UNIVERSITY OF CALIFORNIA SAN DIEGO

Semi-Supervised Semantic Segmentation in UAV Imagery

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Ashlesha Vaidya

Committee in charge:

Professor Ryan Kastner, Chair
Professor Julian McAuley
Professor Curt Schurgers

2020

The thesis of Ashlesha Vaidya is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2020

DEDICATION

I dedicate this thesis to my family for their constant support and inspiration

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

UAV Unmanned Aerial Vehicle

SSL Semi-Supervised Learning

GIS Geographic Information System

GMM Gaussian Mixture Model

EM Expectation Maximization

CAE Convolutional Autoencoder

ELU Exponential Linear Unit

MSE Mean Squared Error

DBSCAN Density-Based Spatial Clustering of Applications with Noise

IoU Intersection over Union

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Firstly, I would like to thank Prof. Ryan Kastner for giving me the opportunity to work with him in the E4E Mangrove Monitoring team, for his support and invaluable feedback throughout the course of the project. I would also like to thank Dillon Hicks, the technical lead on the Mangrove Monitoring team for getting me set up at the start of my research. He has always been proactive about answering my questions and has generously shared his knowledge which help me a lot during the entire course of my project. A big thank you to all the E4E researchers as their work has inspired me in my research and given me direction when I was lost. I also thank Prof. Curt Schugers and Prof. Julian McAuley for their willingness to join my thesis committee and giving me feedback on my thesis.

Lastly, I would also like to thank my family including Anjali Vaidya, Vivek Vaidya, Devansh Zurale, Subhaga Zurale and Avinash Zurale for their constant support and motivation. A big thank you to all my friends who always inculcated a positivity in my personal life which has always helped me move forward with my research.

ABSTRACT OF THE THESIS

Semi-Supervised Semantic Segmentation in UAV Imagery

by

Ashlesha Vaidya

Master of Science in Computer Science

University of California San Diego, 2020

Professor Ryan Kastner, Chair

Mangrove forests are rich ecosystems that support our planet and the mankind in many unique ways. Unfortunately, these mangroves are declining at a rapid rate due to deforestation and other activities of the mankind. Monitoring and tracking of these mangrove trees is essential for their conservation. Machine Learning can be used for this purpose but to take advantage of the power of machine learning, image data needs to be captured for these mangrove ecosystems. This data collection is done using Unmanned Aerial Vehicles like drones in this research. Manually labelling the acquired image data for machine learning applications is a tedious and time-consuming task. This called for development of architectures which could learn from limited labelled data and take advantage of the large amounts of unlabelled data. Such architectures are

the semi-supervised semantic segmentation architectures and are studied in this research. We have shown how different semi-supervised models like the self-learning based Pseudo-Labelling architecture, the graph-based label propagation architecture and the deep leaning UNet-Autoencoder architecture perform on the task of mangrove segmentation in the aerial imagery. In order to evaluate different models we mainly look at the Intersection over Union because of it's popularity in segmentation tasks. Overall, we see that the deep learning UNet-Autoencoder architecture performs the best with an average IoU of 0.78. Conceivably, the performance of each of the models improves as more labelled data is provided for training. The highest IoU obtained in this research is 0.9 with the UNet-Autoencoder when as much as 75% of the data provided is labelled.

Chapter 1

Introduction

1.1 Background

Mangroves are trees and shrubs that predominantly grow in inter-tidal regions along subtropical coastlines. These are extremely valuable species to humankind and to the ecosystems they exist in. Despite their importance, a dominant loss of these mangrove systems has been recorded to an extent of as large as 50% [3]. These losses are largely attributed to the direct or indirect interference of mankind. Over the past decade scientists and researchers have noticed the alarming decline of these species, thus pushing the research community to take steps to help in their protection.

Technological advancements in the fields of Unmanned Aerial Systems(UAV) have helped with the above cause. These UAVs or drones are used to capture very high resolution images of locations from an aerial view . Researchers study this captured imagery in order to identify the presence/absence of mangrove species for their conservation.

Various methods have been employed to identify the mangrove species. Some researchers use manual identification for this. However due to the time consuming and labor intensive nature of the manual identification, this process is rendered quite inefficient. Thus a lot of work is being

done to automate this process. Various machine learning techniques are sought after for this purpose. However the performance and capability of these models is contingent on the availability of labelled training data.Obtaining completely labeled images is challenging and time consuming while unlabelled imagery is relatively easy to acquire. Thus instead of using supervised machine learning algorithms, researchers are looking into semi-supervised algorithms where the model could be trained using the labelled and the unlabelled images.

In this research we show how different semi-supervised models like self-learning based pseudo-labelling, graph based label propagation, deep learning based UNET-Autoencoders perform on the drone captured imagery. We observe that the deep learning architectures utilizing the UNET and the autoencoders perform the best based on different experimentations.

1.2 Conservation of Mangroves

Mangroves provide a lot of value to the ecosystem. Mangroves can support a complete ecosystem that is a conglomeration of several species of flora, fauna and biotic features in an area and their interaction with each other. Due to their sturdy roots, they form a natural barrier against storms, hurricane winds, waves, and floods. Mangroves also help prevent erosion by stabilizing sediments with their tangled root systems. They maintain water quality and clarity, filtering pollutants and trapping sediments originating from land. Mangroves play a major role in carbon sequestration. Thus clearly, mangroves are very crucial to our ecosystem and we need to conserve them.

Recent technological advancements have helped the ecologists by giving them the proper tools to detect habitat loss. Identifying the species will lead to their conservation in the long run. The technological advancements referred to here mainly refer to the machine learning algorithms used for classification and segmentation tasks. Using these tools the exhausting task of identifying these mangrove species can be automated. Researchers from different fields of engineering and

ecology are coming together to make use of these developed tools to help conserve mangrove species.

1.3 Image Segmentation

Image segmentation is a commonly used technique in computer vision. It is the process of partitioning an image into different regions-based on some criteria where the regions are meaningful and disjoint [4]. The goal in many tasks is for these regions to represent meaningful areas of the image, such as the crops, urban areas, and forests of a satellite image. In other analysis tasks, the regions might be sets of pixels grouped into such structures as line segments and circular arc segments in images of 3D industrial objects. This partitioning is performed often based on the individual characteristics of the pixels of the image and their relative orientation to each other.

1.3.1 Types of Image Segmentation

Image Segmentation can be categorized into Region based segmentation, clustering segmentation and edge detection segmentation[3]. Over the years various image segmentation algorithms have been developed and the most recent work in this field involves the use of deep neural networks.

Region based segmentation separates the objects into different regions based on some threshold value(s). Edge detection segmentation makes use of discontinuous local features of an image to detect edges and hence define a boundary of the object. Clustering segmentation divides the pixels of the image into homogeneous clusters. Recent works involve the use of artificial neural networks for the process of segmentation. In this research we have experimented with clustering based segmentation approaches, region based segmentation approaches and deep learning architectures for our application of mangrove detection. Although, after experimentation

it is seen that clustering does not give satisfactory results, but the approach based on deep learning gives the best results.

1.4 Semi-Supervised Learning

Historically there have been two different types of tasks in machine learning. The first task is *unsupervised learning*. In this, the machine learning models are trained on a sample of data points on their own. For example, if X represents a set of a total of n data points where $X = x_1, x_2 \dots x_n$, the machine learning model is trained on all $x_i \in X$. Typically it is assumed that the points are drawn i.i.d. (independently and identically distributed) from a common distribution on X. Fundamentally unsupervised learning is used to find an underlying structure of the data distribution like the density distribution which could have led up to the formation of X. There also exist other forms of unsupervised learning like clustering, outlier detection and dimensionality reduction [5]. In an *supervised learning* setting each point in X exists with a corresponding label y, for example,Y will represent a set of n corresponding labels where $Y = y_1, y_2 \dots y_n$ and the machine learning model is trained on both X and Y. Even in this setting, it is a basic requirement to have (x_i, y_i) samples drawn i.i.d from some distribution taken from $X \times Y$. The goal of these algorithms is well defined as the training itself is performed on some point x_i with it's mapping y_i [6]. Hence it is used in applications where some sort of mapping is required like classification or regression. However, with time the amount of data unlabelled data increased for applications like information retrieval, image processing, bioinformatics and geosensing [5]. If we want to use supervised learning (which is a more powerful tool), we would have to get labels for this abundant unmapped data. Since generating labels for all this data by hand is quite inefficient, researchers began to work on algorithms where we could make use of both labelled and unlabelled data.

Semi-supervised learning is a machine learning paradigm concerned with the study of how computers learn in the presence of both labeled and unlabeled data . In this case the data can be

divided into two parts, one of which is the labelled data : $X_l = x_1, x_2 \ldots x_n$ with its corresponding mapping $Y_l = y_1, y_2 \ldots y_n$ and the other division consists of the unlabelled part of the data which can be represented as $X_u = x_{n+1}, x_{n+2} \ldots x_m$. The machine learning model is trained by making use of both these parts of the data.

Most of the semi-supervised models have few underlying assumptions. Common assumptions include the smoothness assumption and the related low density assumption [5]. By the smoothness assumption if two points x_i and x_j are close in a high density region then their mappings y_i and y_j should also be close. The second common assumption in a semi-supervised learning setting is the cluster assumption. This assumption suggests that if the points in the data form clusters for separate classes then the unlabelled samples could be used to confidently place a good decision boundary in the low density region.

1.4.1 Semi-Supervised Learning in Image Segmentation

The growth of Semi-Supervised Learning in image processing has grown a lot over the years. Semi-Supervised Learning has already been proven to be a powerful paradigm for leveraging unlabeled data to mitigate the reliance on large labelled datasets [7] in Image classification techniques. This research aims to implement semi-supervised models for image segmentation of drone captured images of mangrove sites. In our research the meaning of using semi-supervised approaches for image segmentation means the use of both labelled and unlabelled images captured by a drone to train a machine learning model. The outcome of these models is a decision boundary formed at the edge of mangroves that can be efficiently and confidently produced based on the numerous unlabelled and the limited data samples present. The figure 1.1 shows examples of labelled and unlabelled data for our application of mangrove detection. More detailed information about the actual data is available under later chapters.

Figure 1.1: Example of unlabelled and labelled orthomosaic. The unlabelled orthomosaic is labelled in QGIS with polygons marking the boundaries of the mangroves.

1.5 Deep Learning

Deep learning is yet another concept in the field of machine learning where the model is inspired by the structure of a human brain. Such an architecture is in general called an artificial neural network. Deep learning is getting a lot of attention lately and for good reason. It's achieving results that were not possible before. These deep learning models are giving state of the art performances and thus there is a boom in the use of this technology.

A neural network comprises multiple layers depending on the application basis. Generally, the higher the number of layers in a network the higher the complexity of the model thus resulting in higher learning capacity. Based on the physiology of the human brain, the functional units in each layer are called neurons and they have weighted connections to the neurons of the adjacent layer. The first and the last layers of these networks are the input and output layers and all the layers in between are the hidden layers. The figure 1.2 shows an example of a *fully connected neural network*. A fully connected network means that all neurons of one layer are connected to all other neurons of the adjacent layer. We are going to be dealing with such fully connected networks in our research.

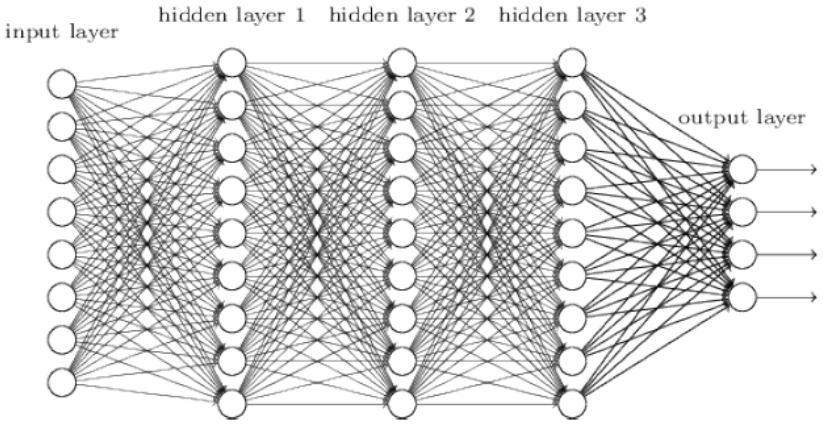


Figure 1.2: Example structure of a fully connected neural network with 3 hidden layers.All nodes of any of the hidden layers are connected to all the nodes of the adjacent layers as in a fully connected network.

In order to train these deep neural networks we give them samples of inputs and outputs represented as numerical features. Using this data they learn a relationship among the input and output variables which can further be extrapolated to other inputs. The complexity of the relationship learnt depends on the number of hidden layers. In order to begin training a network, we need to assign certain initial weights of the connections between all the layers. This initialization is either random or using some existing techniques [8]. We use the concept of transfer learning in our research to initialize the weights for the UNet architecture.

After initialization, the first layer of neurons is fed with (x,y) samples for training the model. Each neuron just outputs a linear combination of all the inputs it receives from the neurons of the previous layer. In order to learn nonlinear relationships among the input-output samples, a nonlinear function is applied to the linear combination found. This nonlinear function is called the *activation function*. There exist a number of activation functions for example, sigmoid activation function. This outputs the values in the range of 0-1 normally used to get probabilities of a certain output. The final goal of the neural network is to minimize some kind of loss i.e. the predicted output of the output layers should be as close to the actual output as possible. In order to achieve this, the weights of all layers are updated continuously for several iterations until the loss reaches

a certain acceptable level. The weights are updated in the gradient where the loss keeps reducing constantly. The rate at which the weight updates also depends on a constant called *learning rate*. This constant can be adjusted to increase or decrease the rate of change of the weights. An optimal learning rate is essential as a high learning rate can lead to a model never converging or a low learning rate can lead to a model taking a very long time to converge. If such a model is trained properly, it has high capability of generalization to unseen data.

Various deep learning architectures have been recently developed for the applications of image segmentation. Since we are looking at semi-supervised learning, we will be using certain concepts in deep learning which can help take advantage of the unlabelled data available along with traditional deep learning architectures. The subsection below explains the concept of transfer learning which is used for the UNET-Autoencoder architecture applied in this research.

1.5.1 Transfer Learning

Transfer learning refers to the process of transferring knowledge from a machine learning task to another related task. While most machine learning algorithms are designed to address single independent tasks, the development of algorithms that facilitate transfer learning is a topic of ongoing interest in the machine-learning community [9].In our work, we train auto-encoders using the unlabelled data and then use the weights learnt by this model for pre-training another deep architecture trained on the labelled samples. This is seen to give the best performance of all the methods tried for the application of semi-supervised image segmentation.

1.5.2 Fully Convolutional Networks

The UNet architecture we have used in our deep learning model is a fully convolutional network. It is built only from locally connected layers, such as convolution, pooling and upsampling. These layers are explained in the following paragraphs.

Convolutions

In our deep learning architecture we make use of autoencoders and a fully convolutional network called UNet. Thus the concept of convolutions becomes important. A convolution is an application of a filter to some input like an image. This results in an activation. Multiple application of filters on one input results in a map of activations called a feature map. The autoencoders used in the UNet-Autoencoder generates such a feature map by applying multiple convolutions. The figure 1.3 shows this process figuratively.

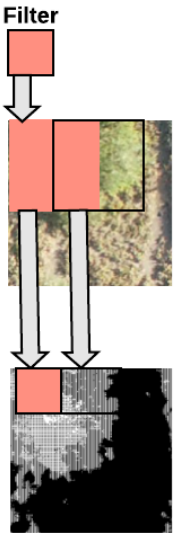


Figure 1.3: Example of a filter applied to a mangrove tile. Multiple applications of such a filter creates a feature map.

Pooling layer

The convolution layer generates a feature map for the input. But the feature map by itself is sensitive to the location of the features in the input. One way to deal with this is to downsample the feature maps which make it robust to the changes in the position of the pixels in the feature map. This is called *local translation invariance*.

In order to do this, pooling layers are used as they can summarize the presence of features

in patches of the feature map. There are two common pooling layers which are usually used. These are called *average pooling* and *max pooling*. We use *max pooling* in our architecture which calculates the maximum value for each patch in the feature map and thus summarizes all the pixels in the feature map.

Upsampling

In segmentation we want our output and input to be of the same size but the pooling layer does not take care of that. The upsampling part of a fully convolutional network creates a pixel wise dense feature map. The output of the upsampling layer will be enlarged yet sparse feature maps. The spare maps are converted into dense mappings using deconvolutions. The following deconvolution layers densify the feature maps through convolution-like operations with multiple filters.

Chapter 2

Data

2.1 Obtaining the data

Mangrove forests are divided between the sea and the land in the tropical and subtropical regions of the world between approximately 30°N and 30°latitude [10]. The mangrove ecosystems used for this research are located in La Paz, in Baja California. The mangroves in this research are mostly dwarf mangroves with heights below 5-10 meters. Such mangroves are found in coastal areas and estuaries. The study site is arid with temperatures above 35°C.

Large geographic areas are called regions in our study, like the region under consideration for the machine learning models for this research is La Paz. Each region is captured for about 6-10 mangrove forests and each of these are called sites. Each site is imaged multiple times (flights) at different altitudes and possibly with different drone or camera settings. The actual image acquisition is done using a DJI Phantom 4 Pro unmanned aerial vehicle (UAV) with its on-board camera. UAVs are proven to be very effective in remote sensing. In [11] the authors have shown the effectiveness of using UAVs for mangrove classification using object based classification approach.

The original raw images had a 4K resolution with the pixel dimensions as 3840 x 2160.

These images were captured at heights of 10m and 120m above the highest point of the mangrove canopy. The 10m data set had obviously a better resolution than the 120m data, an example is shown in figure 2.1. Due to the clearer image obtained at 10m above the canopy, these 10m datasets were used to train the volunteers who were responsible for manually labelling the mangrove images. These volunteers then hand-labelled the 120m imagery in order to get it ready for training the machine learning models. Even though the 10m datasets had a better resolution, we used the 120m datasets because capturing the 120m data in the UAV flights was more efficient as it is considerably faster. To ensure quality orthorectification and sufficient coverage, images were captured using a lawn-mower pattern at an overlap rate of 85% on all sides, as illustrated in figure 2.2 [1]. The reader can find additional information about the flight procedures in [12].

Figure 2.1: The resolution of 10m (left) and 120m (right) data imagery. The example is taken from [1]. The images captured at 10m naturally give higher resolution data than the images captured at 120m.

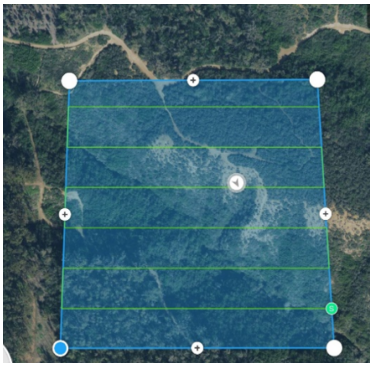


Figure 2.2: Mangrove images captured using a lawn-mower pattern at an overlap rate of 85% on all sides. The image shows one capture of the mangrove ecosystem area. Multiple such captures with an overlapping pattern are taken.

Before delivering the captured raw image as an orthomosaic, some calibrations are performed on the imagery captured by the UAV. The calibrations are different for RGB and multispectral images. The color calibrations are done in order to reduce the impact of natural factors like changes in weather, time of day, and season on the images captured. Without these calibrations the machine learning algorithms trained using these images could be negatively impacted. This calibration is performed using Adobe Lightroom. Once the individual images have been calibrated , they are imported to MetaShape to process them and generate orthomosaics and other image products [12].

2.2 Orthomosaics and Shapefiles

An orthophoto is an aerial photograph or satellite imagery which is geometrically corrected or orthorectified such that the scale of the picture is uniform. Such an aerial photo can be used to measure true distances as it is an accurate representation of the earth's surface. An orthomosaic is a raster image (dot matrix data structure) formed by merging many such orthophotos. After calibration of the UAV mangrove imagery the raw images are exported as orthomosaics by the mangrove monitoring team. They do this using Agisoft Photoscan 1.4.2, which is a photogrammetric software that can be used to generate images for GIS or other imaging applications. These orthomosaics are an accurate representation of an area which in our case is the mangrove ecosystem. This is created by stitching many images together and orthorectifying them. The figure 2.3 shows a drone captured image and a well stitched orthomosaic obtained after orthorectification.

The orthorectified orthomosaic thus far obtained is unlabelled and needs to be labelled(or at least partially labelled for the purposes of semi-supervised learning) if we want to use it with machine learning algorithms. We use QGIS software for this purpose. QGIS is an open-source cross-platform desktop geographic information system application that supports viewing,

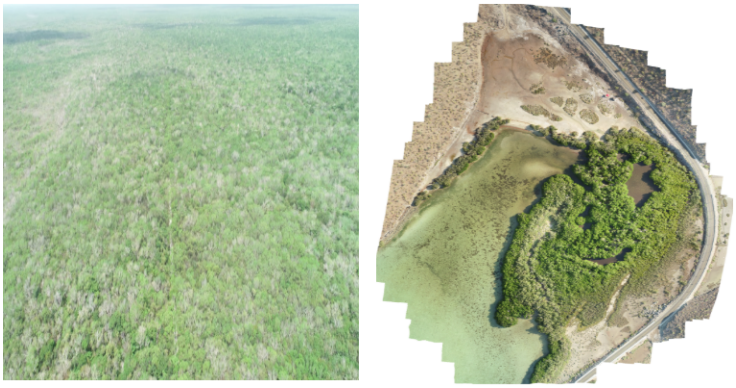


Figure 2.3: The figure on the left shows an image captured by a UAV(drone) and the image on the right shows a compiled and orthorectified orthomosaic. The image on the left is the orthomosaic for one of the sites in La Paz.

editing, and analysis of geospatial data. The unlabelled orthomosaic is labelled in QGIS with polygons marking the boundaries of the mangroves.The labels are stored as shapefiles which is a geospatial vector data format for geographic information system (GIS) software. It is developed and regulated by Esri as a mostly open specification for data interoperability among Esri and other GIS software products [13]. The shapefile format can spatially describe vector features: points, lines, and polygons. Each shapefile usually has attributes that describe it, such as labels for all pixels in the orthomosaic representing whether it's a mangrove or non-mangrove. The figure 2.4 shows the unlabelled orthomosaic and the labelled shapefile used in this research. QGIS is also used to fix geometries of the shapefiles before it can be used as data for our semi-supervised model.

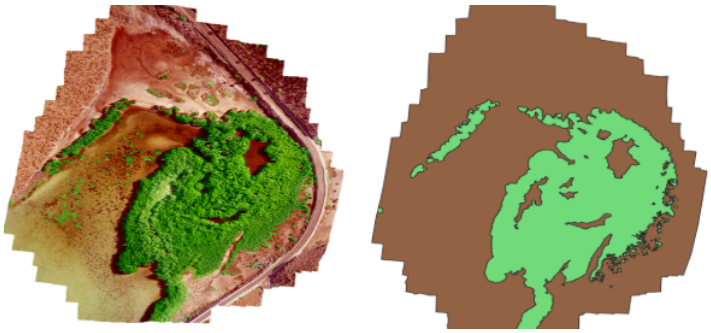


Figure 2.4: Figures showing the unlabelled orthomosaic (left) and labelled shapefile (right) for the corresponding orthomosaic. The shapefiles are visualized using QGIS.

2.3 Pre-processing of the data

Now that we have the orthomosaics and the labelled shapefiles, we can start with pre-processing them in order to feed them to our machine learning algorithms. Each orthomosaic is pretty huge and has data in the range of 1-4GB. Using these images by themselves for training any machine learning model is impractical and would crash the model due to memory issues. Therefore we choose to break down the large orthomosaic into smaller tiles of 256×256 pixels each and then perform coarse segmentation of the image by classifying the tiles instead of pixels as was introduced in [14]. The larger image files are broken down into tiles using a polygon file(shapefile) for the orthomosaic under consideration. The figure 2.5 shows a part of the orthomosaic and it's broken down tiles. After retiling the original orthomosaic into tiles, all the remaining tiles with dimensions less than 256×256 are removed.

Figure 2.5: Figures showing the retiling process. Image on the left shows a part of the entire orthomosaic and the images on the left shows the tiles of the corresponding part of the orthomosaic. The tiles shown here are 4096×4096 and the tile size used to train the model is 256×256.

Each tile is an image containing 4 bands including the RGB values and an alpha band for representing the transparency values. An alpha value of 255 suggests a no-data pixel. We only train our models on the RGB bands of these tiles. The broken down tiles are read using opencv and GDAL library in python. These images are read in as numpy arrays. After reading them in as numpy arrays they are reshaped from $256 \times 256 \times 4$ to $256 \times 256 \times 3$ and also normalized to have pixel values in the range of 0 and 1. The purpose of this normalization is to scale the numerical values in the image array to a common range, as the actual pixel values of a RGB image can

range from anywhere between 0 and 255.

The shapefiles for the labels of the orthomosaic are also broken down into tiles. These tiles map to the tiles formed from the original raster file. These tiles are stored in a jpg or tiff format. The figure 2.6 shows examples of the raster tiles along with it's annotation tiles . The labels are read from these tiles as 0s and 1s representing mangrove and non-mangrove regions respectively.

Figure 2.6: Samples of the orthomosaic tiles and their corresponding annotations.

Finally the dataset for our machine learning model is all the $256 \times 256 \times 3$ images with their respective annotations. This entire data for the La Paz site consists of a total of 2316 tiles. This dataset is initially divided into training and validation sets. This is done in order to validate the machine learning architectures and observe their improvement over several iterations. In order to mimic a semi-supervised implementation we also need some portion of the data to be unlabelled. We have used different amounts of unlabeled data for training and testing the performance of the models. The performance is observed as the amount of labelled data increases. The table 2.1 shows the different amounts of data used in training and validation. The models are finally tested on the entire site data as the test set. The results of these models are segmented tiles of size $256 \times 256 \times 1$. All these tiles are restitched back together to form a segmented visualization of the La Paz site.

Table 2.1: Table showing different data distributions used in the training and validation of the machine learning models.

Data sets	10% Labelled data	25% Labelled data	50% Labelled data	75% Labelled data
Labelled Training set	186, 256, 256, 3	464, 256, 256, 3	829, 256, 256, 3	1060, 256, 256, 3
Labelled Validation set	463, 256, 256, 3	463, 256, 256, 3	463, 256, 256, 3	463, 256, 256, 3
Unlabelled set	1852, 256, 256, 3	1852, 256, 256, 3	1852, 256, 256, 3	1852, 256, 256, 3

Chapter 3

Models

3.1 Clustering based segmentation

When none or 0% of the labelled data is given, the problem becomes an unsupervised machine learning problem. For experimentation purposes, we tried using few of the clustering based algorithms in order to segment the images of the mangrove ecosystems captured by the drones. In this approach we are treating the problem to be a completely unsupervised problem.

Clustering is the task of dividing the given data sample into a number of groups, such that data points in the same groups are more similar to other data points in that same group than those in other groups. These groups are known as clusters.This is one of the preliminary approaches we tried for the semi supervised classification of mangrove ecosystem UAV imagery. A cluster-then-label method as proposed by [15] could potentially be used to identify high-density regions in the UAV imagery. The knowledge from these clusters would then be used to help a supervised SVM in finding the decision boundary and classifying pixels into mangroves and non-mangroves. Before we implemented this approach we looked at our data distribution using unsupervised clustering. We fed our tiles to a clustering algorithm and visualized how well the model performed by looking at the clusters of mangroves and non-mangroves.

There are various types of existing clustering algorithms including connectivity models, centroid models, distribution models and density models. In our research, we have successfully experimented with a centroid model - batched k-means and distribution model - gaussian mixture models for clustering the pixels in the aerial imagery.

Batched k-means is a centroid clustering algorithm. The batched K-means work on a similar concept of k-means algorithm with the difference that the most computationally expensive step is conducted on a random sample of observations as opposed to all observations. This approach significantly reduces the time required to reach convergence. The k-means algorithm searches for a predetermined number of clusters within an unlabeled dataset. The algorithm works by finding centroids for clusters and then assigning the remaining points to their respective closest centroid. These centroids are randomly assigned initially. The k-means clustering works on Expectation Maximization(EM) algorithm for assigning pixels to clusters. The figure 3.1 shows the process of assigning points to clusters. All the orthomosaic tiles are fed to this clustering model and the resulting segmentations are visualized.

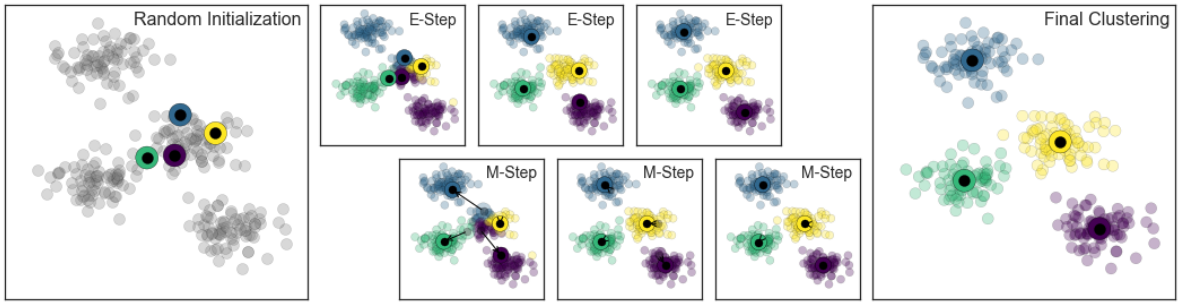


Figure 3.1: Expectation Maximization algorithm used in k-means clustering for assigning points to clusters. The expectation step assigns points to the nearest cluster center Maximization step sets the cluster centers to the mean. The figure is taken from [2].

One drawback of the batch k-means algorithm is a predetermined number of clusters need to be set and it does not account for variance in the data distribution. To overcome this we experiment with the Gaussian Mixture Model as it also accounts for variance along with updating the centroid in the EM process. This is a distribution clustering algorithm. It finds a Gaussian

probability distribution that best represents any input dataset. The results of the batch k-means and the gaussian mixture models is included in the results chapter. We observe that the clusters obtained by both these algorithms are not particularly good. The unsupervised clustering models are unable to differentiate between the mangrove pixels and the blocky regions of water present in the aerial images.

Overall, trying to learn from only unlabelled samples without taking into consideration some labelled samples in the training data gives poor results. It thus proved that using at least some amount of labelled samples along with the large amount of unlabelled data samples in our application of mangrove ecosystem image segmentation is essential. Thus we look into some semi-supervised machine learning algorithms for this and these are described in the next subsections.

3.2 Pseudo-Labelling

3.2.1 Self-Learning

Self-Learning is a concept in semi-supervised machine learning. It has been long used for semi-supervised learning [16] [17]. It is a simple and efficient semi-supervised way of learning from the available data [18]. Briefly, it is essentially a re-sampling technique that repeatedly labels unlabelled training samples and retrains itself based on the confidence scores of the pseudo-annotated labelled data. It uses the limited amount of labelled data for training some classifiers. After this initial training, this classifier is used to label all the unlabelled portions of the data. These are called pseudo-labels. The most confident predictions from among these pseudo-labels are used to retrain the initial classifier. The figure 3.2 shows an example of how the training dataset is reformed or expanded in this self-learning approach. After retraining, the model is ready to segment the mangrove and non-mangrove regions from the drone captured imagery.

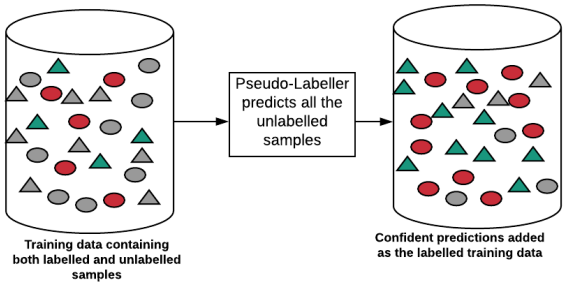


Figure 3.2: Expansion of the labelled training data in the process of self-learning. The labelled training set expands by adding the confident predictions of the pseudo-labeller to it.

In self-learning, the first step as described above is to spread the labels to all the unlabelled samples. This process is called label propagation. Label propagation is a semi automatic annotation process for labeling a large amount of unlabeled data. The objective of label propagation algorithms is to spread the labels from a small set of labeled data to a larger set of unlabeled data [19]. This can be done in multiple ways. We experimented with two ways to accomplish this - pseudo labelling using the self learning concept described above and graph based label propagation. We talk about graph based label propagation in the next section.

3.2.2 Applying self-learning

Using the concept of self-learning, we built a pseudo-labelling segmentation model which could take into account the information provided by the unlabelled data samples with the labelled samples. An architecture for this is shown in 3.3. The name *pseudo-labeller* comes due to the fact that a supervised classifier is used to label the unlabelled points and then these *pseudo-labels* are used to retrain the model.

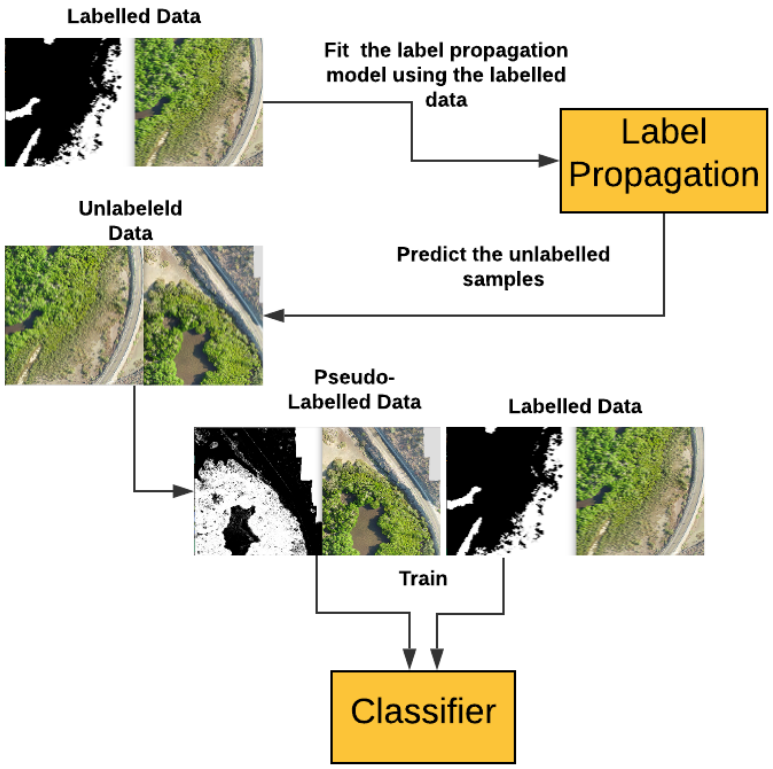


Figure 3.3: The model architecture of the pseudo-labelling classifier. The label propagation model can either be based on self-learning or graph based propagation of labels.

After forming tiles for the site image of the La Paz region, we segregate the labelled tiles and the unlabelled orthomosaic tiles. As can be seen by the architecture diagram, the labelled tiles are fed to a label propagation model, which in this case is a supervised machine learning classifier. This classifier is then used to predict labels of all the pixels of the unlabelled tiles. The train dataset is then expanded based on the confidence of the prediction. We add all predictions with $Pr(y = 1|x) > 0.7$ and $Pr(y = 0|x) > 0.7$ to our training data. This process is done without replacement, this means that the data points added to the labelled set are removed from the unlabelled samples. Our training data thus now also consists of a sample of the unlabelled data with their corresponding *pseudo-labels*.

This gives a classifier which classifies all pixels in an image being either a mangrove or

non-mangrove. This model can be run for multiple epochs where the classifier obtained at the end of each epoch is used to label the remaining unlabelled samples. All the confident samples from this set is then added on to the training labelled samples and the retraining continues. The number of epochs can be decided either by setting it to a constant value or when the performance of the model reaches stagnation. Of Course the number of epochs are limited by the number of unlabelled samples. If all the unlabelled samples are confidently predicted by the obtained classifier, the unlabelled set will become empty and the model does not have any more data to learn from.

We experimented with different machine learning classifiers along with the concept of self-learning in order to build our pseudo-labelling semi supervised segmentation models. These classifiers included Random Forests, MLP classifiers and SVM classifiers. Ultimately, we found that random forests give the best performance as compared to the other classifiers. They were found to be most efficient in terms of speed and storage. Therefore all experimentations with different amounts of labelled data were performed on the random forest classifiers used along with the concept of self-learning.

Random Forests

Random forest is an ensemble tree-based learning algorithm. "Tree-based" means it is a collection of multiple decision trees. Decision trees are algorithms which learn by splitting the dataset into smaller and smaller parts. Each "node" in the tree represents the condition (the condition on each of the features of a particular data point) and the "edge" represents the possible outcomes. The branching of data in a decision tree takes place until either the data cannot be split further or some preset rule is reached for example the maximum number of leaf nodes. The decision trees are generated using an attribute selection measure such as information gain, gain ratio, and Gini index for each attribute. An attribute selection measure is a heuristic for selecting the splitting criterion in a tree that correctly decides the partitioning of data which would give

the individual classes. These attribute selection measures are also known as splitting rules. The attribute that has the best score for the measure is chosen as the splitting attribute for the given data points. Different attribute selection measures are described below.

- Information gain : This measure provides the splitting attributed in terms of the information required to further describe the tree. It minimizes the information needed to classify the data into separate partitions and aims to reduce the randomness in these partitions. Mathematically this can be found as :

$$Info(D) = -\sum^{n} p_i \log_2 (p_i) \tag{3.1}$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \tag{3.2}$$

$$Gain(A) = Info(D) - Info_A(D) \tag{3.3}$$

  Where p_i is the probability that a data point in dataset D belongs to a class C_i where n is the total number of classes (which in our case is 2) and v is the total number of data samples created.

- Gain Ratio : The information gain measure is biased towards the attribute with a large number of values. This bias is dealt with using Gain ratio, which can be mathematically represented as :

$$Split_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|} \tag{3.4}$$

$$GainRatio(A) = \frac{Gain(A)}{Split(A)} \tag{3.5}$$

- Gini Index : This measure considers the binary split for each attribute and can mathematically be represented as follows,

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{3.6}$$

Random forests work by randomly sampling data into smaller parts and it then creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. The figure 3.4 shows this process diagrammatically. Random-forest does both row sampling and column sampling with decision trees as a base. The row sampling is also called bagging or bootstrap aggregation which is a process of reducing variance in the model without impacting bias. Since random forests create samples of data internally, it cross-validates the model by itself and there is no particular need to cross-validate the model separately. The performance metrics of these classifiers even without cross-validation is reliable and a correct measure of their performance.

Scikit Learn offers an ensemble package which has multiple ensemble classifiers including the random forest classifier. There are various hyper-parameters which can be tweaked in order to experiment with the model. The hyper-parameters we have used in our experimentation include the *n_estimators, max_features, max_depth*. The *n_estimators* is a hyper-parameter which is the total number of decision trees of random forest classifier generators before taking the majority vote or average of the results of the trees. A higher number of decision trees make the random forest more powerful and it gives a better performance but it also increases the complexity of the model. The *max_features* is the hyper-parameter which tells the model how many features to use while deciding to split a node. The *max_depth* hyper-parameter defines what is the maximum acceptable depth of each of the decision trees formed in the random forest. Different

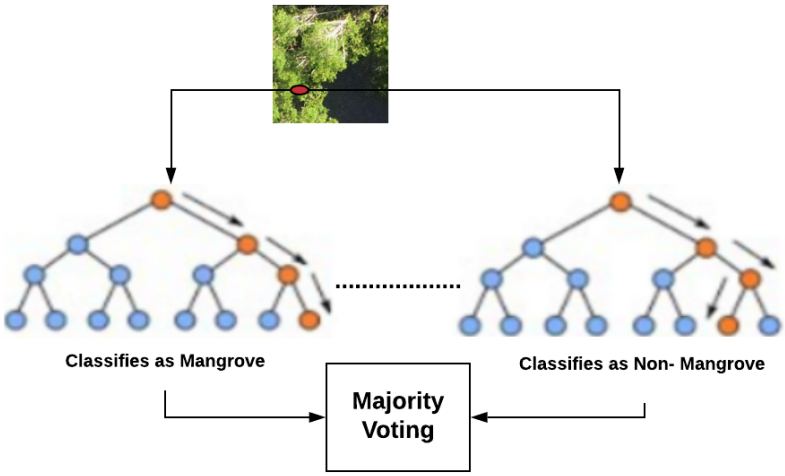


Figure 3.4: Random forests work by randomly sampling data into smaller parts and it then creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting

values of these hyper-parameters are experimented with and an optimal set of values is found for experimentation with different amounts of labelled data. This process in general is called hyper-parameter tuning.

3.3 Graph-Based Label Propagation

There are multiple methods to propagate labels to all the unlabelled data samples. One way is to use a classifier as we have described in the previous section and another way we experimented with is to use graph based approaches.

Graph based label propagation uses the raw similarity matrix constructed from the data with no modifications. The idea is to build a graph connecting similar data points and the label is propagated from the labelled points to the unlabeled points. The edges of this graph are encoded in the similarity matrix constructed from the data. The similarity between two data points can be found using one of the two kernels called the RBF and the KNN kernels. The kernels are chosen based on the scalability of the application.

The RBF kernel produces a fully connected graph which is stored as a dense matrix. The size of this dense matrix may be very large if the data size under consideration is very large. This leads to very long running times and the need of very large storage. The KNN kernel produces a sparse matrix which is much more memory friendly than a dense matrix. This leads to faster running times of the algorithm and less need of storage space. The mathematical representations of these kernels are shown in 3.7 and 3.8

$$\text{RBF} = \exp(-\gamma|x-y|^2) \quad \gamma > 0 \tag{3.7}$$

$$kNN = 1[x^{'} \in kNN(x)] \tag{3.8}$$

After experimentation we found that graph based label propagation does not work well when there is a lot of unlabelled data as it requires a lot of memory to build the graphs for label propagation. The number of nodes in our application would be equal to the number of unlabelled data pixels. For example, if 90% of the data is unlabelled that means around 10M data pixels are unlabelled and building a graph for these many nodes will clearly take up a lot of space. Since we are working on a machine with a limited of 25GB RAM, this process cannot be performed on a large number of points. However we have shown the performance of this model on a smaller scale of data. It is seen that there is slight improvement in the performance when compared to the pseudo-labelling label propagation method. The UNet Autoencoder algorithm described in the section ahead still performs better than the graph based label propagation.

3.4 UNet-Autoencoder

Due to the recent advancements in deep neural networks, we decided to leverage the power of deep architectures for our applications. UNets have been proven to be powerful tools for

segmentation tasks as they were specifically designed for biomedical image segmentation in the first place [20]. We experimented with UNet along with Autoencoders for the purpose of taking advantage of the unlabelled data for our application of mangrove ecosystem image segmentation.

3.4.1 Autoencoders

Autoencoders are neural networks which learn in unsupervised fashion. An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers. Universal applications of autoencoders involve image coloring, feature extraction, image denoising, dimensionality reduction etc. We will use convolutional layers in autoencoders for pre-training another fully-convolutional model for image segmentation. We only need the unlabelled orthomosaics to train autoencoders. It applies backpropagation and sets all the labels to the input features, for example if input is $X = x_1, x_2, ...x_i$ then the each target value is set to be the input as $y_i = x_i$. A general layout of an autoencoder is given in figure 3.5. In our application, we will be feeding the unlabelled tiles of the orthomosaic to the input layer of the autoencoder and the output layer is expected to generate a reconstructed image of the input tiles.

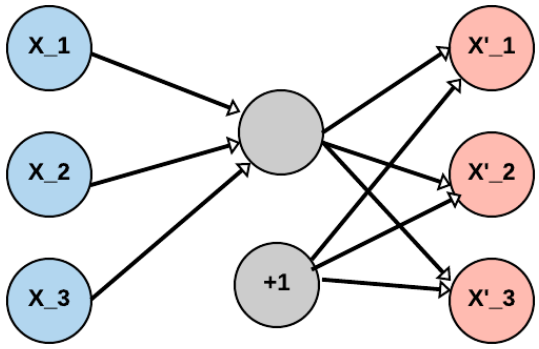


Figure 3.5: A general structure of an autoencoder. The input is any vector X and the output of the autoencoder is a reconstruction of the input X, represented in the image by $X^{'}$

The autoencoder as the one shown in figure 3.6 tries to learn a function, $h_{W,b} \approx x$ which outputs an image as similar to the one it is being provided with. The process of reconstructing an

image using an autoencoder might seem like a trivial task, but if we limit the number of hidden units in the autoencoder we can learn the interesting underlying structure of the input data. An autoencoder consists of 3 parts:

- Encoder. This part of the autoencoder represents the input into latent space. It compresses the input and passes on a distorted version of the input image to the next part i.e. code.
- Code. This part of the network represents the compressed input which is fed to the decoder.
- Decoder. This decompresses the image received from the code to the original dimensions. This is a lossy reconstruction of the input.

We give unlabelled orthomosaic tiles to the encoder as inputs and the outputs of the decoder are lossy reconstructions of the input tiles as is shown in figure 3.6. Although we do not make use of this reconstructed image further in the architecture, but we do make use of the weights learnt by this autoencoder model to pre-train our UNet.

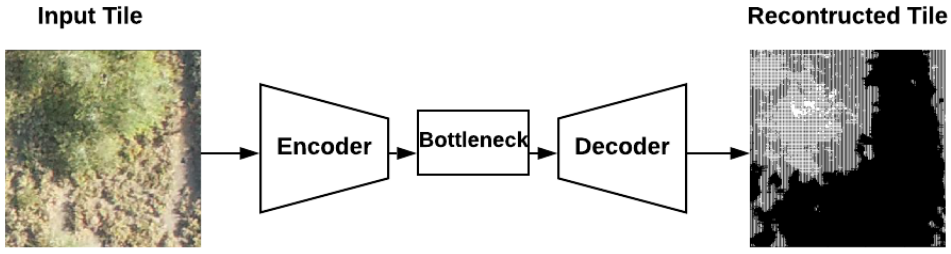


Figure 3.6: The autoencoder representation used in our application. The encoder unit comprises the convolutional layers for downsampling the input tile and the decoder unit comprises the deconvolutional layers for the upsampling of the compressed input image. The decoder outputs the reconstructed image.

We use convolutional autoencoders(CAE) in our research. Since we only use the autoencoder in order to transfer the knowledge extracted from the unlabelled samples, we do not need to generalize the autoencoder's learning to unseen samples as the training data is going to be

constant. Since there is no potential need for generalization a simple linear activation suffices and we do not need non-linear activation. The activation used in the autoencoder is also called identity activation and can mathematically be represented as in 3.9

$$f(x) = x \tag{3.9}$$

The encoder and decoder parts of a general autoencoder are represented by the convolutional and deconvolutional layers of the CAE respectively [21]. A convolution operation is performed between each input image having a depth D. If X represents the image $X = X_1, X_2 ..... X_D$ and a set of n convolutional filters like $F_1^1 ..... F_n^1$ to produce a set of n feature maps. The activation function $a()$ in our case is a linear activation function.

$$z_m = a(X \times F_m^1 + b_m^1) \quad , \quad m = 1 \dots n \tag{3.10}$$

Where X is the input image, F_m^1 is the m^{th} convolutional filter and b_m^1 indicates the bias of the m^{th} feature map. The reconstructed image $\hat{X}$ is the result of convolution between the dimension of the feature Z and the deconvolutional filter F^2

$$\hat{X} = a(Z \times F_m^2 + b^2) \tag{3.11}$$

. We use the mean squared error(MSE) as the loss function for our autoencoder, shown in 3.12

$$L(x, \hat{x}) = \frac{1}{2}(||X - \hat{X}||_2^2) \tag{3.12}$$

The autoencoder tries to minimize this loss with each epoch and the model is said to be converged when the loss is either constant or reaches a certain acceptable level. A better autoencoder will have a lesser loss.

3.4.2 U-NET Segmentation

UNET is a fully convolutional network approach which was first proposed by [20] with the goal of developing better segmentation models for biomedical images. U-Net is more powerful than conventional convolutional deep learning architectures, in terms of architecture and in terms of pixel-based image segmentation predicted by convolutional neural network layers. It is even effective with limited dataset images which is why it also suited our application where it is difficult to obtain annotated data. The input to this U-Net segmentation model is the labelled samples of tiles of the orthomosaic and the output is expected to be a segmented image of the tile.

The network architecture as shown in figure 3.7 consists of a contracting path and an expansive path, which gives it the u-shaped architecture. The contracting path is a typical convolutional network that consists of repeated application of convolutions. Each of these convolutions is followed by some activation function(ELU in our case) and a max pooling operation. During the contraction or down-sampling, the spatial information is reduced while feature information is increased. After this a sequence of up-convolutions are applied in order to combine the spatial information learnt in down-sampling along with the feature information by concatenations of high resolution features from the contracting path [20]. It follows an overlap-tile strategy as shown in figure 3.8 i.e. the segmentation map only contains the pixels for which the full context is available in the input image [22]. The border pixels are predicted by extrapolating the missing context by mirroring the input image. This makes tiling of a large input image essential for the application of U-Net because otherwise the resolution would be limited by the storage available.

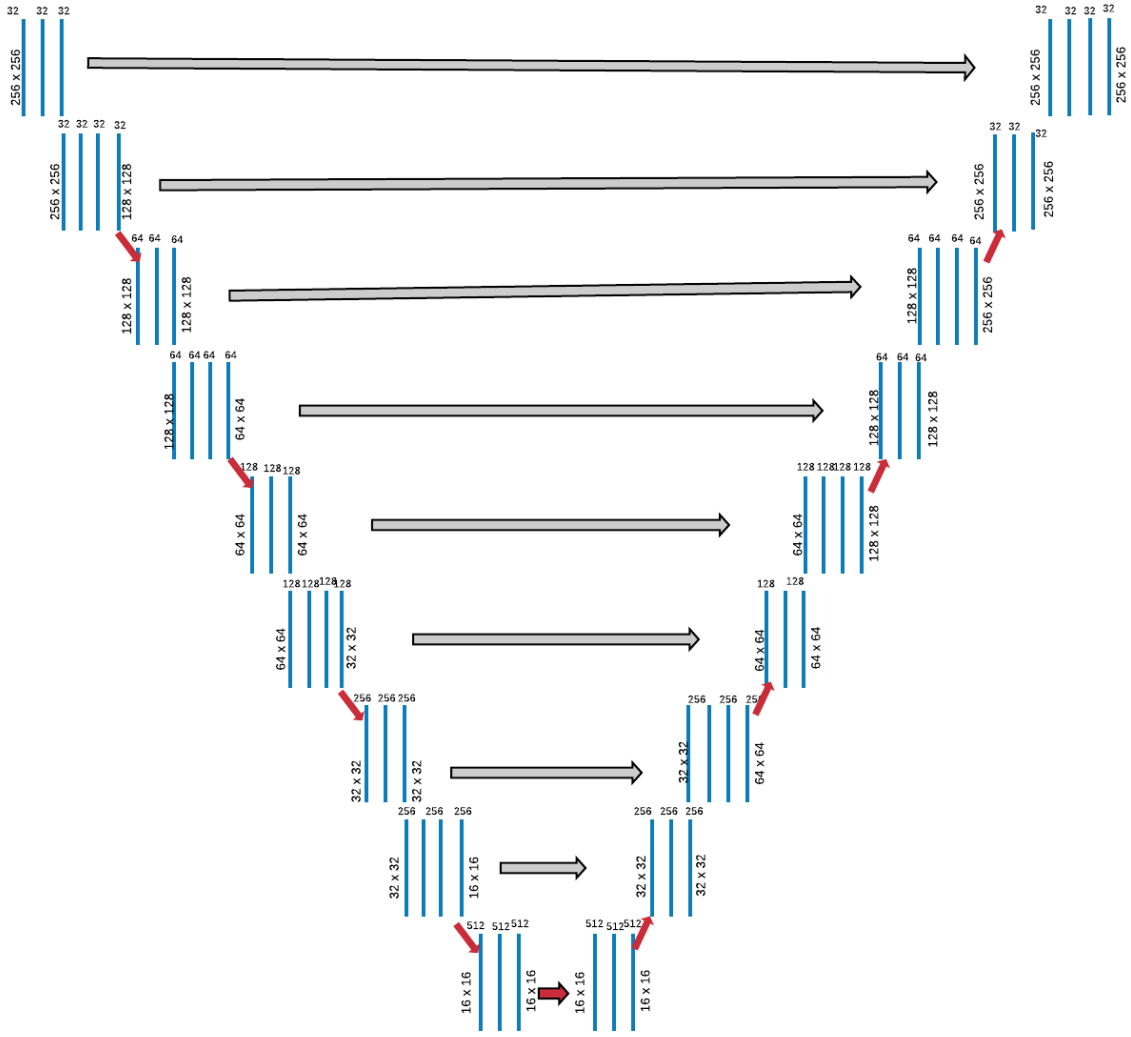


Figure 3.7: UNet architecture used in our application. The network architecture as shown consists of a contracting path and an expansive path, which gives it the u-shaped architecture.

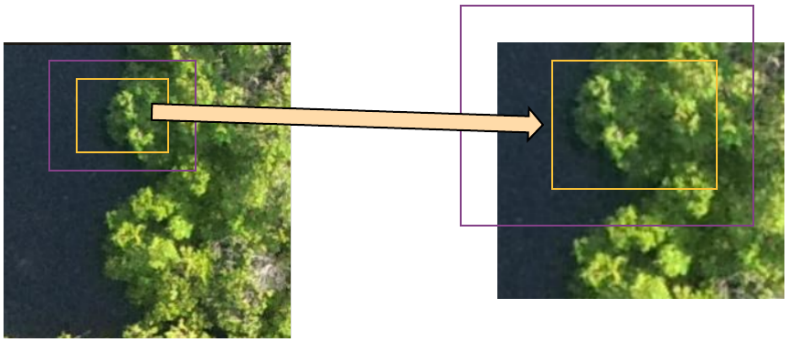


Figure 3.8: Overlapping tile strategy for segmentation of large images. The prediction of the pixels in the yellow area require the image data within the blue area as input.

The activation used on the convolutions is called the Exponential Linear Unit (ELU). This activation function has recently been found to converge the cost function to zero faster and also produce more accurate results [23]. It is different from other activation functions, as it has an extra alpha constant which should be a positive number. This activation function is basically used due to its capability to speed up learning [22] by decreasing the bias shift by pushing the mean activation towards zero. The ELU is given as :

$$elu(x) = \begin{cases} \alpha(\exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{3.13}$$

and it has the gradient

$$\frac{\partial elu(x)}{\partial x} = \begin{cases} elu(x) + \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{3.14}$$

The loss function used in our experimentation is the Binary Cross Entropy loss. It is also called Sigmoid Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss. The mathematical formulation of this loss function is given in 3.15

$$L = \sum_i y_i \log_{10} o_i + (1 - y_i) \log_{10} (1 - o_i) \tag{3.15}$$

The UNet tries to reduce this loss with each epoch and as in the autoencoder, the model is said to have converged once the loss either reaches stagnation or attains an acceptable value.

3.4.3 Semi-Supervised UNet-Autoencoder Architecture

In our application of semi-supervised semantic segmentation we leverage the concept of transfer learning with the autoencoder and the U-Net architectures described above. The weights learned by the autoencoder are saved and transferred onto a supervised U-Net model as a pre-training measure. In this way the U-Net model learns the data embeddings from the unlabelled data sample. After pre-training the model with the weights learnt by the autoencoder, it is trained on the limited amount of labelled data available. This yields a more powerful tool for segmentation than a simple baseline U-Net which is only trained on the small labelled sample without taking into account the unlabelled sample. The performance improvement is shown in the results chapter. The figure 3.9 shows the working of this semi-supervised architecture.

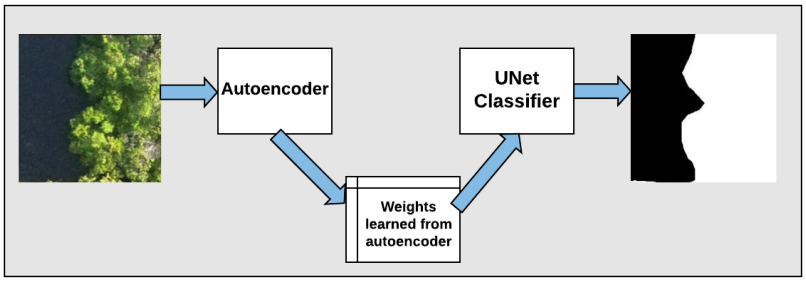


Figure 3.9: UNet Autoencoder workflow used for semi-supervised semantic segmentation. The weights learnt from the autoencoder are transferred to the UNet model. This transfer learning mechanism enables the model to learn from the unlabelled tiles.

Chapter 4

Results

4.1 Performance evaluation metrics

The performance of the semi-supervised segmentation model is evaluated by mainly looking at the Intersection over Union score (IoU). IoU, also known as the Jaccard index, is the most popular evaluation metric for tasks such as segmentation, object detection and tracking [24]. This is a metric to evaluate how similar a bounding box in a predicted image is to the ground truth bounding box. It works by finding a ratio of the area where the two boxes overlap to the total combined area of the two boxes which is mathematically shown in 4.1 . The figure 4.1 shows an example of this process.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

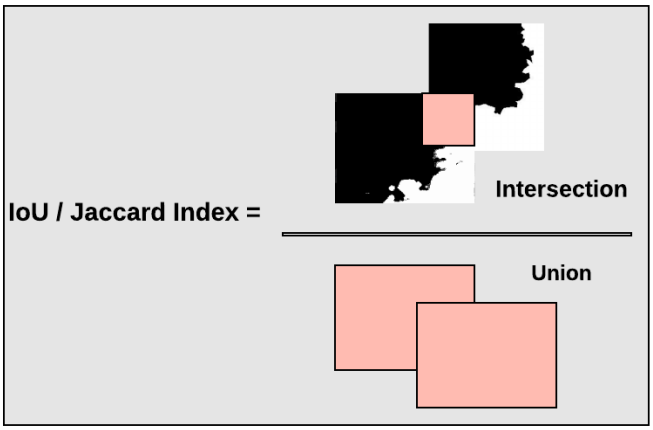
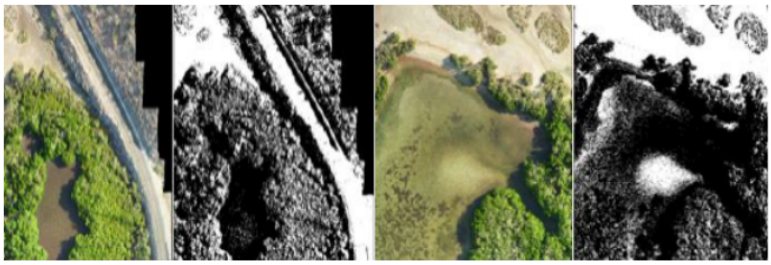


Figure 4.1: The intersection over Union. This metric finds the ratio of where the two vectors under consideration overlap to the total combined area of the two vectors.

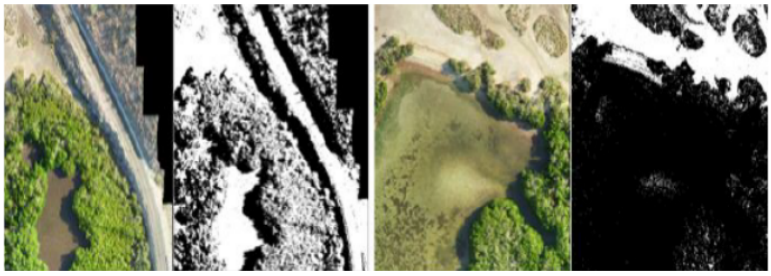
We also plot precision recall curves for the classification of the pixels obtained after training each of the models. This plot summarizes the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

4.2 Clustering based segmentation

We experimented with clustering based segmentation models. For this we clustered our data (the tiles generated from the mangrove ecosystem site) using various algorithms. The clustering of tiles using mini batch k means and gaussian mixture model is shown in figure 4.2. We can see that the results obtained using only the unlabelled data points are not that good. The pixels that are classified as non-mangrove need to be very different from the mangroves for this algorithm to work, which is not the case here. The images have non mangrove green areas which the algorithm misclassifies as mangroves. Thus, using these clusters to spread labels to the unlabelled pixels in our aerial imagery was not an option. Therefore we tried out other methods using the semi-supervised semantic segmentation.



a - Results of Mini Batch K-Means clustering



b - Results of Gaussian Mixture Model

Figure 4.2: Results of clustering sample tiles of the La Paz site. In figure *a* the mini batch k-means clustering algorithm is used while in figure *b* the gaussian mixture model is used.

Since the mangroves are densely clustered together so we also tried another clustering algorithm (DBSCAN) which we thought could have performed better as it is density based clustering. This is based on the idea that a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density. Since this clustering method is inefficient for large datasets due to it's recursive nature, we had speed issues implementing this. We were not able to successfully implement this.

4.3 Pseudo-Labelling

The Pseudo-Labelling semi-supervised segmentation model was trained on various amounts of labelled data. Some of the available labelled data was kept aside as a validation set and this was used for performance analysis of the model after each training session.

Before we began to test the model with different amounts of labelled data, we experimented with different hyper-parameters used in the random forest classifier used in the model. We did our hyper-parameter tuning on this using the 10% labelled data sample. For all other experimentation with different amounts of labelled data, the hyper-parameters obtained with this initial tuning are used. We tried out different values for the following hyper-parameters:

- Max_depth : [5,10]
- N_estimators : [5,10,15,20,25,30]
- Max_features : ['auto','sqrt']

Table 4.1: Best hyper-parameters obtained with the hyper-parameter tuning on 10% labelled data.

Max Depth	N Estimators	Max Features
10	20	auto

As the number of decision trees in a random forest architecture increases, it adds to the complexity of the model and thus the performance of the segmentation model is expected to increase with the increase in the number of trees. Of course this takes a toll on the run time complexity of the model, thus a *good enough* number for these trees needs to be selected. This number is selected based on the percent of improvement of the model as compared to a lesser number of trees. When the model stops performing significantly better, that particular number is selected for the parameter *n_estimators* The graph in figure 4.3 shows how the performance of the model varies with increasing number of decision trees in the random forest model. We can see that for 20 decision trees the model performs best in terms of performance and computation cost.

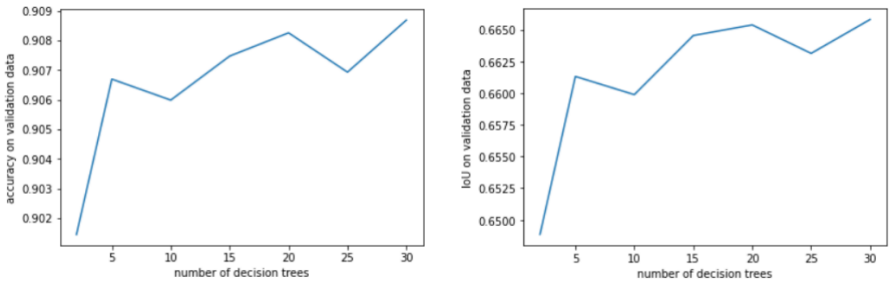


Figure 4.3: Performance of the Pseudo-Labelling semi supervised segmentation model with 10% labelled data. The figure on the left shows how the accuracy varies and the figure on the right shows how the IoU varies.

The precision recall curve of the final Pseudo-Labelling model is depicted in figure 4.4. This is plotted when the classifier is tested on the entire site with 10% labelled data available for training the model.

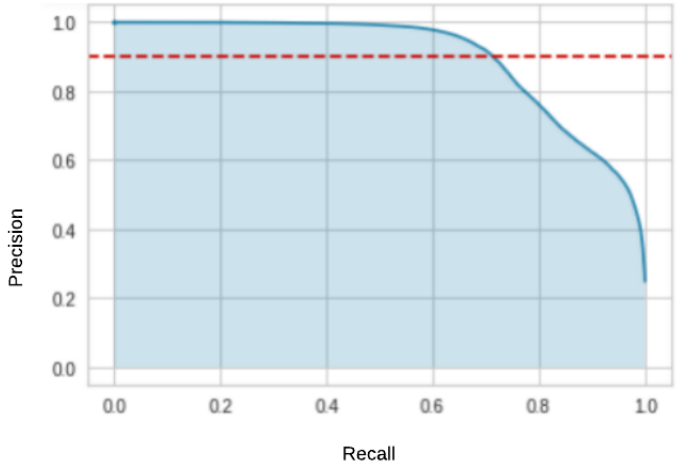


Figure 4.4: The precision recall curve for the pseudo-labelling model for 10% labelled data and tested on the entire site.

Once the hyper-parameters are chosen, the model is trained on the available limited data. This then classifies the unlabelled data and the confident predictions amongst these are used to

retrain the model. Once the final model is ready, this is used to segment all the tiles of the La Paz site. The figure 4.5 shows examples of tiles as segmented with the Pseudo-Labelling model when different amounts of labelled data are given to it.

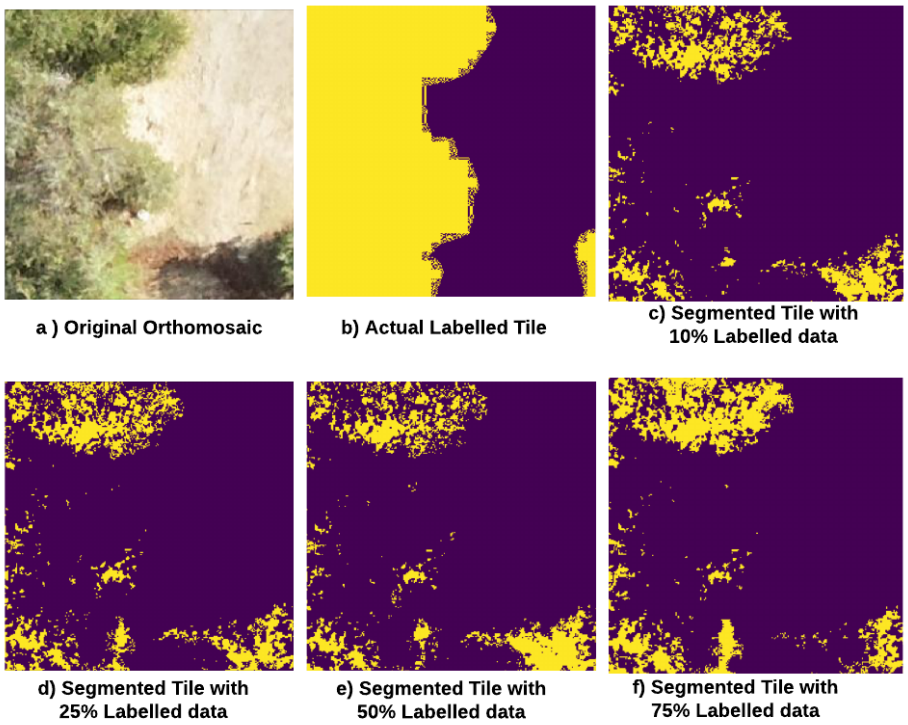


Figure 4.5: Sample segmented tiles by self learning based Pseudo-Labeller for different amounts of labelled data.

We mainly look at the IoU scores as a performance evaluation metric for the pseudo-labelling model. This is shown in table 4.2 . It can be seen that as the labelled data increases the performance of the model improves as expected. The IoU values range from 0.58 for as little as 10% labelled data to 0.66 IoU for as much as 75% labelled data.

Table 4.2: Performance of Pseudo-Labelling semi-supervised semantic segmentation model with different amounts of labelled data

Performance Metric	10% Labelled data	25% Labelled data	50% Labelled data	75% Labelled data
Accuracy	0.890	0.8965	0.8978	0.9058
IoU	0.5869	0.6195	0.62015	0.6612

4.4 Graph based Label Propagation

When we use graph label propagation instead of using a random forest classifier as a label propagator as in the previous section, we observe an improvement in the IoU score even when as little as 10% of the data is given. One drawback of this model is that it takes up to 3 times as long as the pseudo-labelling model to train and also it is very memory inefficient. Due to these restrictions we couldn't't train it on all the tiles available. But we made sure to keep the percentage of labelled data the same as other experiments so that the evaluations can be correctly compared with other models. The table 4.3 shows the improved performance of the graph based pseudo-labelling model on different percentages of labelled data. The figure 4.6 shows the precision recall of this model when 10% of labelled data is provided for training. Even an eye test on the segmented tiles given in figure 4.7 show that the graph based model performs better that our original pseudo-labeller. Even for as little as 10% labelled data the graph based label propagation model improves by 0.10 IoU i.e. for as little as 10% labelled data, the graph based model gives an IoU of 0.66 which the previous pseudo-labeller could only obtain when around 75% labelled data was provided.

Table 4.3: Performance of graph based semi-supervised segmentation model with different amounts of labelled data

Performance Metric	10% Labelled data	25% Labelled data	50% Labelled data	75% Labelled data
Accuracy	0.8451	0.8594	0.881	0.899
IoU	0.66170	0.65610	0.7390	0.7644

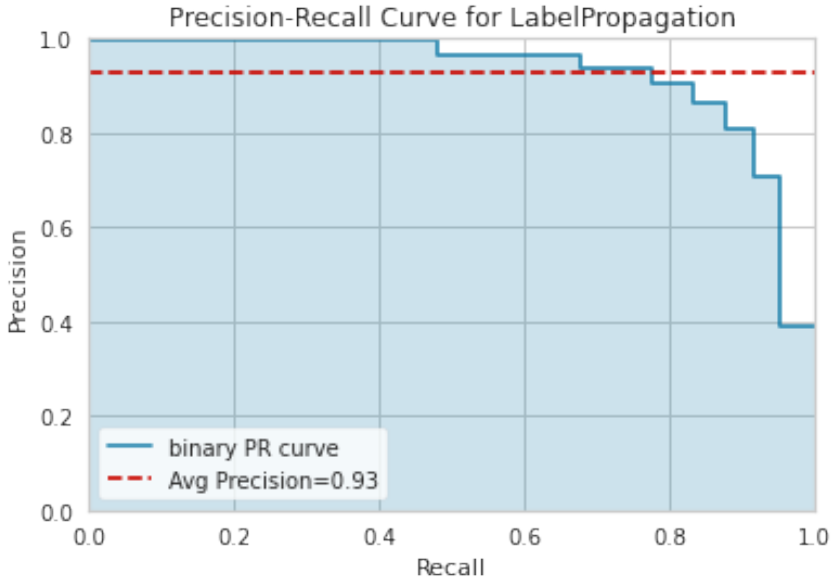


Figure 4.6: The precision recall curve for the graph based semi-supervised segmentation model for 10% labelled data and tested on the validation data.

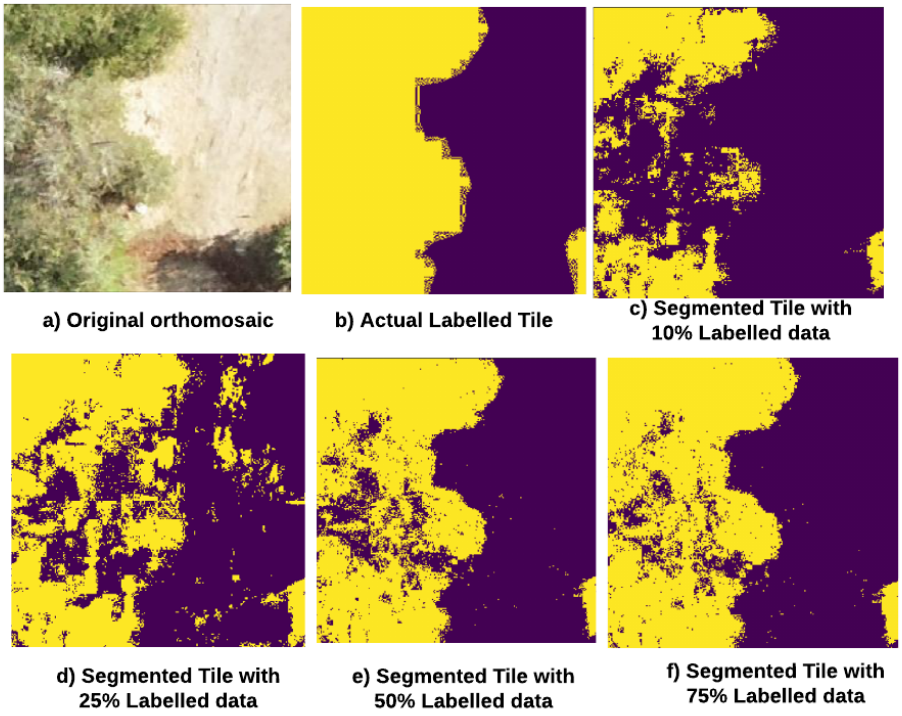


Figure 4.7: Sample segmented tiles by Graph based label propagation for different amounts of labelled data.

4.5 UNet-Autoencoder

UNets are believed to give state of the art performance on image segmentation tasks and so is also proved with our experiments with our UNet-Autoencoder model. The autoencoder part learns the patterns from the unlabelled data and this is used with a traditional UNet to build a semi-supervised semantic segmentation model. We have trained our autoencoder for 50 epochs and our pre trained UNet for 70 epochs. The figure 4.4 shows the performance of this model with different amounts of labelled data. It clearly performs better than both our previous models. Increasing the training time for this model is expected to further improve the IoU score for this model. When we train our autoencoder and the UNet for 150 and 170 epochs respectively, we

observe an IoU of *0.9* which is the highest obtained in our application. The graphs in figure 4.8 and figure 4.9 show how the model improves with each epoch.

Table 4.4: Performance of UNet-Autoencoder semi-supervised segmentation model with different amounts of labelled data

Performance Metric	10% Labelled data	25% Labelled data	50% Labelled data	75% Labelled data
Accuracy	0.954	0.9594	0 0.9658	0.9716
IoU	0.7254	0.737	0.806	0.8426

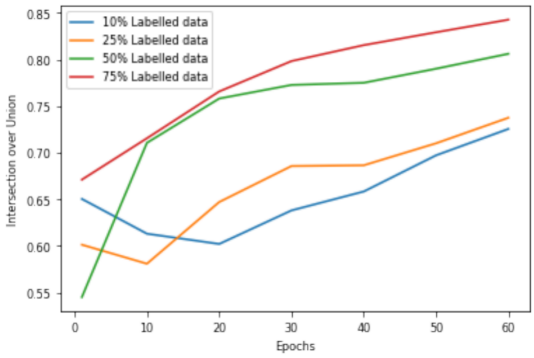


Figure 4.8: Graph showing the performance of the UNet-Autoencoder in terms of Intersection over Union score for different epochs (in multiples of 10)

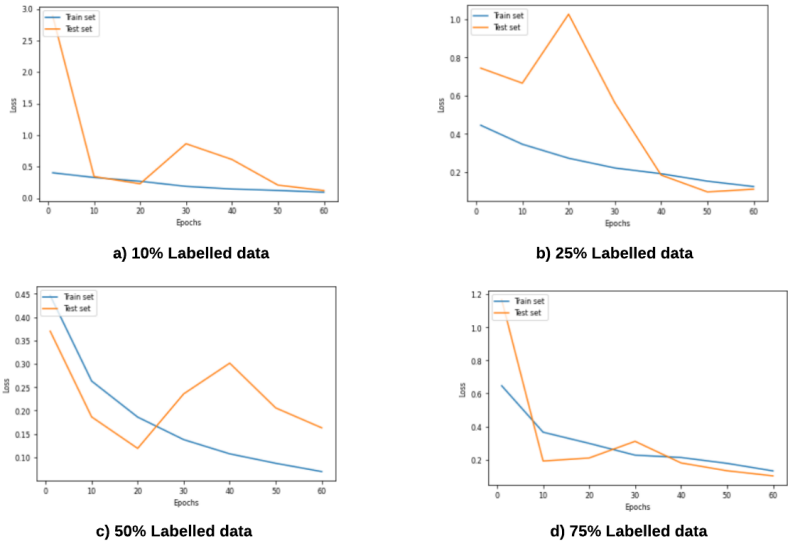


Figure 4.9: showing the loss of UNet-Autoencoder with increasing epoch

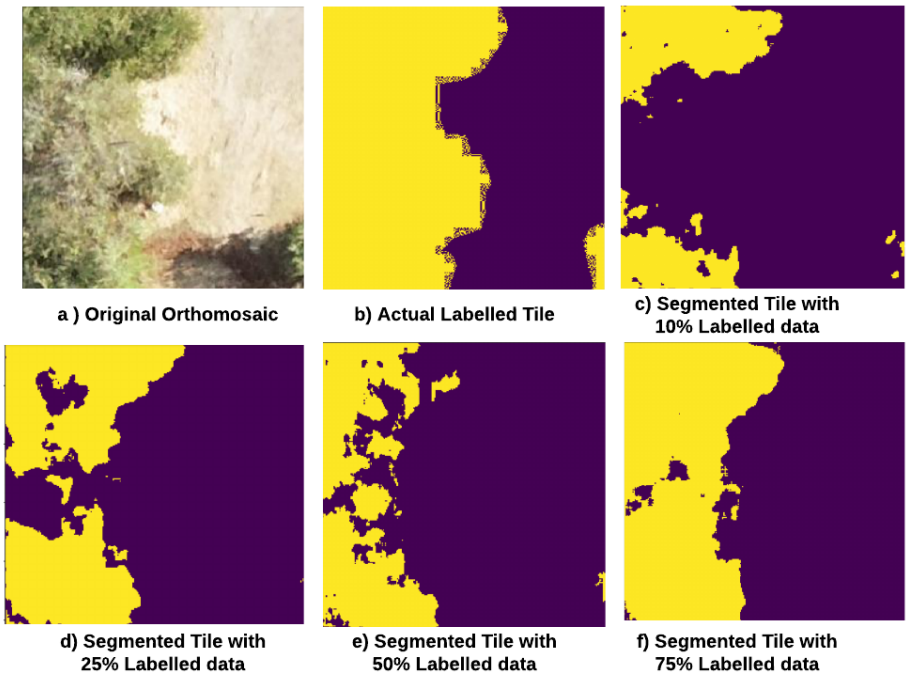


Figure 4.10: Sample segmented tiles by UNet-Autoencoder for different amounts of labelled data.

4.6 Comparing performance

We observe that the UNet-Autoencoder model performs the best among all our models with the scope of further improvement. The graph in figure 4.11 shows a performance comparison of all the models when different amounts of labelled data is given to each of the models. We can see a clear improvement in the performance of each of the models.

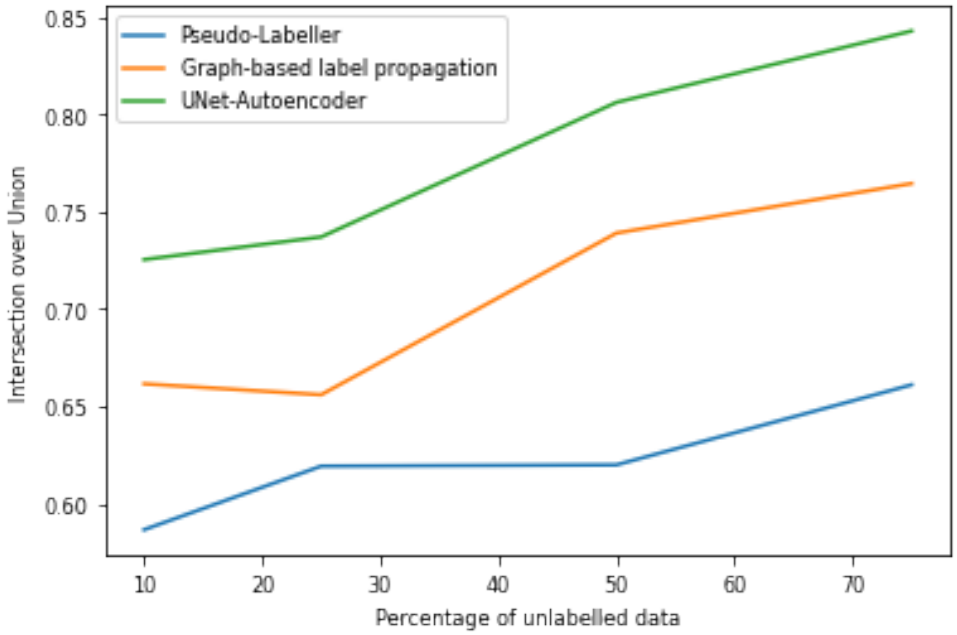


Figure 4.11: Performance comparison of different semi-supervised semantic segmentation models w.r.t. different percentages of labelled data.

The models are tested on all the tiles of the orthomosaic of the mangrove ecosystem site. These predicted segmented tiles are stitched back together to a single segmented orthomosaic using GDAL and openCV. This is done in order to create better visualizations of the performance of the models.The figures 4.12, 4.14 and 4.13 show the completely segmented image of the La Paz site that we used to train and validate different models. We can see that although the Pseudo-Labelling model doesn't improve a lot when a lot of labelled data is given for training, but the UNet-Autoencoder improves by a lot. Comparison of the two different models show that the UNet-Autoencoder gives a much better segmentation of the entire site as compared to the

Pseudo-Labelling model for all percentages of labelled data.

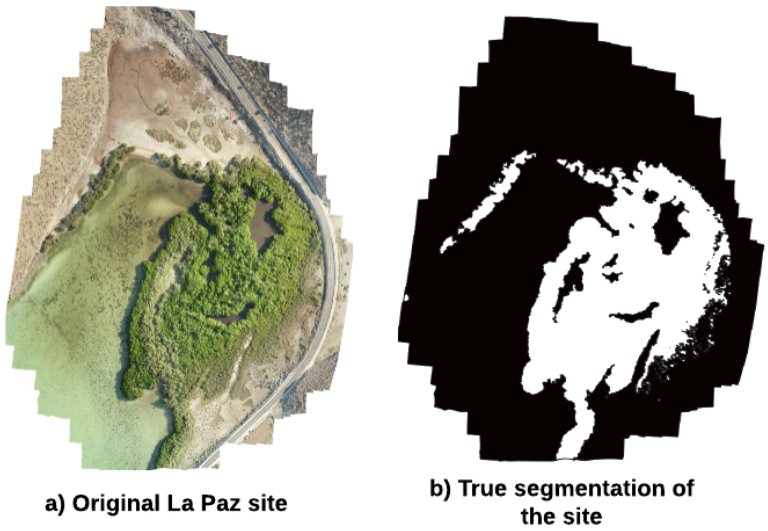


Figure 4.12: The La Paz under consideration in this research and the actual segmentation for the site.

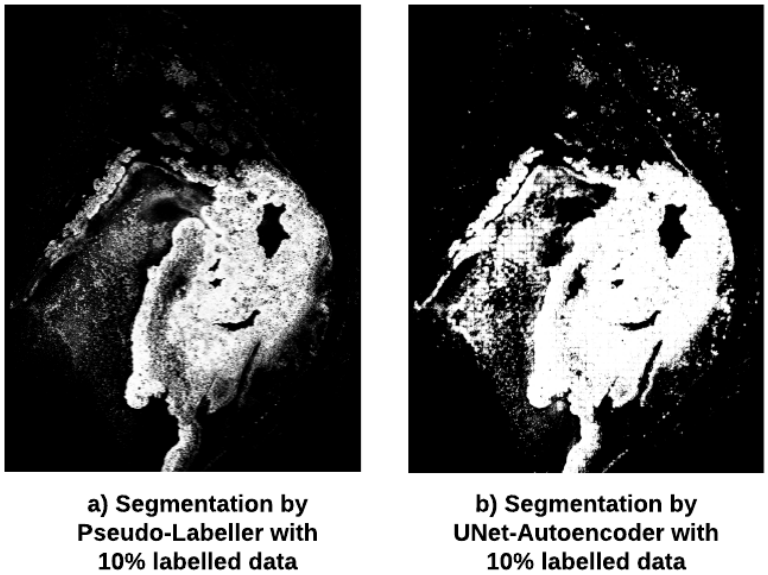
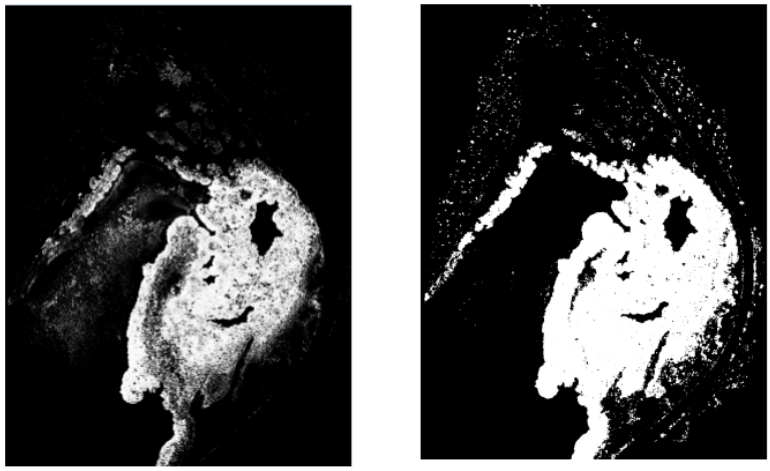


Figure 4.13: Segmentation results of the whole La Paz site obtained by Pseudo-Labeller and the UNet-Autoencoder when 10% labelled data is used for training



a) Segmentation by Pseudo-Labeller with 75% labelled data

b) Segmentation by UNet-Autoencoder with 75% labelled data

Figure 4.14: Segmentation results of the whole La Paz site obtained by Pseudo-Labeller and the UNet-Autoencoder when 75% labelled data is used for training

Chapter 5

Conclusion and Further Improvements

In this research we looked into various semi-supervised semantic segmentation methods in order to help with the conservation of mangroves. Since hand labelling the aerial imagery captured by UAV is costly in terms of human effort and time, models which can learn using a limited amount of labelled data can be very helpful. Our semi-supervised methods varied from the simplest of self-learning based Pseudo-labelling semi-supervised segmentation model to a complex high performing deep learning based UNet-Autoencoder architecture. We found that the UNet-Autoencoder performs the best in a semi-supervised setting for our segmentation task. This model attains an Intersection over Union score of around 0.72 for as little as 10% labelled data. An increase in the amount of labelled data expectantly improves the performance of the model. The UNet-Autoencoder gives an improved performance of around 0.84 IoU when 75% labelled data was given for training the semi-supervised model. The current UNet-Autoencoder model could be improved by training it for longer. Both the autoencoder and the UNet architecture can be trained for more epochs. Some post-processing steps like masking the resulting segmentations could result in better overall segmentations. More future work could be to apply the current model to different sites of mangrove ecosystems so as to see how well the current model generalizes.

Bibliography

[1] Dillon Hicks, Arden Ma, John Dorian, Astrid Hsu, Katherine Qi, Eric Lo, Ryan Kasnter, Curt Schurgers, and Octavio Aburto. Mangrove ecosystem detection using transfer learning-based convolutional neural networks and high spatial-resolution uav imagery. unpublished, 2020.

[2] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.", 2016.

[3] Stephanie S Romañach, Donald L DeAngelis, Hock Lye Koh, Yuhong Li, Su Yean Teh, Raja Sulaiman Raja Barizan, and Lu Zhai. Conservation and restoration of mangroves: Global status, perspectives, and prognosis. *Ocean & Coastal Management*, 154:72–82, 2018.

[4] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[5] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[6] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. A discussion of semi-supervised learning and transduction. In *Semi-supervised learning*, pages 473–478. MIT Press, 2006.

[7] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[9] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[10] Chandra Giri, Edward Ochieng, Larry L Tieszen, Zq Zhu, Ashbindu Singh, Tomas Loveland, Jeff Masek, and Norman Duke. Status and distribution of mangrove forests of the world using earth observation satellite data. *Global Ecology and Biogeography*, 20(1):154–159, 2011.

[11] Jingjing Cao, Wanchun Leng, Kai Liu, Lin Liu, Zhi He, and Yuanhui Zhu. Object-based mangrove species classification using unmanned aerial vehicle hyperspectral images and digital surface models. *Remote Sensing*, 10(1):89, 2018.

[12] Astrid J Hsu, Eric K Lo, John B Dorian, and Benigno Guerrero Martinez. Drone flight manual ucsd mangrove imaging procedure (version 1.2). 2019.

[13] ESRI ESRI. Shapefile technical description. *An ESRI white paper*, 4:1, 1998.

[14] Thomas Moranduzzo, Mohamed L Mekhalfi, and Farid Melgani. Lbp-based multiclass classification method for uav imagery. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2362–2365. IEEE, 2015.

[15] Mohammad Peikari, Sherine Salama, Sharon Nofech-Mozes, and Anne L Martel. A cluster-then-label semi-supervised learning approach for pathology image classification. *Scientific reports*, 8(1):1–13, 2018.

[16] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

[17] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[18] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[19] Olga Zoidi, Eftychia Fotiadou, Nikos Nikolaidis, and Ioannis Pitas. Graph-based label propagation in digital media: A review. *ACM Computing Surveys (CSUR)*, 47(3):1–35, 2015.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[21] Hongfei Li, Lili Meng, Jia Zhang, Yanyan Tan, Yuwei Ren, and Huaxiang Zhang. Multiple description coding based on convolutional auto-encoder. *IEEE Access*, 7:26013–26021, 2019.

[22] Martin Heusel, Djork-Arné Clevert, Günter Klambauer, Andreas Mayr, Karin Schwarzbauer, Thomas Unterthiner, and Sepp Hochreiter. Elu-networks: fast and accurate cnn learning on imagenet. *NiN*, 8:35–68, 2015.

[23] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[24] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.