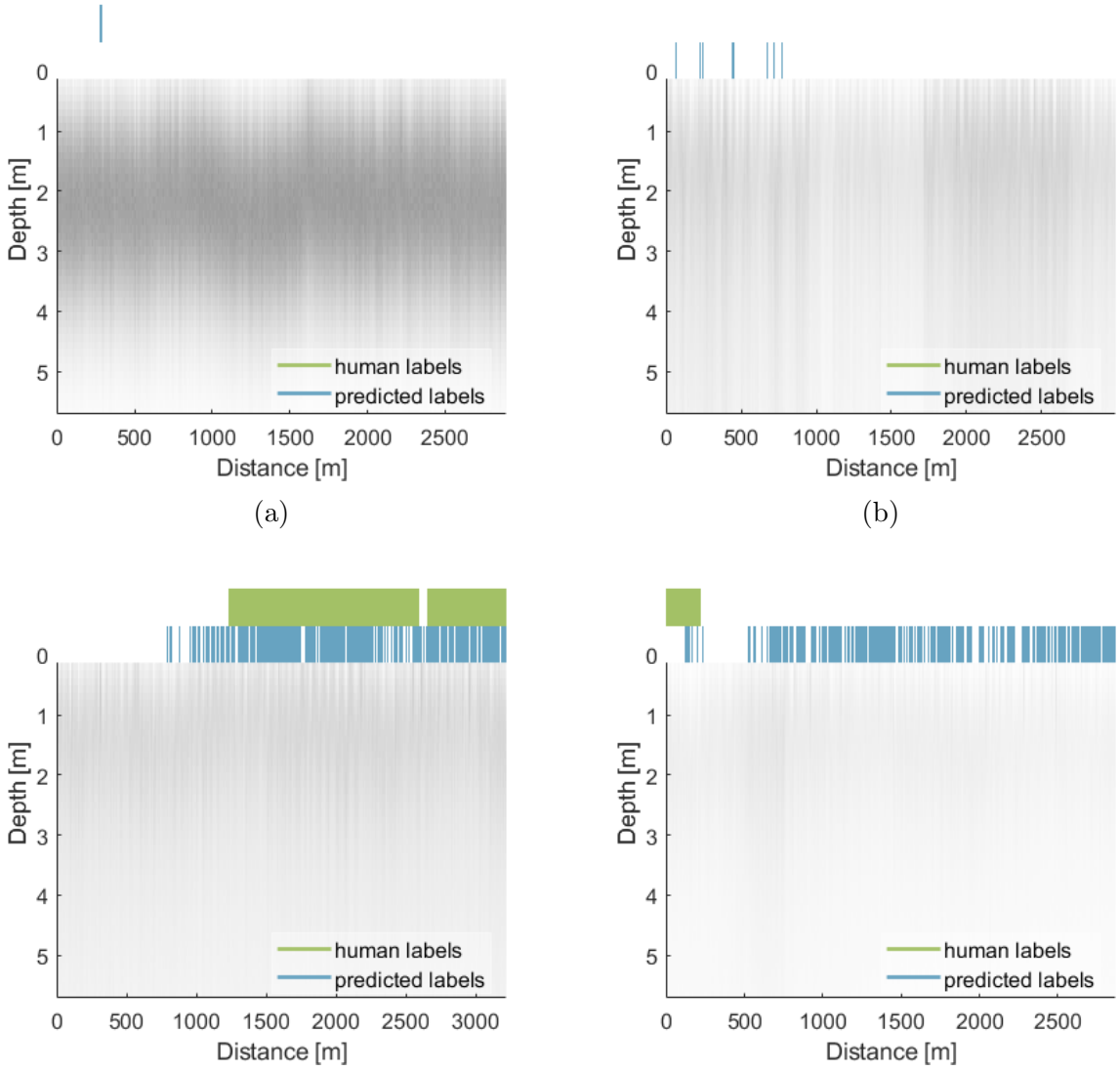


Figure 5.7: Shown here are examples of true positive, false positive, and false negative fish instances in the Gulf of Mexico data. Specifically, (a) is a false negative, (b) is a false positive, (c) is a true positive, and (d) is a true positive.

THESIS SUMMARY

In this work, a pipeline was developed for rare-event detection in SAR and lidar data. The pipeline was developed and tested for an SAR dataset surveying the surface of Venus. Here, we identified volcanoes in dated, low-quality images with a high class imbalance present in the data. This combination served to be difficult in designing an autonomous machine learning pipeline with sufficient classification results. To increase the robustness of the pipeline, the model was trained and tested on two lidar datasets containing geographical surveys of Yellowstone Lake and the Gulf of Mexico. Here, we identified invasive lake trout in Yellowstone Lake and flying fish, schools of fish, jellyfish, and plankton layers in the Gulf of Mexico.

The machine learning pipeline consisted of data preprocessing, clustering, feature extraction, and finally, classification. Examples of preprocessing steps used in this work include: reshaping, applying sliding window function, removing invalid numbers, surface detection and correction, generating and concatenating statistical features, and window resizing. Additionally, GMMs were used to cluster the datasets into a specified number of clusters in an unsupervised manner. The number of clusters were tested for a variety of values to determine the optimal parameter for each application. Then, diverse combinations of feature extraction and classification algorithms were tested and recorded to find the most effective algorithms for each dataset. Given the high class imbalance present in the datasets, precision, recall, and F_3-score were the primary metrics used to quantify the success of each algorithm combination.

For the volcano datasets, the sliding window results were trained, tested, analyzed and mapped back to the larger, mother image. This method of training and analyzing sub-images allows us to not only identify the presence of volcanoes in the mother image, but also the location. Similarly, the fish lidar datasets were trained, tested, analyzed and mapped to

areas of interest by combining individual neighbor images. Then, regions of interest were identified and assigned accuracy metrics. The classification results for the mother images in the volcano analysis and ROIs in the fish analysis were the most meaningful from a real-world, research perspective. Typically, when scientists perform a geographical survey with one of these two optical sensors, they are more interested in the overall behavior of the landscape and region characteristics rather than very fine resolution, individual small-area characteristics. Hence, our emphasis on providing analysis on the small area shots, but then scaling them back to an overall behavior of the two applications. In the mother images and ROIs, the pipeline primarily identifies important data (images with volcanoes and fish), but includes several false positive predictions as well. Therefore, the classification results are not sufficient for a complete autonomous system for either volcano or fish research. Instead, the pipeline serves as an effective step of the data collection process to remove a significant portion of the unimportant data. This initial filtering will greatly reduce the scale of manual filtering required and is especially useful in optics applications like these when the amount of data collected is extravagant.

REFERENCES CITED

[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[2] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[3] J. H. Churnside, R. Wells, K. M. Boswell, J. A. Quinlan, R. D. Marchbanks, B. J. McCarty, and T. T. Sutton, "Surveying the distribution and abundance of flying fishes and other epipelagics in the northern gulf of mexico using airborne lidar," *Bulletin of Marine Science*, vol. 93, no. 2, pp. 591–609, 2017.

[4] NOAA, "What is lidar?," National Ocean Service website, 2020.

[5] M. Soumekh, *Synthetic aperture radar signal processing*, vol. 7. New York: Wiley, 1999.

[6] M. A. Lefsky, W. B. Cohen, G. G. Parker, and D. J. Harding, "Lidar remote sensing for ecosystem studies: Lidar, an emerging remote sensing technology that directly measures the three-dimensional distribution of plant canopies, can accurately estimate vegetation structural attributes and should be of particular interest to forest, landscape, and global ecologists," *BioScience*, vol. 52, no. 1, pp. 19–30, 2002.

[7] F. F. Sabins Jr, *Remote sensing–principles and interpretation*. WH Freeman and company, 1987.

[8] S. M. Young, "National land imaging program," tech. rep., US Geological Survey, 2020.

[9] M. R. Chmielewski and J. W. Grzymala-Busse, "Global discretization of continuous attributes as preprocessing for machine learning," *International journal of approximate reasoning*, vol. 15, no. 4, pp. 319–331, 1996.

[10] C. J. Gleason and J. Im, "Forest biomass estimation from airborne lidar data using machine learning approaches," *Remote Sensing of Environment*, vol. 125, pp. 80–91, 2012.

[11] P. Angelov, *Autonomous Learning Systems*. Wiley Online Library, 2013.

[12] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surv.*, vol. 49, pp. 31:1–31:50, Aug. 2016.

[13] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 20–27, IEEE, 2005.

[14] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[15] F. Mena, "Volcanoes on venus," 2018.

[16] M. R. Roddewig, J. H. Churnside, F. R. Hauer, J. Williams, P. E. Bigelow, T. M. Koel, and J. A. Shaw, "Airborne lidar detection and mapping of invasive lake trout in yellowstone lake," *Applied Optics*, vol. 57, p. 4111, May 2018.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[18] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[19] C. E. Rasmussen *et al.*, "The infinite gaussian mixture model.," in *NIPS*, vol. 12, pp. 554–560, 1999.

[20] G. Xuan, W. Zhang, and P. Chai, "Em algorithms of gaussian mixture model and hidden markov model," in *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, vol. 1, pp. 145–148, IEEE, 2001.

[21] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[22] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*, pp. 80–111, Springer, 1981.

[23] R. M. Haralick and K. S. Shanmugam, "Combined spectral and spatial processing of ERTS imagery data," *Remote Sensing of Environment*, vol. 3, no. 1, pp. 3 – 13, 1974.

[24] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[25] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[26] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014.

[27] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1794–1801, 6 2009.

[28] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, pp. 4311–4322, 11 2006.

[29] B. T. Carroll, B. M. Whitaker, W. Dayley, and D. V. Anderson, "Outlier learning via augmented frozen dictionaries," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1207–1215, 6 2017.

[30] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[31] I. Stephen, "Perceptron-based learning algorithms," *IEEE Transactions on neural networks*, vol. 50, no. 2, p. 179, 1990.

[32] F. Meyer, "Spaceborne synthetic aperture radar: Principles, data access, and basic processing techniques," *The SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation; NASA: Washington, DC, USA*, 2019.

[33] T. P. Scofield and B. M. Whitaker, "Machine learning pipeline for shift-invariant detection of volcanoes on venus," in *2020 Intermountain Engineering, Technology and Computing (IETC)*, pp. 1–6, IEEE, 2020.

[34] T. P. Scofield, J. Belford, J. H. Churnside, M. R. Roddewig, J. A. Shaw, and B. M. Whitaker, "Applying Gaussian Mixture Models to Detect Fish from Airborne LiDAR Measurements," Submitted.

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

[36] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[37] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, 2009.

[38] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[39] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, pp. 28–31, IEEE, 2004.

[40] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[41] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[42] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *icml*, vol. 99, pp. 124–133, Citeseer, 1999.

[43] A. Navlani, "Decision tree classification in python," datacamp, 2018.

[44] E. Kolçe and N. Frasheri, "The use of heuristics in decision tree learning optimization," *International Journal of Computer Engineering in Research Trends*, vol. 1, no. 3, pp. 127–130, 2014.

[45] W. Commons, "Artificial neural network," 2006.

[46] M. Ranzato and Y. LeCun, "A sparse and locally shift invariant feature extractor applied to document images," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, pp. 1213–1217, IEEE, 2007.

[47] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. Lecun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer*

Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pp. 1–8, IEEE, 2007.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[49] N. Martinson, "Obstacle avoidance guidance and control algorithm for spacecraft maneuvers," in *AIAA Guidance, Navigation, and Control Conference*, p. 5672, 2009.

[50] A. Jagannath, J. Jagannath, and A. Drozd, "Artificial intelligence-based cognitive cross-layer decision engine for next-generation space mission," in *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, pp. 1–6, June 2019.

[51] A. S. Wu and I. I. Garibay, "Intelligent automated control of life support systems using proportional representations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, pp. 1423–1434, June 2004.

[52] R. Arora, Z. E. Fleetwood, E. X. Zhang, N. E. Lourenco, J. D. Cressler, D. M. Fleetwood, R. D. Schrimpf, A. K. Sutton, G. Freeman, and B. Greene, "Impact of technology scaling in sub-100 nm nmosfets on total-dose radiation response and hot-carrier reliability," *IEEE Transactions on Nuclear Science*, vol. 61, pp. 1426–1432, June 2014.

[53] J. A. Hogan, R. J. Weber, and B. J. LaMeres, "Reliability analysis of field-programmable gate-array-based space computer architectures," *Journal of Aerospace Information Systems*, vol. 14, no. 4, pp. 247–258, 2017.

[54] J. S. Hane, B. J. LaMeres, T. Kaiser, R. Weber, and T. Buerkle, "Increasing radiation tolerance of field-programmable-gate-array-based computers through redundancy and environmental awareness," *Journal of Aerospace Information Systems*, vol. 11, no. 2, pp. 68–81, 2014.

[55] M. R. Roddewig, N. J. Pust, J. H. Churnside, and J. A. Shaw, "Dual-polarization airborne lidar for freshwater fisheries management and research," *Optical Engineering*, vol. 56, p. 031221, Mar. 2017.

[56] J. H. Churnside, R. D. Wells, K. M. Boswell, J. A. Quinlan, R. D. Marchbanks, B. J. McCarty, and T. T. Sutton, "Surveying the distribution and abundance of flying fishes and other epipelagics in the northern gulf of mexico using airborne lidar," *Bulletin of Marine Science*, vol. 93, pp. 591–609, apr 2017.

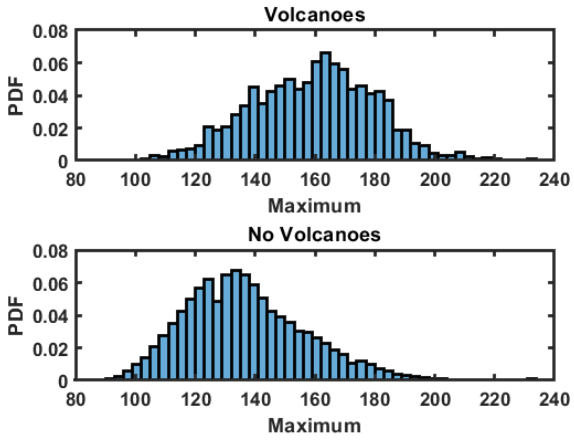
[57] J. H. Churnside, R. D. Marchbanks, P. L. Donaghay, J. M. Sullivan, W. M. Graham, and R. J. D. Wells, "Hollow aggregations of moon jellyfish (aureliaspp.)," *Journal of Plankton Research*, vol. 38, pp. 122–130, nov 2015.

[58] J. H. Churnside, "Review of profiling oceanographic lidar," *Optical Engineering*, vol. 53, p. 051405, dec 2013.

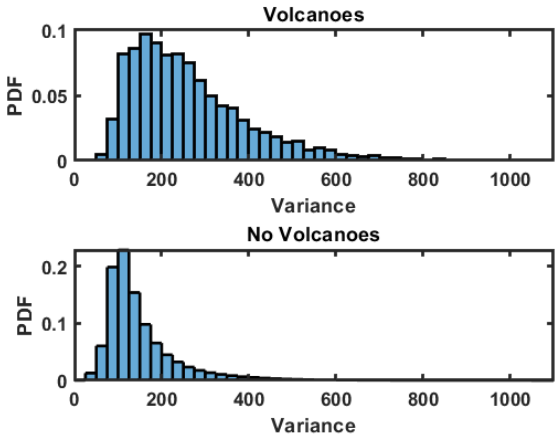
APPENDICES

APPENDIX A

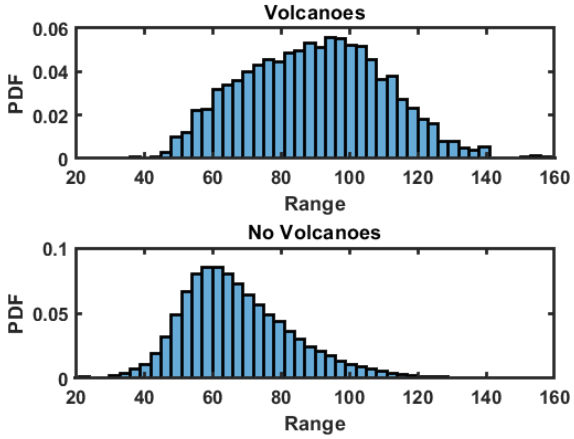
VOLCANOES ON VENUS HISTOGRAM DISTRIBUTIONS



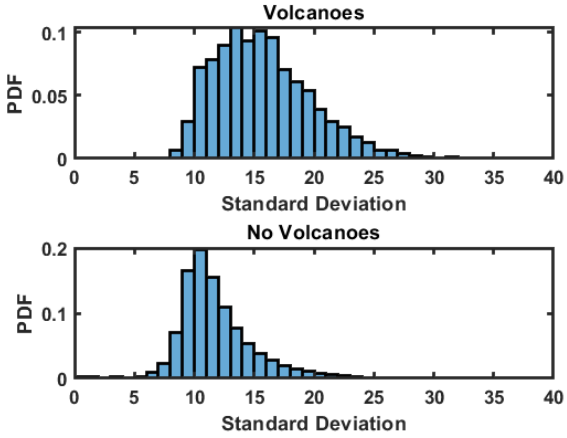
(a) Maximum elevation distribution



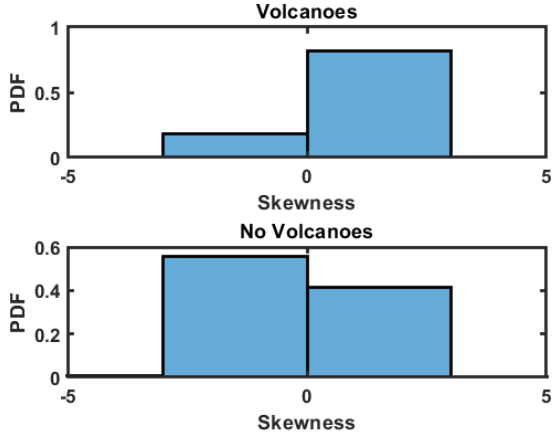
(b) Elevation variance distribution



(c) Elevation range distribution

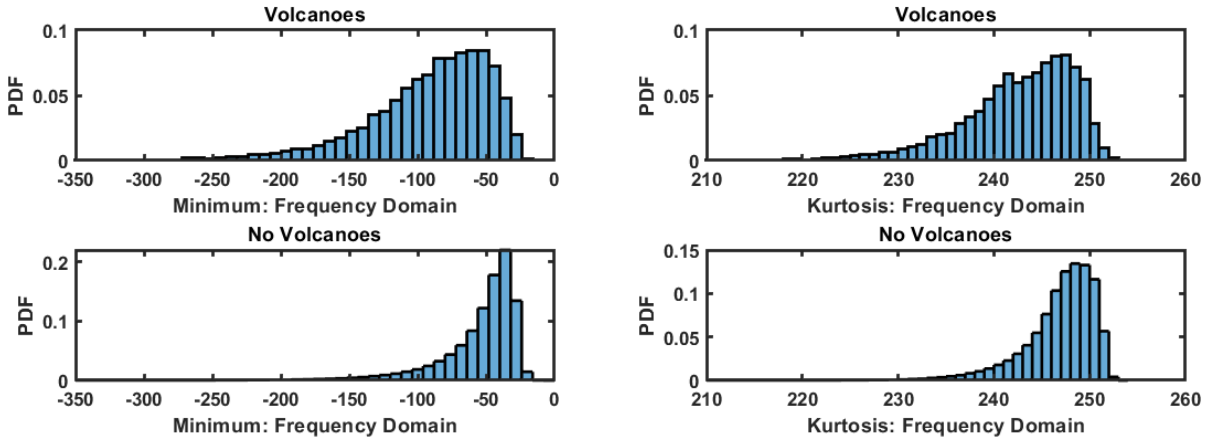


(d) Elevation standard deviation distribution



(e) Elevation skewness distribution

Figure A.1: Histograms of the selected statistical measurements in the time domain of the 16×16 pixel dataset comparing the volcano and non-volcano classes.



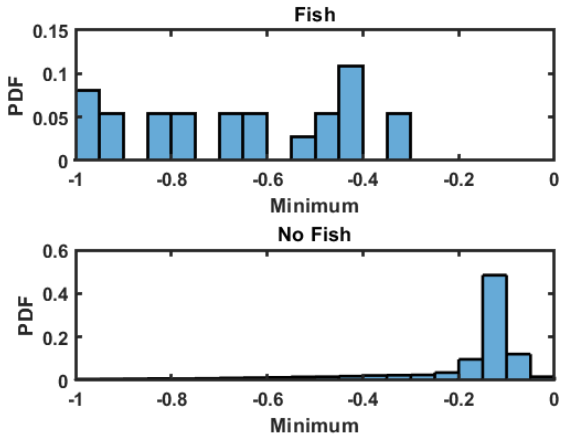
(a) Minimum elevation distribution

(b) Elevation kurtosis distribution

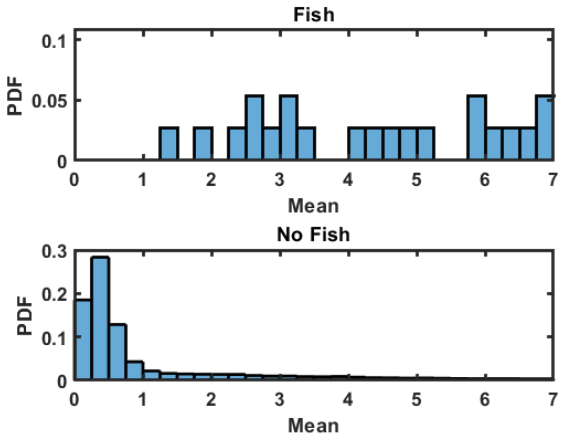
Figure A.2: Histograms of selected statistical measurements in the frequency domain of the 16×16 pixel dataset comparing the volcano and non-volcano classes

APPENDIX B

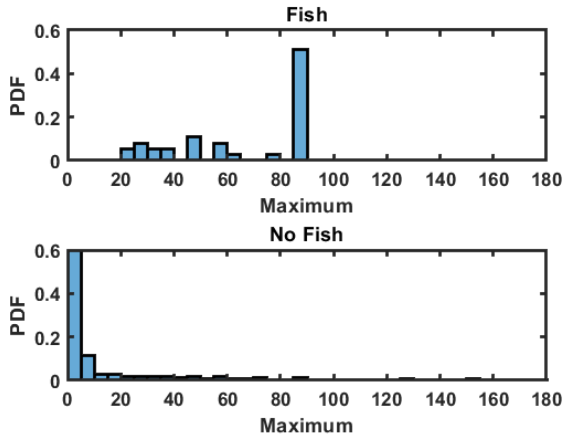
FISH LIDAR HISTOGRAM DISTRIBUTIONS



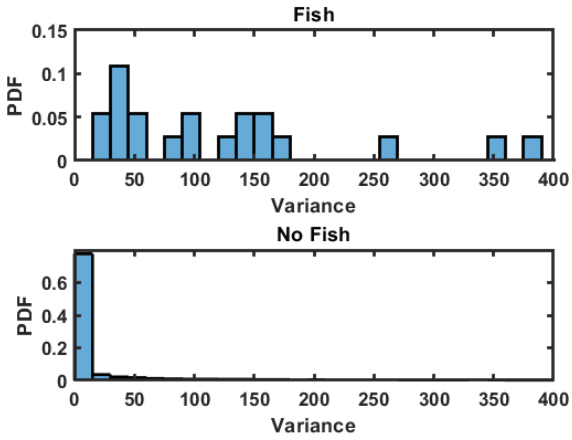
(a) Minimum depth distribution



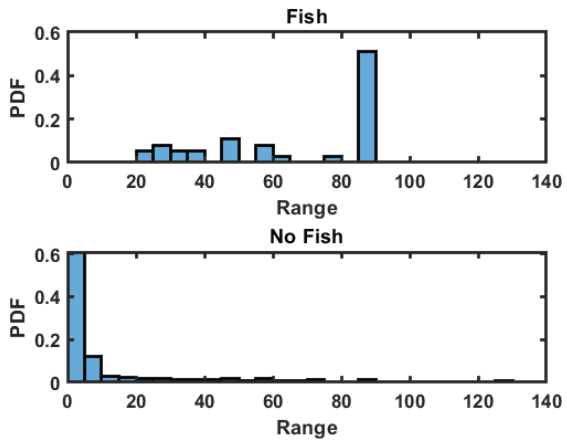
(b) Depth mean distribution



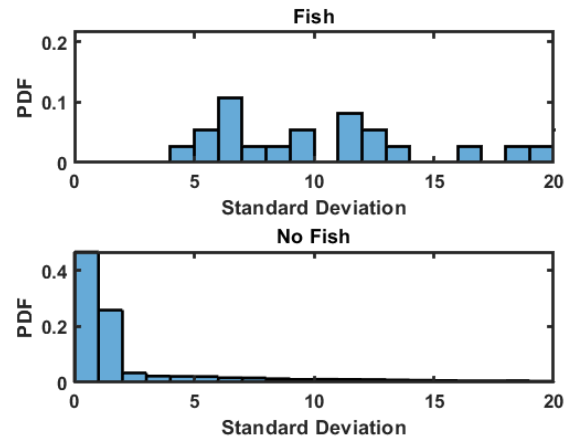
(c) Maximum depth distribution



(d) Depth variance distribution

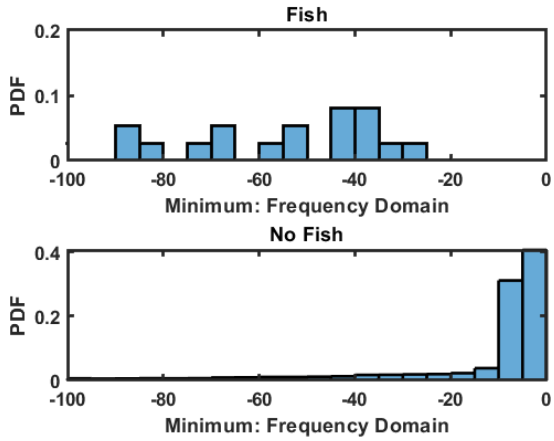


(e) Depth range distribution

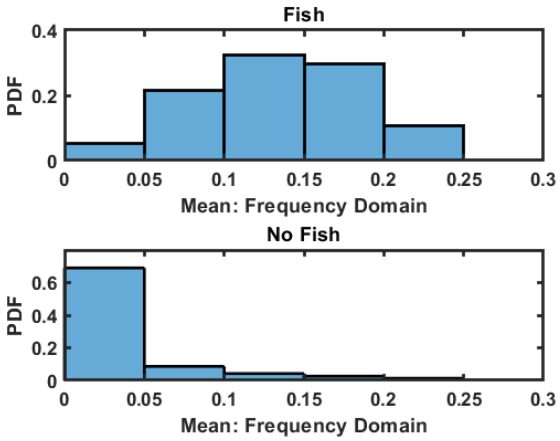


(f) Depth standard deviation distribution

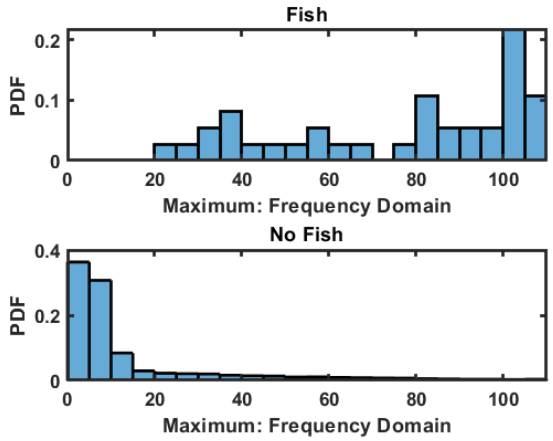
Figure B.1: Histograms of selected statistical measurements in the time domain comparing the fish and non-fish classes from the Yellowstone Lake data



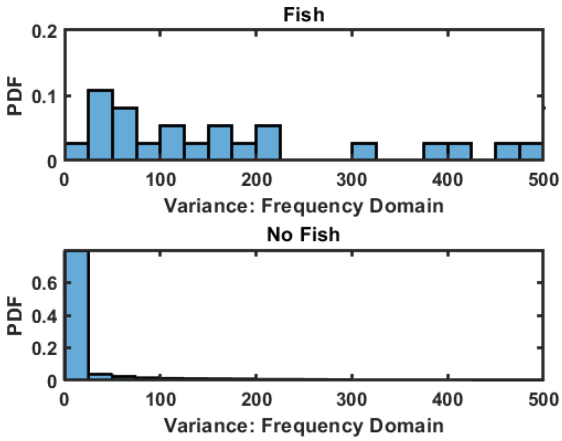
(a) Minimum depth distribution



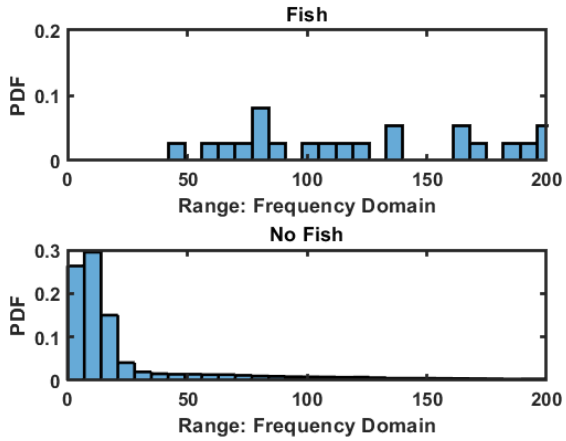
(b) Depth mean distribution



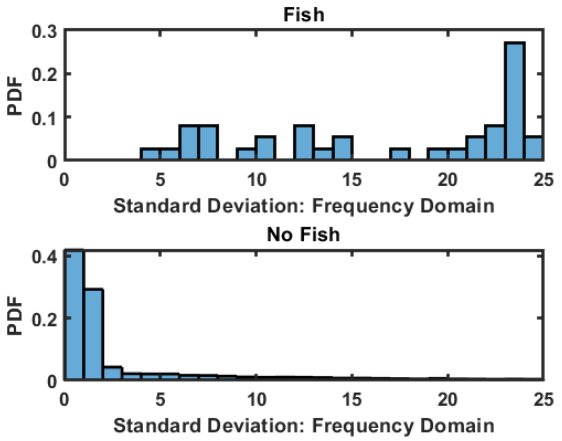
(c) Maximum depth distribution



(d) Depth variance distribution

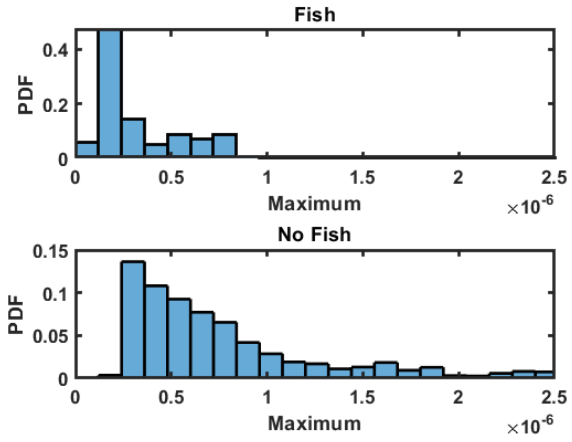


(e) Depth range distribution

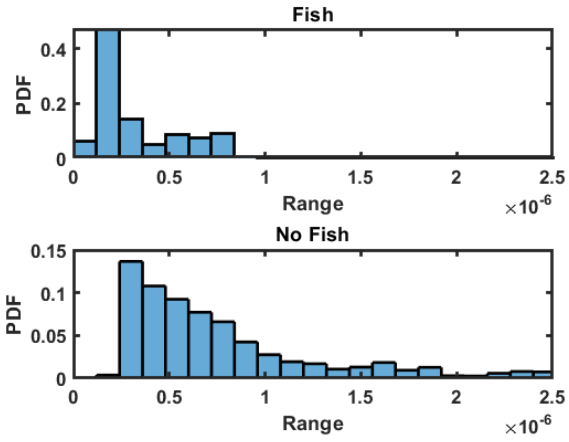


(f) Depth standard deviation distribution

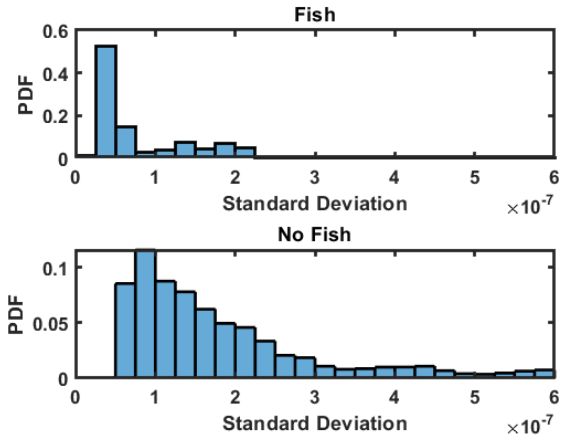
Figure B.2: Histograms of selected statistical measurements in the frequency domain comparing the fish and non-fish classes from the Yellowstone Lake data



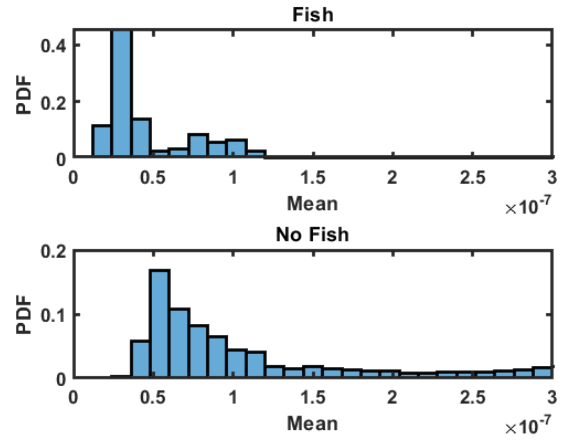
(a) Maximum depth distribution



(b) Depth range distribution

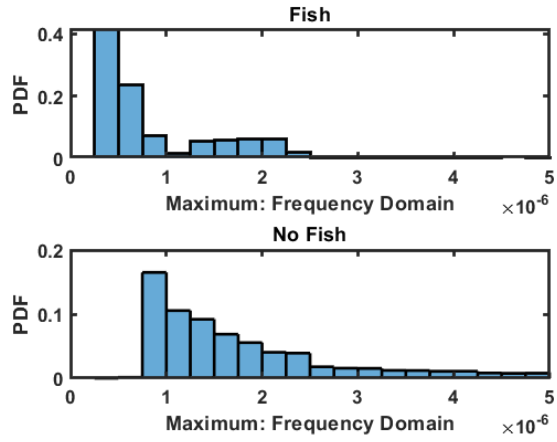


(c) Depth standard deviation distribution

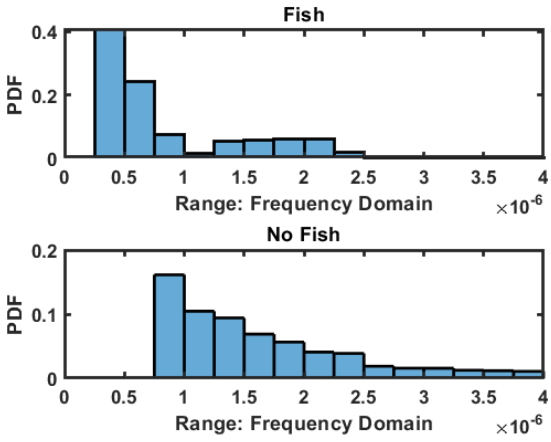


(d) Depth mean distribution

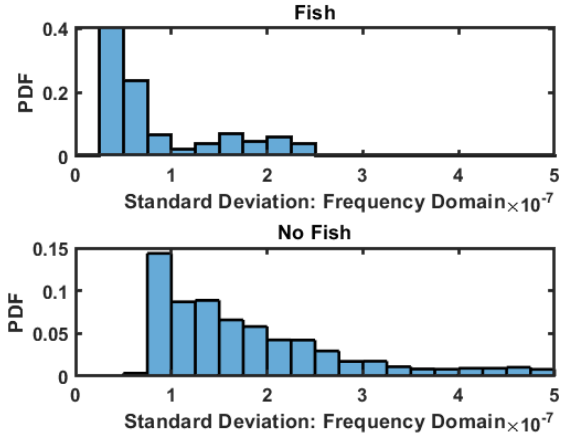
Figure B.3: Histograms of selected statistical measurements in the time domain comparing the fish and non-fish classes from the Gulf of Mexico data



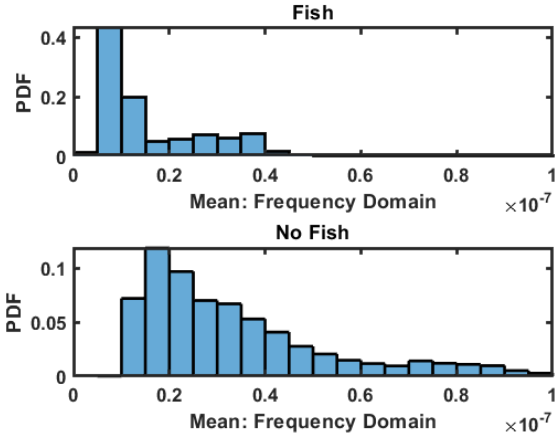
(a) Maximum depth distribution



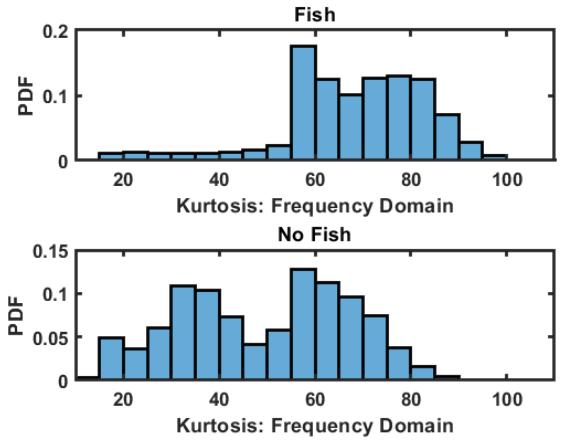
(b) Depth range distribution



(c) Depth standard deviation distribution



(d) Depth mean distribution



(e) Depth kurtosis distribution

Figure B.4: Histograms of selected statistical measurements in the frequency domain comparing the fish and non-fish classes from the Gulf of Mexico data