

Identifying Rocky Intertidal Zone Plant Species using Convolutional Neural Networks

A Thesis Presented to
The Faculty of the Computer Science Department
California State University Channel Islands

In (Partial) Fulfillment
of the Requirements for the Degree
Masters of Science in Computer Science

by
Student Name: Mitali J. Shah

Advisor: Dr. Jason Isaacs

August 16, 2018

© 2018
Mitali J. Shah
ALL RIGHTS RESERVED

APPROVED FOR MS IN COMPUTER SCIENCE



8/16/18

Advisor: Dr. Jason Isaacs

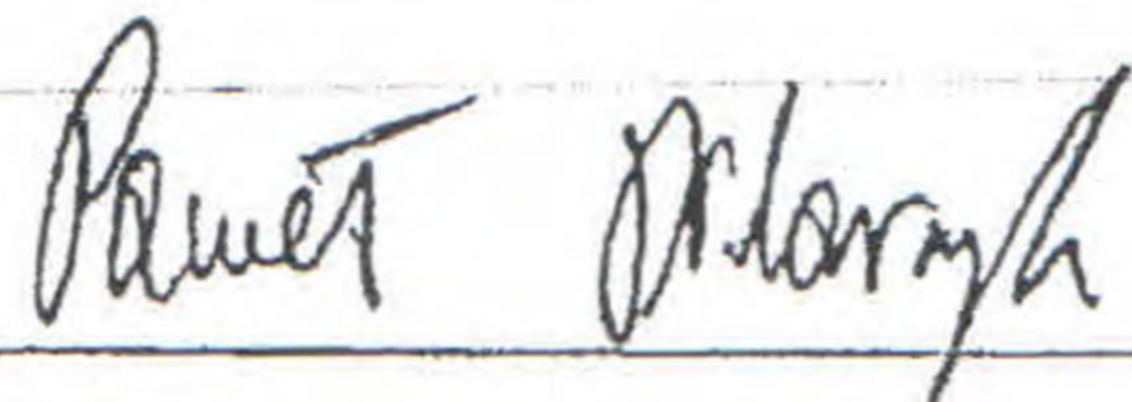
Date



8/16/18

Dr. Brian Thoms

Date

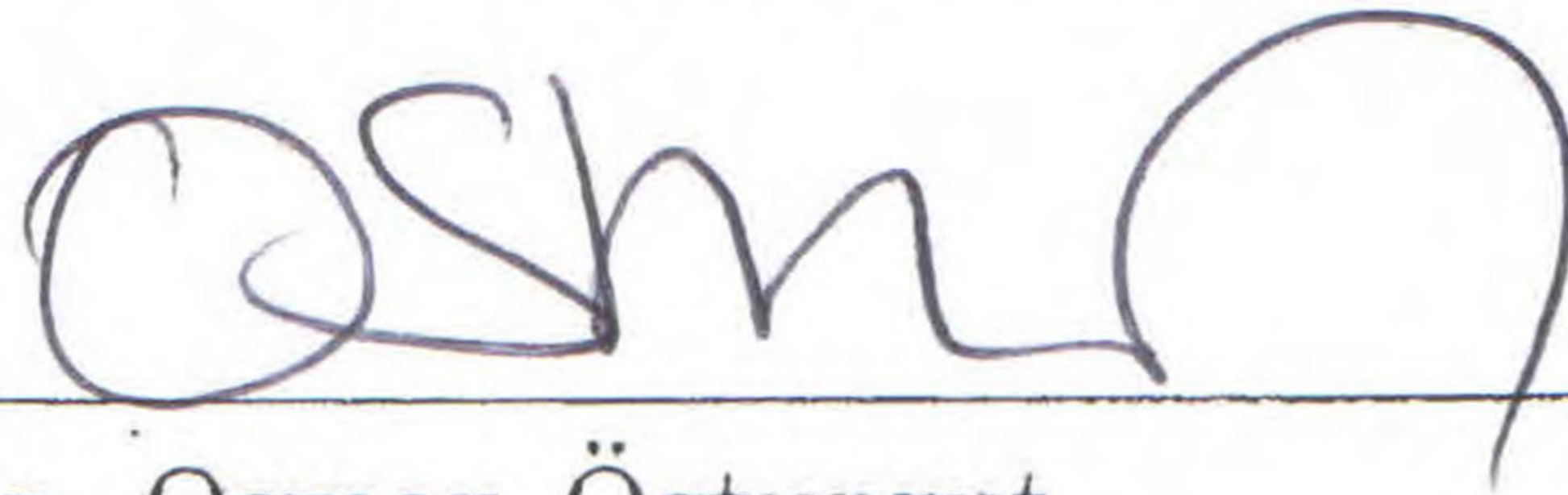


8/16/2018

Dr. Pawel Pilarczyk

Date

APPROVED FOR THE UNIVERSITY



8/31/18

Dr. Osman Özturgut

Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Identifying Rocky Intertidal Zone Plant Species using Convolutional Neural Networks

Title of Item

Machine Learning, Convolutional Neural Network, Classification, Semantic Segmentation

3 to 5 keywords or phrases to describe the item

MITALI SHAH

Author(s) Name (Print)

M. Shah

Author(s) Signature

08/16/2018

Date

Identifying Rocky Intertidal Zone Plant Species using Convolutional Neural Networks

Mitali J. Shah

August 16, 2018

Abstract

Ecological monitoring of plant and animal species helps in maintaining ecological balance. It helps in understanding the species, their assemblage, changes that occur in their assemblage and factors causing those changes. The present study involves monitoring of plant species in rocky intertidal zones of Santa Rosa Island, California. Traditionally, ecological monitoring has been done using photo transects. These photo transects are then quantified manually by humans, but quantifying a huge amount of data manually can be time-consuming and prone to errors.

The present study helps to address these problems by using a machine learning technique - semantic segmentation. Additionally, image classification is also performed. The study involves building two convolutional neural networks - one from scratch and the other using transfer learning on a publicly available network. Datasets used in the study were collected by the Biology department at California State University, Channel Islands and the network is built using a publicly available framework - Keras.

This thesis is dedicated to my dear parents, Jagdish and Bhavna Shah, my sister Henal Shah and my fiancé Saurabh Trivedi.

Acknowledgement

Firstly, I would like to thank my thesis advisor, Dr. Jason Isaacs, for his technical guidance, continuous support and encouragement throughout the process.

I would like to thank Dr. Brian Thoms and Dr. Pawel Pilarczyk for graciously accepting to serve on my thesis committee.

I would like to thank Dr. Geoffrey Dilly, Biology department at California State University, Channel Islands, for introducing me to the problem. I would also like to thank his team, especially Kaylen Meeker for providing the rocky intertidal zone dataset. I would also like to thank Mark Getzinger for helping with the computer setup.

I would like to thank my family and my fiancé and his family for their constant moral support.

Lastly, I would like to thank my roommates and friends for being there for me.

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Organization of Thesis	3
2	Background	4
2.1	Convolutional Neural Network (CNN)	7
2.2	State-of-the-art Dataset	10
2.3	Machine Learning Techniques	10
3	Objective	15
4	Dataset	17
4.1	Rocky Intertidal Zone Data	17
4.2	Preprocessing Data	17
5	Model Architecture	22
5.1	Classification	22
5.2	Semantic Segmentation	23
6	Implementation	25
6.1	Classification	25
6.2	Semantic Segmentation	28
6.3	Hyperparameters	29
7	Results	34
7.1	Classification	35
7.2	Semantic Segmentation	36
8	Conclusion and Future Work	40

List of Figures

1	Changes in assemblage of species over time at Beachers Bay, Santa Rosa Island Intertidal Zone, CA. Image courtesy: Kaylen Meeker.	1
2	A simple neural network architecture.	4
3	An underfitted, well fitted and overfitted model [5].	5
4	A loss vs epoch graph with different learning rate [7].	6
5	Explanation of gradient descent concept [8].	7
6	The convolution operation where the output matrix is the feature map [9].	8
7	Working of convolutional layer.	9
8	The max pooling operation [12].	9
9	Multi-layer structure for handwritten character recognition [16].	10
10	U-Net encoder-decoder architecture [29].	12
11	Sample image of algae species - <i>Silvetia Compressa</i>	15
12	Sample image of <i>Mytilus</i>	18
13	Data augmentation results. The various geometric transformations used are rescale, zoom, shear, rotate, horizontal flip and vertical flip.	19
14	An example of annotated image. Left image is the original image and right image is an annotated image created using Image Segmenter App. The white pixels indicate foreground region and black pixels indicate background region.	20
15	A screenshot of Image Segmenter App. The toolbox on top displays different tools used in the Graph-cut. The area under green scribbles represent foreground element and red scribbles represent background element.	20
16	A 5-layer CNN for classifying <i>Silvetia Compressa</i>	22
17	A fine-tuned U-Net architecture for performing semantic segmentation.	24
18	Results of training evaluation for classifier model without data augmentation where overfitting is seen. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	26

19	Results of the training evaluation for classifier model using few data augmentations. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	26
20	Results of training evaluation for classifier model using L2 regularizer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	27
21	Results of training evaluation for classifier model using SGD with L2 regularizer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	27
22	Results of training evaluation of U-Net model using SGD optimizer and without dropout layer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	28
23	Results of training evaluation of U-Net model using SGD optimizer and with dropout layer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	29
24	The accuracy and loss graph of the selected model for classification problem. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	32
25	The accuracy and loss graph of the selected model for semantic segmentation problem. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.	32
26	Result of unprocessed test image. The left image is original image, center image is the annotated image, right image is the predicted mask.	36
27	Result of unprocessed test image. The left image is the original image, center image is the annotated image, right image is the predicted mask.	37

28	Result of test images after histogram equalization. The left image is the original image after histogram equalization, center image is the annotated image, right image is the predicted mask.	37
29	Result of test image after adaptive histogram equalization. The left image is the original image after adaptive histogram equalization, center image is the annotated image, right image is the predicted mask.	38
30	Result of test images segmented falsely in few regions. The left image is the original image after adaptive histogram equalization, center image is the annotated image, right image is the predicted mask. The predicted mask has some extra region segmented.	38
31	The predicted mask overlapping the original image for researchers to analyze. The left image is original test image, the center image is the predicted mask and the right image is the overlapped image.	39

List of Tables

1	Classifier summary.	29
2	Segmentation summary	30
3	Confusion matrix used for evaluation.	34
4	Confusion matrix created after testing the classification model.	35
5	Confusion matrix created after testing the segmentation model.	39

1 Introduction

1.1 Introduction

Environment management is very important to maintain the ecological balance. It can operate effectively with reliable information on the changes to the environment and on the causes of those changes. Through ecological monitoring, an important source of information is provided. Ecological monitoring aims at inferring causes of ecosystem changes, by measuring ecosystem state variables in space and time [1].

Ecological monitoring in rocky intertidal zones, helps to track the species assemblage and their biodiversity. It helps in understanding the species, changes that occur in their assemblage over time and the factors that cause those changes, and to make informed decisions pertaining to the ecological balance. One such monitoring of plant and animal species in rocky intertidal zone of Santa Rosa Island is shown in the Figure 1. Figure 1 shows the variation in the assemblage of plant species during the winters over a period of three years. It is seen that during the winters of 2016 and 2017 the area was dominated by two species *Phragmatopoma Californica* and *Phyllospadix*; however, in winter 2018 *Phyllospadix* was replaced by *Silvetia Compressa*. Also, in winter 2018 *Phragmatopoma Californica* was present in abundance as compared to winter 2017.



Figure 1: Changes in assemblage of species over time at Beachers Bay, Santa Rosa Island Intertidal Zone, CA. Image courtesy: Kaylen Meeker.

Ecological monitoring can be done using point intercepts, vertical transects or photo transects. The point intercept method gives clustered data

with good resolution of the species in random portions of the overall site, vertical transects method gives a low resolution of the site but allows researchers to collect data more evenly throughout the site and photo transects method gives the highest amount of resolution of the site but the data from this method is extremely difficult to process.

In the present study, photo transects are used as the source of information. Small sections of the region are captured in each image. These images collect a huge amount of ecological data for a region and provide a snapshot in time of what the region looks like. The images quantify the presence of each species in the region. Quantifying images, allows researchers to establish a baseline and track any changes that occur in the species assemblage over time. Further details about these images are provided in Chapter 4.

Typically, these images are quantified by humans to see the seasonal variation in the assemblage of different species in a particular region. But quantifying huge amounts of data manually can be time-consuming and can contain human error. In recent years, the machine learning research community has developed many techniques, as discussed in Chapter 2, to address the problems that arise in manual quantification.

Machine learning research originates from the idea that a computer can be given the ability to learn, as a human would do, without being explicitly programmed. There are three types of machine learning techniques: Supervised learning, Unsupervised learning and Reinforcement learning.

- Supervised learning aims to learn a mapping from input to output whose correct values are provided by a supervisor [2]. Supervised learning problems are grouped into classification and regression problems. A classification problem is when the output is a categorical response value i.e., where the data can be separated into specific classes such as ‘spam’ or ‘ham’ email, or ‘red’ or ‘blue’ color. A regression problem is when the output is a continuous response such as value of a stock or price of a house in a specific area.
- Unsupervised learning aims to find regularities in data without the help of a supervisor [2]. Clustering is a type of unsupervised learning problem. Clustering is the task of grouping a set of objects/ inputs in such a way that objects/ inputs in the same group are similar to each other than to those in another group.
- Reinforcement learning [3] is learning from the environment. In this

learning, an agent (e.g. robot or controller) learns to take optimal actions based on outcomes of the past actions.

1.2 Organization of Thesis

A brief description about to the motivation of the thesis is provided so far. The remaining chapters are arranged as follows:

Chapter 2 gives the background knowledge needed to understand the functionality of the present study. This chapter explains machine learning in depth and gives a brief explanation of the available dataset. A thorough study and reference to techniques currently used and those used in the past are also included. Chapter 3 defines the problem statement of the present study in detail. This chapter introduces the dataset and techniques used in the present study to solve the problem. Chapter 4 gives brief details about the datasets used in the present study. This chapter includes the details about image preprocessing techniques used. Chapter 5 describes the model architectures used in the present study. Chapter 6 demonstrates the implementation of the solution provided by the present study, based on the concepts mentioned in Chapter 2. This chapter explains how each approach is used to solve the problem. Chapter 7 presents the results using evaluation metrics. Chapter 8 provides the concluding remarks and possible extensions to the present study.

2 Background

This chapter focuses on the details about basic concepts and terminology used in machine learning. It also includes the details about related machine learning techniques.

Neural Networks: One of the definitions of a neural network was provided in [4] where the author stated that “*a neural network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*”

The network’s organization and functioning are similar to that of neurons in the human brain. Each layer in the neural network consists of group of small units called neurons, followed by an activation function. An activation function helps to non-linearly identify important features. The neurons in one layer are connected to the neurons of the next layer.

The neural networks contain adaptive weights between the neurons. These weights are then tuned by the learning algorithm. A cost function is used along with a learning algorithm to optimize the model. Mathematical calculations are performed by each node and the results are transmitted to all the connected nodes. Figure 2 shows a simple neural network with an input layer containing inputs x_1 , x_2 , x_3 having weights w_1 , w_2 , w_3 . Mathematical operations are performed, and the result is passed to the hidden layer. The hidden layer then produces the input for the output layer which gives the outputs y_1 and y_2 .

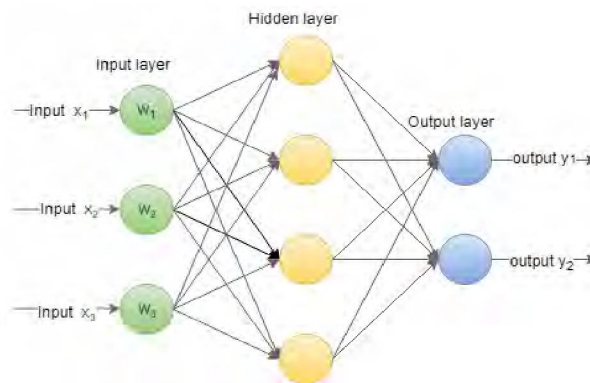


Figure 2: A simple neural network architecture.

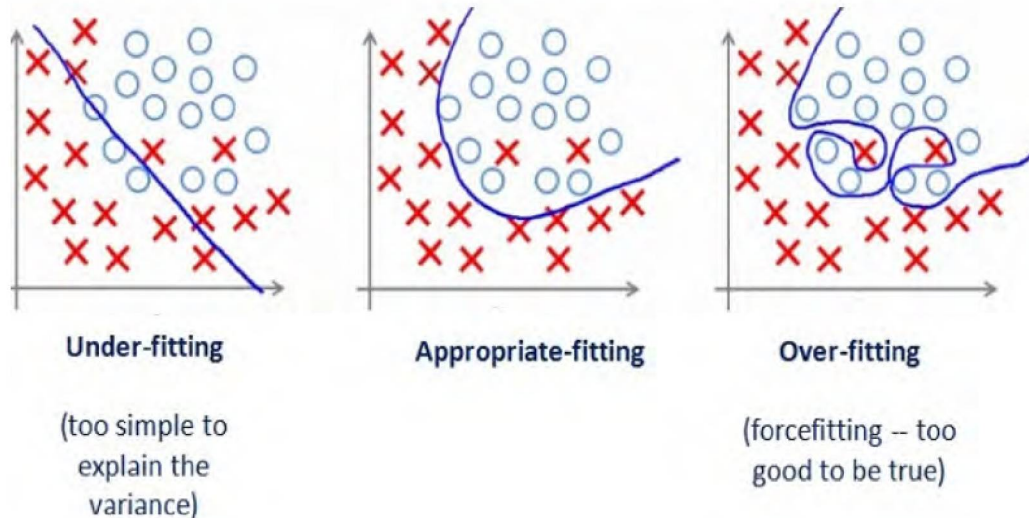


Figure 3: An underfitted, well fitted and overfitted model [5].

Overfitting and Underfitting: A machine learning algorithm is required to perform well on unseen data. If the algorithm does not perform well on the unseen data, then there are two reasons: either the model does not have enough capacity, or it has too much capacity. The case where the model has too much capacity is called overfitting. It happens when the model captures feature as well as noise from the training data. Here, the training error is low. However, it fails to perform well with test data and the test error is high. The case where model has less capacity is underfitting. It means that the model is not able learn well from the training data itself. In this case, both the training and the testing error is high.

A model that has both the training and testing error low is a well fitted model. Figure 3 shows the different cases of model fit. The left is an underfit model, the center one is a well fit model and the right one is an overfit model.

Hyperparameters: Hyperparameters are the variables that determine the network structure and how the network will be trained. They are set by the user before the training. They regulate the capacity of the network. Hyperparameters can be: number of layers in a network, learning rate¹, etc.

¹Learning rate is a hyperparameter that controls the weights adjustment required by the network to converge [6].

They are parameters that are not learned during training.

Several models are tested with different hyperparameters and the model that returns the lowest error rate is selected. For tuning the hyperparameters, the dataset is divided into three sets: training, validation and test datasets. The training dataset is used to fit the model, the validation dataset is used to provide an unbiased evaluation of the model fit on the training dataset while fine-tuning the hyperparameters and the test set is used to evaluate the model performance.

Loss function: A loss function maps one or more variables onto a real number, and this value represents loss. A loss is the discrepancy between the predicted function and the target function. The learning of the model involves reducing the loss described by the loss function. For classification and semantic segmentation, cross-entropy loss is used. **Cross-entropy loss** is a log loss function. It is used when the output of the model is a probability distribution (probability of the test data belonging to a certain class).

Optimization: An optimization algorithm helps to minimize the loss function during training. A learning rate hyperparameter is mostly used in optimization algorithms. It is an important hyperparameter of selection. With a very small learning rate, the model will take a long time and may be stuck at a local minimum. With a large learning rate, the model may not converge. Figure 4 shows the loss vs epoch² graph for different learning rates.

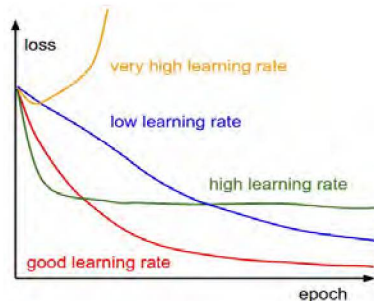


Figure 4: A loss vs epoch graph with different learning rate [7].

Gradient Descent [8]: It is an iterative optimization algorithm to find a local minimum of a function. It can be illustrated from the following

²One epoch means the entire dataset is passed through the neural network once.

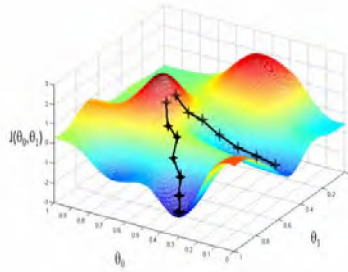


Figure 5: Explanation of gradient descent concept [8].

example. Suppose a person is on a mountain and wants to go back to the valley. The visibility is low due to fog. Therefore, only local information is available to reach the valley. So, the person searches for steepest descent from the current position and takes a step. By following the descending path at each position, the person is likely to reach the valley. Figure 5 explains the concept of gradient descent. The gradient descent algorithm calculates the gradient on the whole dataset and performs only one update. Therefore, it is slow to converge and difficult to control when the dataset is too large.

Stochastic Gradient Descent (SGD): It is a variant of the gradient descent algorithm. It solves the problems of traditional gradient descent with a fixed learning rate. It performs parameter updates for each training example and therefore is faster than gradient descent. Since it performs parameter updates, there are fluctuations in the loss function which sometimes complicates the convergence to the minimum.

Adaptive Moment Estimation (Adam): It is an algorithm that computes adaptive learning rates for each parameter. It is a popular optimization algorithm because it is efficient, and it converges fast.

2.1 Convolutional Neural Network (CNN)

CNN is a type of neural network that is useful in finding patterns in images. They are neural networks that have a convolutional layer as the first layer in the network. The CNN architecture is built using different layers, like convolutional layers, pooling layers and a fully connected layer. Parameters are calculated at each layer.

As mentioned earlier, the first layer in a CNN is the convolutional layer. The convolutional layer is a very important layer as most of the computa-

tional work is done by this layer. It contains a filter that convolves on the input image and computes the dot products between the filter value and pixel value of the image, summing up to produce an activation/ feature map. This feature map helps in identifying features. Figure 6 shows the convolution operation. In the figure, I represents the image, K represents the filter and $I * K$ represents the resulting feature map.

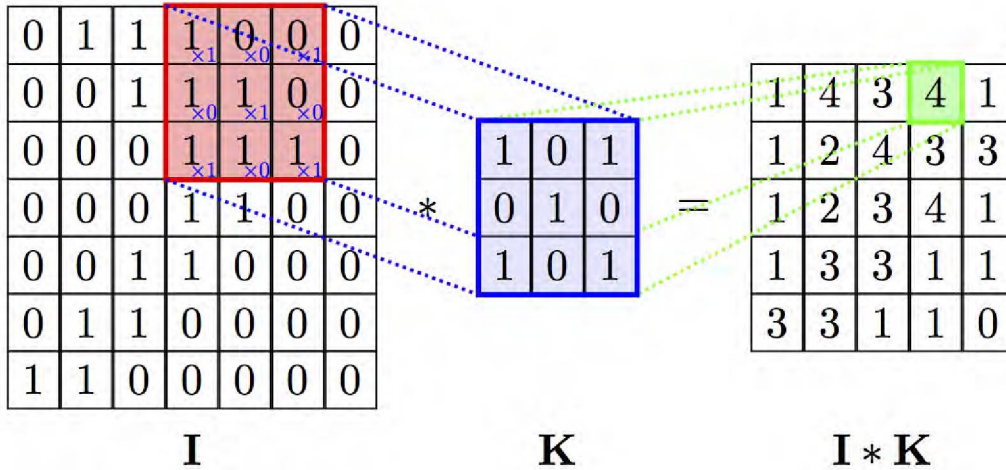


Figure 6: The convolution operation where the output matrix is the feature map [9].

A convolutional layer is usually followed by an activation layer to introduce nonlinearity to the network using an activation function (A). The activation function decides which neuron should be fired.

$$A = \Sigma(\text{weight} * \text{input}) + \text{bias}$$

A few examples of non-linear activation functions include tanh ($A = \frac{2}{1+e^{-2x}} - 1$), sigmoid ($A = \frac{1}{1+e^{-x}}$) and Rectified Linear Units (ReLU) ($A = \max(0, x)$). ReLU has become a popular activation function in the last few years and works better compared to the other two [10]. ReLU activates neurons with positive values thereby reducing the computationally expensive exponential operations like in sigmoid and tanh. It enables the network to converge faster without affecting the accuracy as seen in [11]. It prevents the problem of vanishing gradient. Figure 7 shows the working of a convolutional layer. The next layer typically, is a pooling layer also known as downsampling layer. It is used to reduce the dimensionality of each feature map but retains important

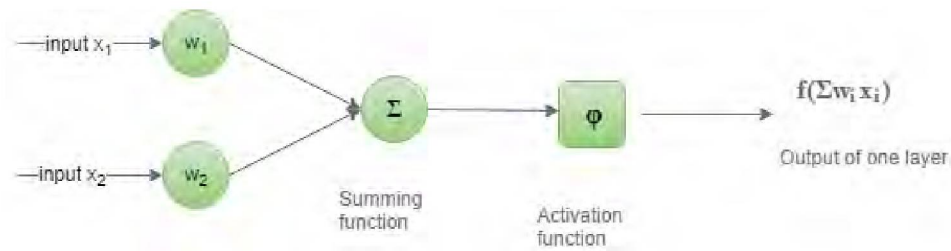


Figure 7: Working of convolutional layer.

information. There are various pooling methods, but mostly max pooling is used. In max pooling, the largest element in the feature map within the window is taken. Figure 8 shows the max pooling operation.

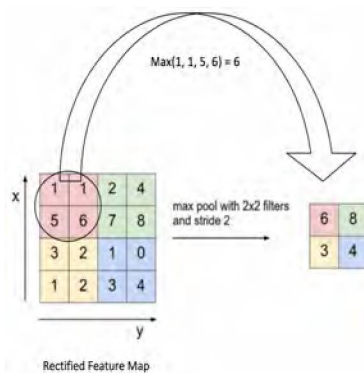


Figure 8: The max pooling operation [12].

Dropout layers [13] are used in CNN as a regularizer³. They reduce overfitting by preventing complex co-adaptions⁴ on training data. The final layer that is used in a typical CNN is a fully-connected layer. This layer looks at the high-level features, in the output from previous layer, that strongly correlate to a particular class. An example of a CNN is shown in Figure 9.

³Regularization is a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting [14].

⁴Co-adaptions is when two or more neurons repeatedly begin to detect the same features [15].

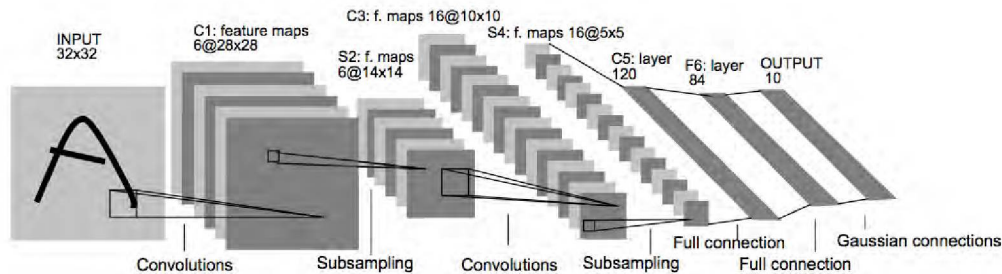


Figure 9: Multi-layer structure for handwritten character recognition [16].

2.2 State-of-the-art Dataset

Images are becoming the fastest growing content. Datasets are created using these images to analyze and find patterns. One of the earliest datasets used for machine learning was proposed by Fei Fei Li (Chief Scientist of AI/ML at Google Cloud and Director of the Stanford Artificial Intelligence Lab and the Stanford Vision Lab), who created ImageNet [17] - a large-scale ontology of images built upon the backbone of the WordNet structure. The other state-of-the-art image datasets like Modified National Institute of Standards and Technology (MNIST)[18] database - a large database of handwritten digits, Pascal Visual Object Classes (VOC), Microsoft Common Objects in Context (COCO) [19], were created to perform machine learning tasks like image classification, object localization, object recognition, semantic segmentation and instance segmentation. These datasets were used in benchmark models discussed in the following machine learning techniques.

2.3 Machine Learning Techniques

Traditionally, images were learned by Support Vector Machines (SVM)[20] on a histogram of local features. Current approaches use artificial neural networks. With the availability of a huge amount of data and increased processing power, these approaches have led to human-like performance in image classification, facial recognition and image segmentation.

Image classification: Image classification is to classify images based on the dominant object in the image. It is an important and challenging problem in the field on Computer Vision. Using machine learning for image classifica-

tion saw early success in 2012, when AlexNet [11], a large and deep CNN won the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC)⁵ with an error rate of 15.4%. Since then, variants of CNNs were produced for the ILSVRC and they have exceeded human accuracy, which is considered to lie in the 5-10% error range.

Other networks similar to Alexnet were built e.g. ZfNet [21] - a modified AlexNet (The network obtains more information from the training data by using 7×7 filter instead of 11×11 filter in the first convolution layer) was the winner for ILSVRC 2013 with an error rate of 11.4%. A new network with Inception concept was built. Inception includes replacing conventional convolutional filter with complex filters to increase their learning abilities and abstraction power [22]. GoogleNet [23] - a complex 22-layer network is based on the concept of Inception. It won the ILSVRC 2014 with an error rate of 6.7%.

Visual Geometry Group (University of Oxford) Network VGGNet [24] is a simple and effective network that uses stacks of small-kernel convolution instead of a large-kernel convolution network architecture. It did not win the ILSVRC but is popular for image classification and localization task. The most recent Squeeze-and-Excitation Network(SENET)[25] architecture with a top-5 error rate of 3.79% was developed to achieve the state-of-the-art accuracy on ILSVRC 2017 for classification and localization. There are many other networks that were built to test against various datasets.

Semantic Segmentation: Semantic segmentation is understanding the image at the pixel level, i.e., each pixel of the image is labeled with the object class it belongs to. Here an image is trained along with the image mask that contains the part of the image concerned (foreground, and the remaining is background).

Before deep learning, approaches like TextxonForest [26] and Random Forest [27] based classifiers were used for semantic segmentation. Fully Convolutional Networks (FCN) for Semantic Segmentation [28], was the CNN proposed for semantic segmentation. It was a dense network without a fully connected layer producing segmented images. Using a CNN for segmentation was not desirable because of pooling layers in it. The pooling layers increase the field of view and collect the information but discard the location

⁵A benchmark challenge in object category, classification and detection on hundreds of object categories and millions of images.

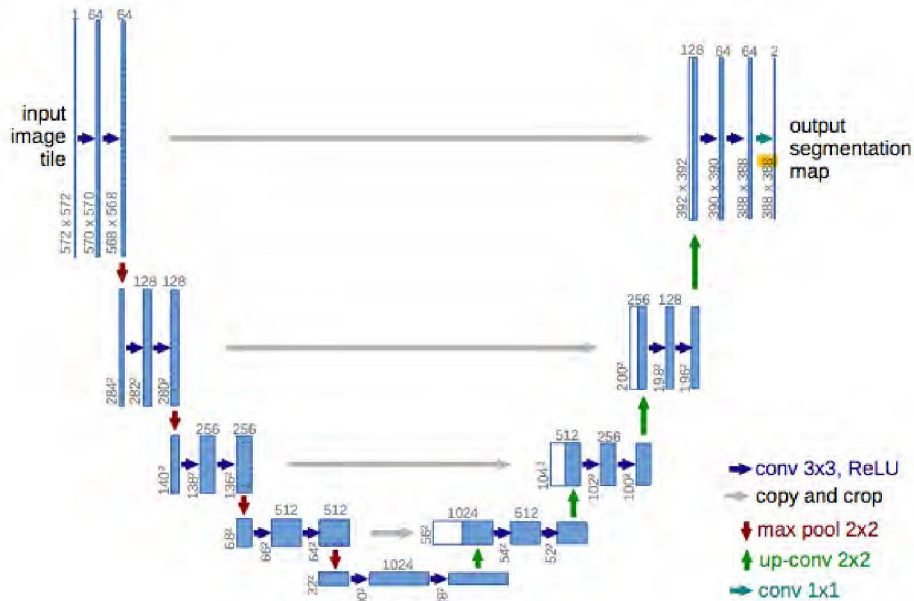


Figure 10: U-Net encoder-decoder architecture [29].

of the information. However, in semantic segmentation, the source of the information is necessary, so two classes of networks evolved to tackle this problem: One is an encoder-decoder and the other is a Conditional Random Field (CRF).

In an encoder-decoder network, the encoder gradually reduces the spatial dimension with pooling layers and the decoder gradually recovers the object details (through the connection between encoder and decoder) and spatial dimension. There are usually short cut connections from encoder to decoder to help the decoder recover the object details better. CRFs are graphical models that smoothen segmentation by observing that similar intensity pixels tend to belong to the same class. CRF post-processing is used after segmentation.

FCN and SegNet [30] were two initial encoder-decoder architectures. These architectures did not have a CRF. Multi-Scale Context Aggregation by Dilated Convolutions [31] and DeepLab [32] were based on dilated convolutions that performs convolution operations with a modified (wider) kernel. U-Net [29] an encoder-decoder network working on a small number of bio-medical images and DeepLab v3 [33] are a few networks used for semantic segmentation. Figure 10 shows U-net encoder-decoder architecture. It is a

U-shaped network, hence the name.

Training of all the above-mentioned networks (both image classification and semantic segmentation) required huge datasets except U-Net. But in many domains very few data samples are available for training. The problem with a small dataset is that it leads to overfitting and reduces the accuracy of the network. To overcome this problem data augmentation, dropout [13] and transfer learning [34] were evolved.

Data Augmentation: In order to reduce overfitting caused by a small dataset, data enhancement techniques are used to increase the amount of data. Data enhancement involves adding a few geometric transformations like flip, rotate, shift, zoom, scale, contrast, noise and color to the original image dataset to increase the amount of dataset.

Transfer Learning: Transfer learning is used to take the knowledge learned in a model and apply it to another task. This helps to use existing networks without worrying about the computational power required to train the network. There are three major transfer learning scenarios:

- CNN as a fixed feature extractor: In this method a CNN pretrained on an existing dataset is used. The last fully-connected layer is removed, and the remaining network is treated as a fixed feature extractor for the new dataset.
- Fine-tuning the CNN: This method, not only involves replacing and retraining the top layers of the CNN, but also fine-tuning the weights of the pre-trained network. All layers can be fine-tuned or a higher-level portion of the network is fine-tuned while keeping the earlier layers fixed
- Pretrained models: Since it takes time to train a CNN, some people release the model weights of their CNN trained on the state-of-the-art datasets which can be used by others on their new datasets

CNN features are more generic in the early layers and more original dataset specific in the later/higher layers. The selection of the transfer learning depends on various factors, but the size of the dataset and its similarity to the original dataset are the most important ones.

- If the new dataset is small and similar to the original dataset then using CNN as feature extractor is beneficial to avoid overfitting.
- If the new dataset is large and similar to the original dataset then fine-tuning is used.
- If the new dataset is small and different from the original dataset then it is better to train the SVM classifier using activations from earlier layers.
- If the new dataset is large and different from the original dataset then fine-tuning partially/completely is appropriate.

Example networks like [35], [36] and [37] were successfully created using transfer learning.

3 Objective

The goal of the thesis is to perform machine learning techniques like image classification and semantic segmentation on the photo transects containing a specific algae species, *Silvetia Compressa*, as in Figure 11 found in two rocky intertidal zones in Santa Rosa Island - Beachers Bay and Skunk Point.



Figure 11: Sample image of algae species - *Silvetia Compressa*.

The objective can be defined as:

- Conduct a brief study on different machine learning techniques for computer vision.
- Conduct a brief study on ways to apply machine learning techniques on the given dataset.
- Analyze the dataset and apply image preprocessing required before applying machine learning.
- Apply data augmentation techniques to increase the dataset size.

- Build a classifier using a convolutional neural network. For the convolutional neural network, try different variations of layers and hyperparameters and compare them.
- Build a network to perform semantic segmentation using transfer learning. For segmentation, a fine-tuned U-Net model is used.

4 Dataset

This chapter provides details about the datasets used in the current study. It contains information like where and how the images were captured and the preprocessing techniques used on the images for machine learning.

4.1 Rocky Intertidal Zone Data

The images of various species were collected from photo transects at two rocky intertidal zones in Santa Rosa Island - Beachers Bay and Skunk Point. These images were captured using an SLR camera. For capturing the images, eleven photo transects, each 20 *m* long and spaced 3 *m* away from one other, were placed in each zone. A rig of 1 *m* × 1 *m* was used to capture a single image and 58 such images were captured every 35 *cm* in order to create 65% overlap. The images were preferably captured during low-tide. The low tides in these zones were mostly during the night. Due to this many images were captured using flash light and with this variability in the lighting conditions different colors of the same species were observed.

The images contain nine different species namely *Mytilus*, *Silvetia Compressa*, *Phragmatopoma Californica*, *Phyllospandix*, *Endocladia*, *Ulva*, *Anthropleura Sola* and Red Algae. The present study focuses on *Silvetia Compressa*. For classification, another species - *Mytilus* was used during the training process. Figure 12 shows sample image of *Mytilus* used during the training of classification model. Each image is an RGB image with a resolution of 3400 × 3400 approximately.

A total of 592 images having dominant species as *Silvetia Compressa* were taken. These images contain *Silvetia Compressa* along with other species. Out of the 592 images only 200 images that contain this species in abundance were selected for training purpose and 50 images for validation and remaining were used as testing datasets.

4.2 Preprocessing Data

Classification: For image classification using CNN, a huge amount of data is required. Since a small amount of data was available, data augmentation was used. The *ImageDataGenerator* class in Keras [38] provides the ability to increase the size of datasets by altering the images. It has various altering parameters and the current study uses *rescale*, *zoom_range*,

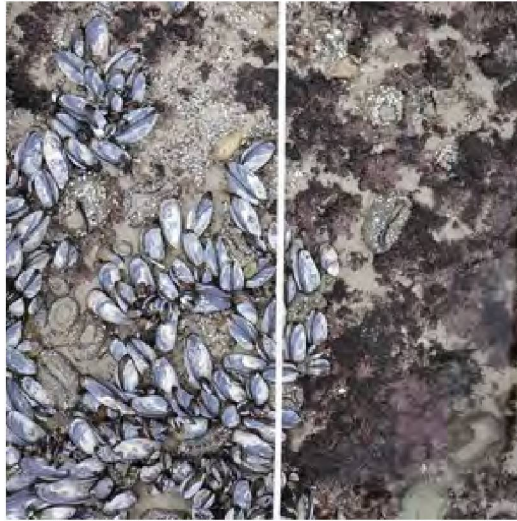


Figure 12: Sample image of Mytilus.

shear_range, *rotation_range*, *vertical_flip* and *horizontal_flip* parameters for training datasets. For validation datasets, only the *rescale* parameter was used.

The images have RGB coefficients in range 0-255 that are very high for the model to process. So *rescaling* i.e. multiplying the data with a factor of $1/255$. is done to target values between 0 and 1. The *zoom_range* randomly zooms inside the image and the *shear_range* applies a random shearing transformation. The *rotation_range* defines the amount of rotation. The *vertical_flip* and the *horizontal_flip* is for randomly flipping half of the images vertically and horizontally respectively. Figure 13 shows data augmentation of a sample image using ImageDataGenerator class in Keras.

Semantic Segmentation For semantic segmentation, annotated images (images containing only foreground object i.e. *Silvetia Compressa*) were created. Here, out of total images only 85 images and their labels (annotated images) were used for training and 15 images and their labels were used for validation. The entire dataset was randomly split into training and validation dataset while training the model. Figure 14 shows an example of a labeled image. The left image is the original image and the right image is the image label containing only the region covered by *Silvetia Compressa*. For creating image labels, the Image Segmenter App in Matlab Image Processing Toolbox

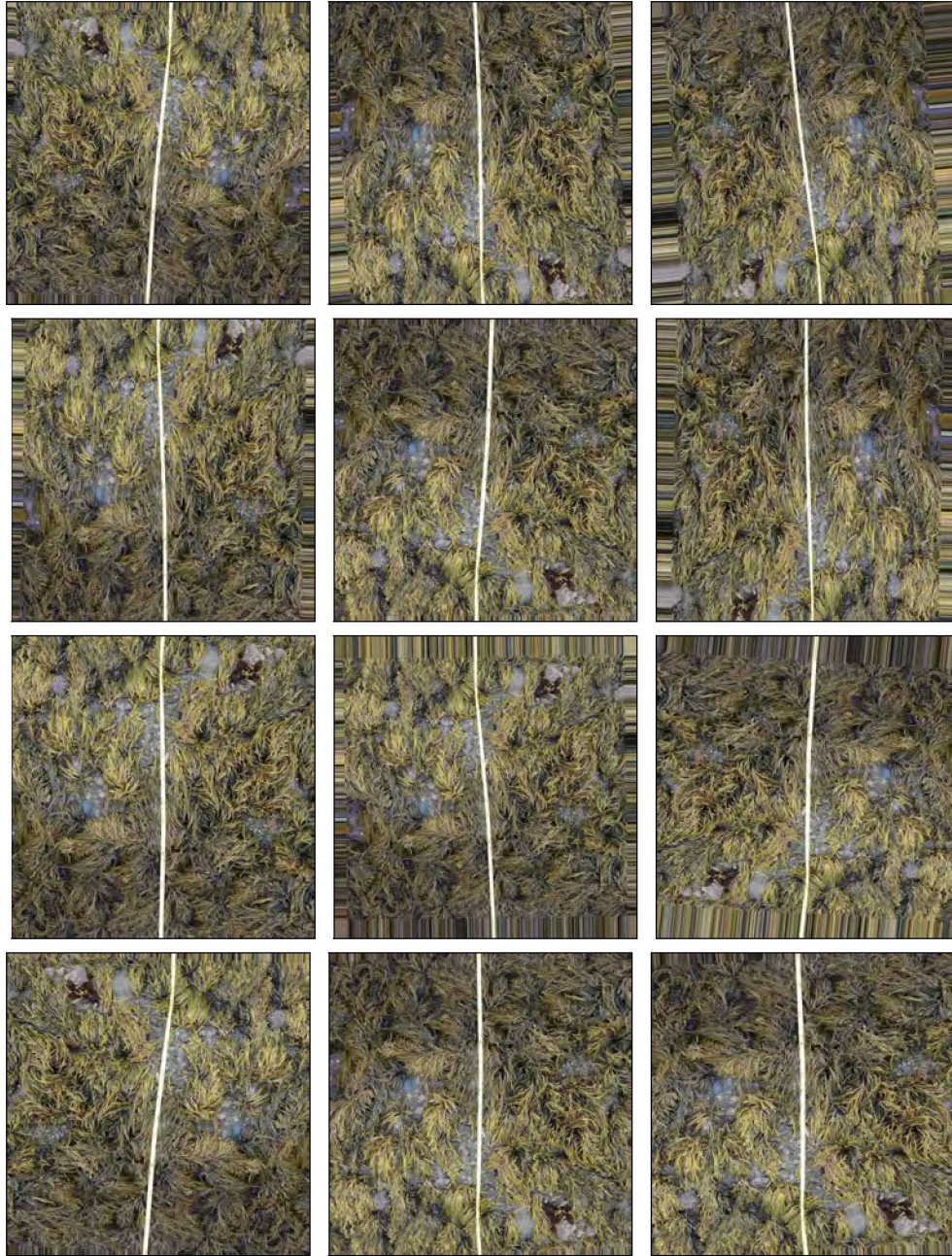


Figure 13: Data augmentation results. The various geometric transformations used are rescale, zoom, shear, rotate, horizontal flip and vertical flip.



Figure 14: An example of annotated image. Left image is the original image and right image is the annotated image created using Image Segmenter App. The white pixels indicate foreground region and black pixels indicate background region.

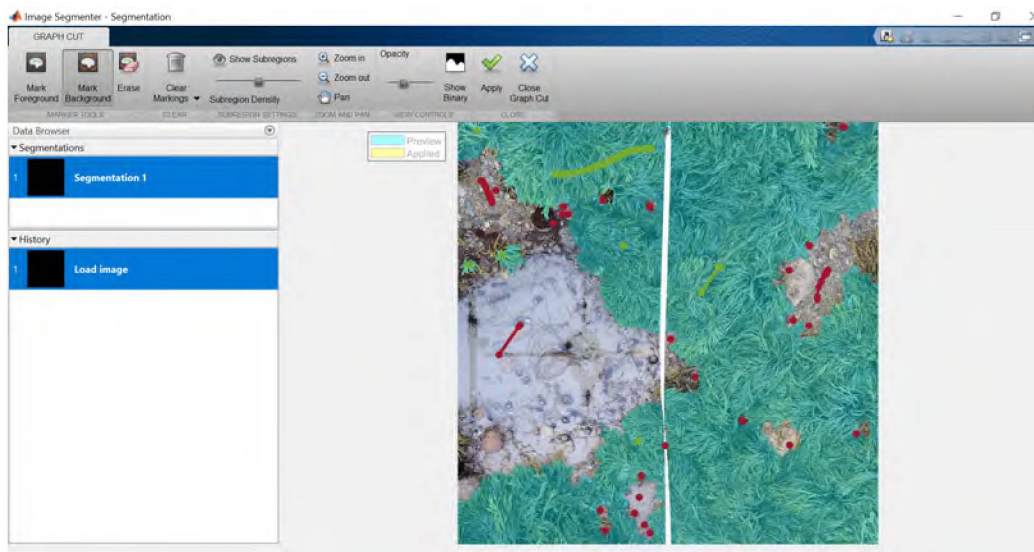


Figure 15: A screenshot of Image Segmenter App. The toolbox on top displays different tools used in the Graph-cut. The area under green scribbles represent foreground element and red scribbles represent background element.

was used.

The *Image Segmenter App* provides many different ways to annotate an image. The Graph-cut method was used in the present study. Graph-cut is a semi-automatic segmentation technique used to annotate images into foreground and background elements. To mark foreground and background elements, lines called scribbles are drawn. Based on the scribbles, the image segmenter segments the images automatically.

The segmented image is a binary image with white pixels indicating foreground (Silvetia Compressa) region and black pixels indicating the background region. The segmented image might have some imperfections, so morphological tools like dilation and erosion are used to fix the imperfections and to create a well-defined border. The segmented binary image was then stored and used as label while training the model. Figure 15 shows the working of the Graph-cut tool in the Image Segmenter App.

A total of 200 images were annotated out of which 100 were used as training and validation datasets and remaining were used as ground truth images to analyze the predicted mask of the test images. Further details about this are provided in Chapter 6.

5 Model Architecture

This chapter contains details about the models used for classification and semantic segmentation in the present study.

5.1 Classification

For classification, a 5-layer deep CNN was created in Keras. It consists of four convolutional layers and a final fully connected layer. The input shape of the first layer is 300×300 , the feature map is 32, having filter size of 5×5 and an ReLU activation function. The other three convolutional layers consist of 32, 64 and 64 feature maps. A max pooling layer is used after each convolutional layer. Dropout layers are added to avoid overfitting. The model was trained and validated on two sets of images, one set containing *Silvetia Compressa* and the other set containing *Mytilus*. Figure 16 shows the model architecture used.

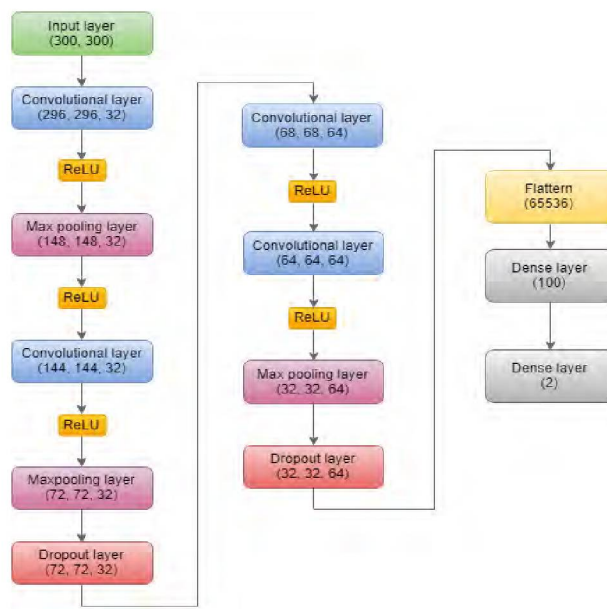


Figure 16: A 5-layer CNN for classifying *Silvetia Compressa*.

5.2 Semantic Segmentation

For semantic segmentation, the present study uses the existing U-Net model architecture and performs fine-tuning. U-Net was developed to perform semantic segmentation on microscopy images. As mentioned earlier, U-Net is an encoder-decoder architecture with skip connections. It consists of an encoder (contracting) path and a decoder (expansive) path. In Chapter 2, Section 2.3, Figure 10, the left part is the contracting path and the right part is the expansive path.

The left part follows a typical CNN architecture with repeated application of two 3×3 convolutional layers, each having an ReLU activation function, and followed by a 3×3 pooling layer with the max pooling operation having stride⁶ of 2. Downsampling operations are performed in this part. Downsampling is a max pooling layer which is an operation that summarizes each neighborhood of 2×2 neurons with its maximum value thereby reducing the dimension of the data by a factor of 4.

Each step in the expansive part has two 3×3 convolutional layers followed by upsampling operations that double the output layer's image dimension by repeating each neuron's value twice. The skip connections that are used are operations that merge the output of last convolutional layer of each step at the downsampling part onto the output of convolutional layer with the same resolutions at the upsampling part.

The U-Net architecture was fine-tuned, as shown in Figure 17, for the present study as mentioned below:

- An RGB input image of size 512×512 was used.
- No data augmentation was performed.
- Dropout layers were added to avoid overfitting caused in small training datasets.
- Instead of a SGD optimizer an Adam optimizer was used.

⁶A stride is defined as the number of pixels by which the filter matrix shifts over the input matrix while performing convolution [39].

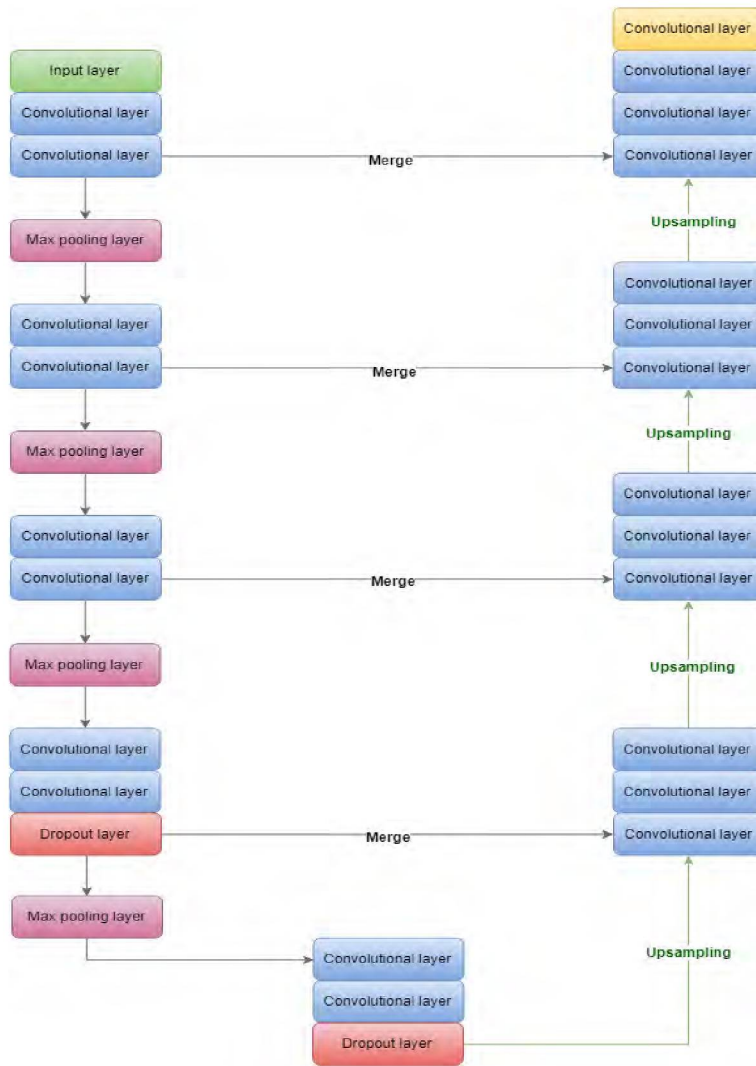


Figure 17: A fine-tuned U-Net architecture for performing semantic segmentation.

6 Implementation

This chapter contains details about performing training on the models discussed in Chapter 5. The training was performed on a machine with an Operating System (OS) - Ubuntu 16.04 and a Graphical Processing Unit (GPU) - Nvidia GTX 1080Ti installed. Using a GPU, speeds up the training process. Python code running on the Keras framework with a Tensorflow backend was used.

Keras: Keras is a high-level neural networks Application Programming Interface (API), written in Python enabling fast experimentation of various machine learning techniques. It runs on top of either TensorFlow, Theano or Microsoft Cognitive Toolkit (CNTK), which are software libraries for machine learning. Keras provides:

- Easy and fast prototyping through user friendliness, modularity and extensibility.
- Support for convolutional neural networks, recurrent networks and their combination.
- CPU and GPU compatibility.

6.1 Classification

The **data augmentation** parameters were selected based on the training evaluation results. The training evaluation results of a model that had a high training and validation accuracy and a low training and validation loss were selected. The first try was just rescaling the data. Figure 18 shows the result of training without performing data augmentation. Since the data size was small, overfitting was experienced. Next try was using zooming, shearing along with rescaling. A huge difference between training and validation accuracy and losses was seen, as shown in Figure 19. So instead of just zooming and shearing, mirroring using horizontal and vertical flip and rotation were performed which gave better results as compared to the previous data augmentations. The evaluation metrics are discussed later in this chapter.

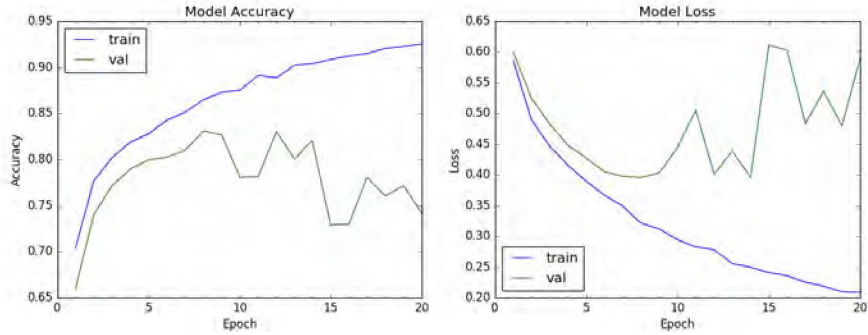


Figure 18: Results of training evaluation for classifier model without data augmentation where overfitting is seen. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

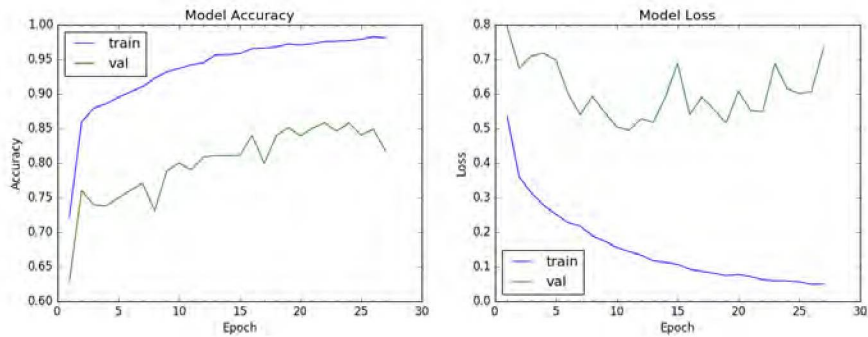


Figure 19: Results of the training evaluation for classifier model using few data augmentations. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

Regularizers were used to avoid overfitting. Dropout layers with probability between 0.2 and 0.5 were evaluated. Two dropout layers with probabilities 0.25 and 0.5 were added. This enabled capturing more features in lower layers but avoiding overfitting. A different regularization technique using L2 regularizer was also evaluated. But the evaluation results were different from the expected results. Figure 20 shows the training and validation accuracy and losses obtained using L2 regularizer. It was seen from the figure that the validation accuracy was fluctuating and higher than the training accuracy indicating that a lot of important features were dropped while training.

Optimizers are needed to train the model and set the learning rates. Three different optimizers were tried. The first one being the standard optimizer - SGD, but it is not adaptive. Figure 21 shows the result of using SGD with L2 regularizer. The next try was using RMSProp, which allows adaptive learning rates, but it did not give expected results. Here, there was a large difference between training and validation losses. So, Adam optimizer was used, because it gave the best results as compared to the other two. Adam also computes adaptive learning rates for each parameter. Adaptive learning rate is known to give good results with sparse datasets [40].

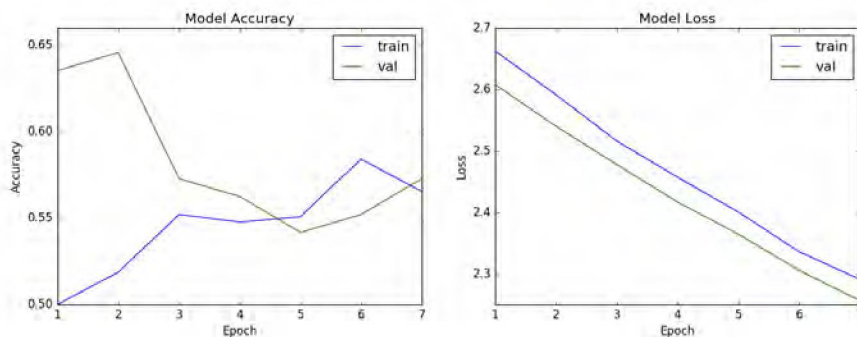


Figure 20: Results of training evaluation for classifier model using L2 regularizer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

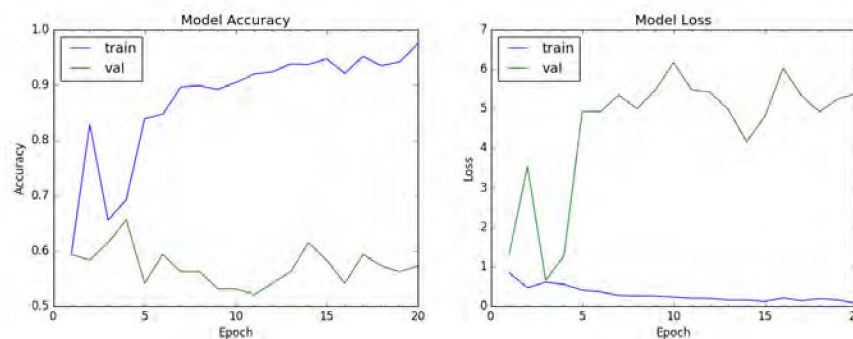


Figure 21: Results of training evaluation for classifier model using SGD with L2 regularizer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

6.2 Semantic Segmentation

For segmentation, no data augmentation was performed. The first try was using the original U-Net model without dropout. The original U-Net model used SGD optimizer. The training evaluation results can be seen in Figure 22. From the figure, it is seen that the training accuracy and the validation accuracy was low for semantic segmentation. Also, the training and validation loss was high.

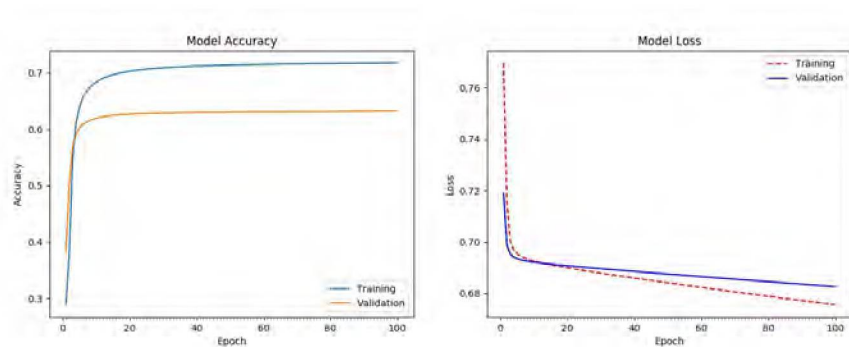


Figure 22: Results of training evaluation of U-Net model using SGD optimizer and without dropout layer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

So, dropout layers were added for the next try. The training evaluation results can be seen in Figure 23. From the figure, it is seen that there was no change in the training accuracy and the validation accuracy. Also, the training and validation losses are high. Then the modifications mentioned earlier i.e. addition of two dropout layers and using an Adam optimizer were performed.

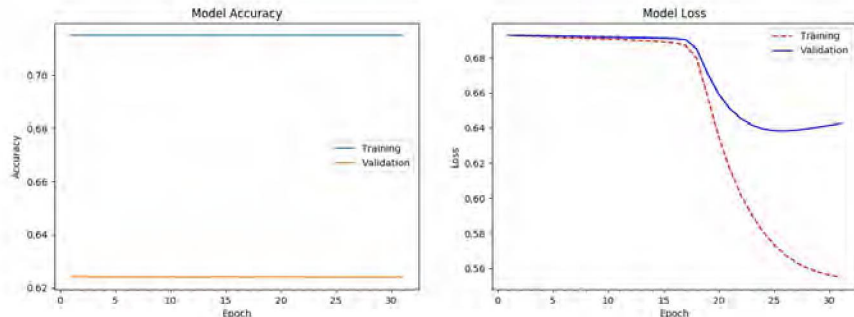


Figure 23: Results of training evaluation of U-Net model using SGD optimizer and with dropout layer. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

6.3 Hyperparameters

There are millions of parameters during training and they can cause the CNN to behave differently than expected, so hyperparameters are tuned methodically to get good convergence. The summary of each model with number of parameters at each layer is provided in Table 1 and Table 2.

Table 1: Classifier summary.

Layer (type)	Output Shape	Param
conv2d_9 (Conv2D)	(None, 296, 296, 32)	2432
activation_9 (Activation)	(None, 296, 296, 32)	0
max_pooling2d_7 (MaxPooling2)	(None, 148, 148, 32)	0
conv2d_10 (Conv2D)	(None, 144, 144, 32)	25632
activation_10 (Activation)	(None, 144, 144, 32)	0
max_pooling2d_8 (MaxPooling2)	(None, 72, 72, 32)	0
dropout_7 (Dropout)	(None, 72, 72, 32)	0
conv2d_11 (Conv2D)	(None, 68, 68, 64)	51264
activation_11 (Activation)	(None, 68, 68, 64)	0
conv2d_12 (Conv2D)	(None, 64, 64, 64)	102464
activation_12 (Activation)	(None, 64, 64, 64)	0
max_pooling2d_9 (MaxPooling2)	(None, 32, 32, 64)	0
dropout_8 (Dropout)	(None, 32, 32, 64)	0
Continued on next page		

Table 1 – continued from previous page

Layer (type)	Output Shape	Param
flatten_3 (Flatten)	(None, 65536)	0
dense_5 (Dense)	(None, 100)	6553700
dropout_9 (Dropout)	(None, 100)	0
dense_6 (Dense)	(None, 2)	202
Total parameteres: 6,735,694		
Trainable parameters: 6,735,694		
Non-trainable parameters: 0		

Table 2: Segmentation summary

Layer (type)	Output Shape	Param
input_2 (InputLayer)	(None, 512, 512, 3)	0
conv2d_25 (Conv2D)	(None, 512, 512, 64)	1792
conv2d_26 (Conv2D)	(None, 512, 512, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(None, 256, 256, 64)	0
conv2d_27 (Conv2D)	(None, 256, 256, 128)	73856
conv2d_28 (Conv2D)	(None, 256, 256, 128)	147584
max_pooling2d_6 (MaxPooling2D)	(None, 128, 128, 128)	0
conv2d_29 (Conv2D)	(None, 128, 128, 256)	295168
conv2d_30 (Conv2D)	(None, 128, 128, 256)	590080
max_pooling2d_7 (MaxPooling2D)	(None, 64, 64, 256)	0
conv2d_31 (Conv2D)	(None, 64, 64, 512)	1180160
conv2d_32 (Conv2D)	(None, 64, 64, 512)	2359808
dropout_3 (Dropout)	(None, 64, 64, 512)	0
max_pooling2d_8 (MaxPooling2D)	(None, 32, 32, 512)	0
conv2d_33 (Conv2D)	(None, 32, 32, 1024)	4719616
conv2d_34 (Conv2D)	(None, 32, 32, 1024)	9438208
dropout_4 (Dropout)	(None, 32, 32, 1024)	0
up_sampling2d_5 (UpSampling2D)	(None, 64, 64, 1024)	0
conv2d_35 (Conv2D)	(None, 64, 64, 512)	2097664
merge_5 (Merge)	(None, 64, 64, 1024)	0
Continued on next page		

Table 2 – continued from previous page

Layer (type)	Output Shape	Param
conv2d_36 (Conv2D)	(None, 64, 64, 512)	4719104
conv2d_37 (Conv2D)	(None, 64, 64, 512)	2359808
up_sampling2d_6 (UpSampling2D)	(None, 128, 128, 512)	0
conv2d_38 (Conv2D)	(None, 128, 128, 256)	524544
merge_6 (Merge)	(None, 128, 128, 512)	0
conv2d_39 (Conv2D)	(None, 128, 128, 256)	1179904
conv2d_40 (Conv2D)	(None, 128, 128, 256)	590080
up_sampling2d_7 (UpSampling2D)	(None, 256, 256, 256)	0
conv2d_41 (Conv2D)	(None, 256, 256, 128)	131200
merge_7 (Merge)	(None, 256, 256, 256)	0
conv2d_42 (Conv2D)	(None, 256, 256, 128)	295040
conv2d_43 (Conv2D)	(None, 256, 256, 128)	147584
up_sampling2d_8 (UpSampling2D)	(None, 512, 512, 128)	0
conv2d_44 (Conv2D)	(None, 512, 512, 64)	32832
merge_8 (Merge)	(None, 512, 512, 128)	0
conv2d_45 (Conv2D)	(None, 512, 512, 64)	73792
conv2d_46 (Conv2D)	(None, 512, 512, 64)	36928
conv2d_47 (Conv2D)	(None, 512, 512, 2)	1154
conv2d_48 (Conv2D)	(None, 512, 512, 1)	3
Total parameters: 31,032,837		
Trainable parameters: 31,032,837		
Non-trainable parameters: 0		

Tuning learning rate: Different approaches were tried to tune the hyperparameter. Through trial and error method, a learning rate was decided. For the training of CNN for classification, an *Adam* optimizer was used with learning rate of $1e-5$. For the training of U-Net for segmentation, an *Adam* optimizer was used with learning rate of $1e-4$. The learning rate was decided based on the training loss. The higher learning rates were causing the model to converge faster, and the model was also overfitting. The training loss was decreasing, and accuracy was increasing whereas there was very slow change in validation loss and accuracy. The smaller learning rates caused the model to converge very slowly and the difference in the optimal training loss with

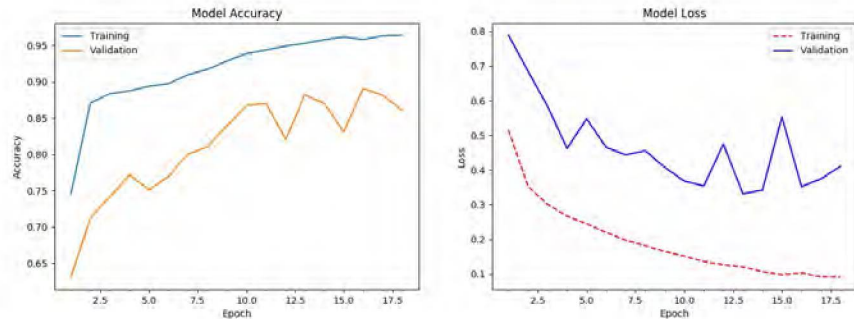


Figure 24: The accuracy and loss graph of the selected model for classification problem. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

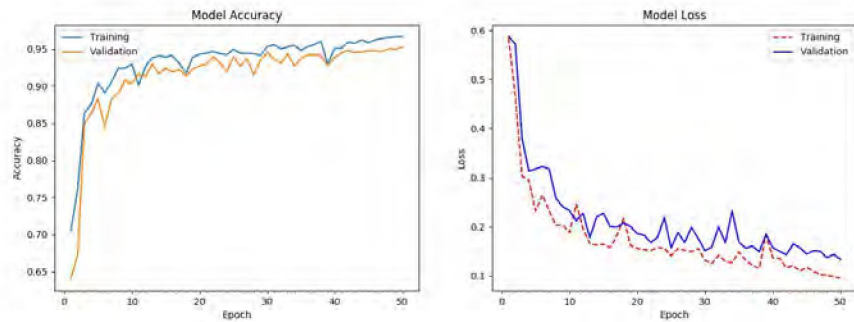


Figure 25: The accuracy and loss graph of the selected model for semantic segmentation problem. The left graph shows the accuracy vs epoch for training and validation datasets and the right graph shows the loss vs epoch for training and validation datasets.

the selected learning rate and the one with small learning rate was small.

Training was carried on until the stopping criteria has met. Initially the *number of epochs* was used as a stopping criteria but the model was not trained optimally i.e., training continued even after the accuracy started decreasing or loss started increasing. So, to avoid this situation *EarlyStopping* class in Keras was used.

With *EarlyStopping*, validation loss is monitored and if there is no change or increase in the validation loss for specified number of epochs then the training stops. Model checkpoint was added to save best weights. The model and weights were saved and used for testing the classifier and segmentation. It took 5 to 6 hours to train the model for classification and 30 to 40 mins

to train the model for segmentation.

In classification, categorical cross entropy was used as a loss function and accuracy metrics were used to evaluate the model during the training process. Categorical cross entropy was used instead of binary cross entropy, to classify two classes, because the training and validation losses were very high when using binary cross entropy. In semantic segmentation, binary cross entropy was used as loss function and accuracy metrics were used to evaluate the model during the training process.

When training the model with training and validation datasets, it is important to analyze the accuracy and loss of training versus validation datasets. For this, a graph was plotted for both classification and segmentation. Figure 24 and Figure 25 show the accuracy and loss graph of the selected model for classification and semantic segmentation respectively. For a good training model, not only the accuracy should be as high as possible and the loss should be as low as possible, but also the difference between the training and validation accuracy and loss should be as small as possible. From Figure 24 it is seen that the difference was more, so the trained model will not perform as desired whereas in Figure 25 it is seen that the difference was low and the performance of this model will be good.

7 Results

This chapter consists of the results obtained after testing the model on test datasets and the details about how the model was evaluated. Since the model and best weights were saved after the training is performed, it was easier to test the images for classification and semantic segmentation.

Confusion matrix is a table that describes the performance of a classification model on the test datasets. It consists of four different combinations of predicted and actual values. Table 3 shows the four different combinations.

Table 3: Confusion matrix used for evaluation.

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP	FP
	Negative	FN	TN

- **True Positive (TP):** When the predicted value is positive and the actual value is also positive
- **False Positive (FP):** When the predicted value is positive but the actual value is negative.
- **True Negative (TN):** When both, the predicted value and the actual value are negative.
- **False Negative (FN):** When the predicted value is negative but the actual value is positive.

This table helps to find Recall, Precision, F-score, etc. In the present study, precision and recall were used to evaluate the model on test datasets along with accuracy.

Recall: It is a measure that calculates the fraction of predicted true positive values over total actual positive values. The higher the recall the better the model. It is given by the formula:

$$Recall = \frac{TP}{TP+FN}$$

Precision: It is a measure that calculates the fraction of predicted true positive values over total predicted positive values. It is given by the formula:

$$Precision = \frac{TP}{TP+FP}$$

Accuracy: It is a measure that calculates the fraction of predicted true values over total predicted values. It is given by the formula:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

7.1 Classification

For classification, a test dataset with 96 images that are not seen by the model was used, out of which 60 images contain *Silvetia Compressa* and 36 contain other species. The images were tested using saved model with 95.82% training accuracy and 86.23% validation accuracy. While testing the images for classification, the image size was changed to the size of the input data used during training. In the present study the test image size was set to 300×300 . These images were then tested using the *predict* class in Keras.

The results of the test images were stored in a CSV file. The confusion matrix for the tested images is given in Table 4.

Table 4: Confusion matrix created after testing the classification model.

		Actual Value	
		Positive	Negative
Predicted Value	Positive	43	17
	Negative	17	19

From the above confusion matrix, it is seen that out of 60 images consisting *Silvetia Compressa* the model was able to identify 43 images correctly and failed to identify 17 images. It is also seen that there were 17 images that the model wrongly predicted as *Silvetia*.

A model with higher precisions relates to low false positive rate. Therefore, the higher the value of precision the better the model. However, precision alone is not enough to evaluate the model. So, recall was also used. A model with low recall relates to high number of false negatives. Therefore, higher the value of recall the better is the model. The tested model has a

precision measure of 0.716, recall measure of 0.716 and accuracy of 64.6% which indicated that the classifier was not accurate.

It was seen that the model had good training and validation accuracy but the accuracy on test result was not good. On checking the images that were marked as false positive and false negative it was seen that the variability in the color of *Silvetia Compressa* in the images captured in different lighting condition affected the accuracy of the model. The accuracy of the model can be increased if the model is trained using dataset containing a higher number of such images.

7.2 Semantic Segmentation

For semantic segmentation a test dataset of 50 images was used. These images were annotated manually, and the annotated images served as ground truth images for evaluation. The testing was performed using the model with training accuracy of 97.56 % and validation accuracy of 95.24 % and training loss of 0.098 and validation loss of 0.152.

Initially the test images were not processed so the images that were bright or taken using flash were not segmented well. Figures 26 and 27 show the results of the model on unprocessed test images. Figure 26 shows that for the image taken under normal lights the model segments properly. Figure 27 shows that for the images that are bright, the model does not segment properly. So, two processing techniques - *histogram equalization* and *adaptive histogram equalization (CLAHE)* were tried.



Figure 26: Result of unprocessed test image. The left image is original image, center image is the annotated image, right image is the predicted mask.

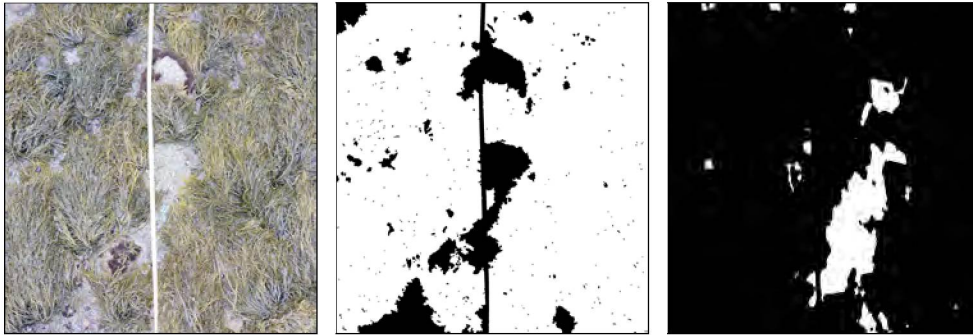


Figure 27: Result of unprocessed test image. The left image is the original image, center image is the annotated image, right image is the predicted mask.

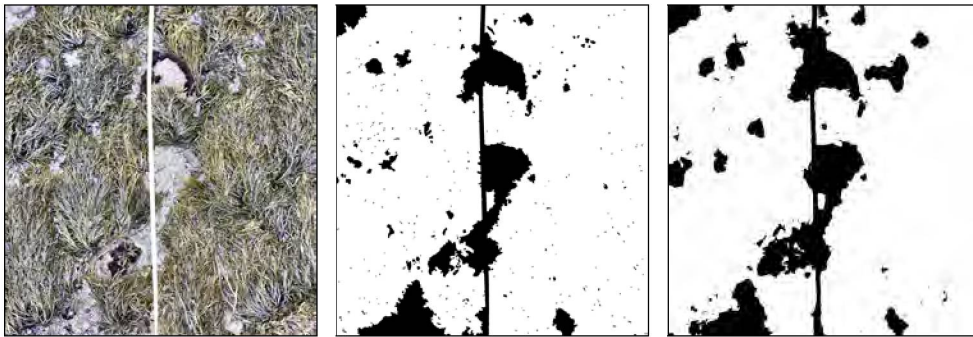


Figure 28: Result of test images after histogram equalization. The left image is the original image after histogram equalization, center image is the annotated image, right image is the predicted mask.

After trying histogram equalization, the results were good. The model was able to segment all the test images properly. Figure 28 shows the results of the model on test images after histogram equalization.

Adaptive histogram equalization (CLAHE) did a good job segmenting most of the test images however the results with histogram equalization were better. Figure 29 shows the results of the model on test images after CLAHE. On comparing the right image of Figure 28 and Figure 29 it is seen that the result after histogram equalization is similar to the ground truth image (annotated image).

There were some pixels that were incorrectly segmented as *Silvetia Compressa*, as shown in Figure 30. But the overall results of the segmented images were good. So, histogram equalization was performed on all the test images

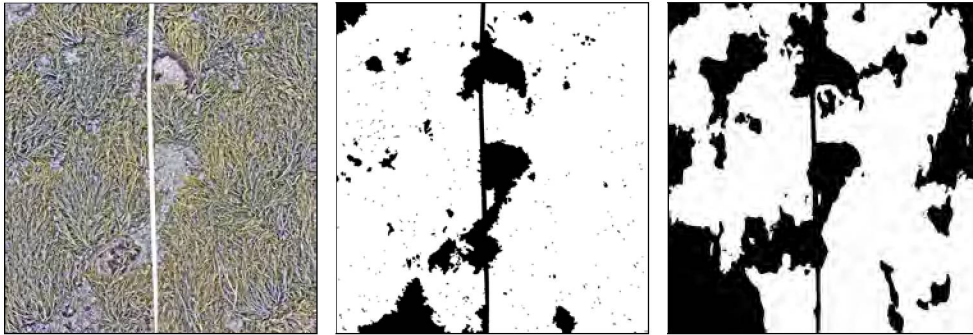


Figure 29: Result of test image after adaptive histogram equalization. The left image is the original image after adaptive histogram equalization, center image is the annotated image, right image is the predicted mask.

before testing.

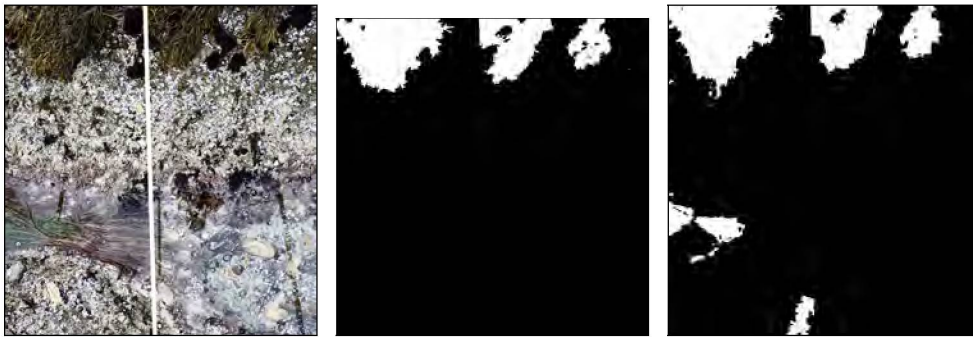


Figure 30: Result of test images segmented falsely in few regions. The left image is the original image after adaptive histogram equalization, center image is the annotated image, right image is the predicted mask. The predicted mask has some extra region segmented.

The segmented results of these pre-processed images were then evaluated by calculating the confusion matrix for each image. Using the confusion matrix, Sørensen-Dice coefficient was calculated for each image. Table 5 shows the confusion matrix created using the average count true positive, true negative, false positive and false negative pixel values of 50 test images.

Sørensen-Dice coefficient: It is a statistical metric used for comparing the similarity between two images. It is given by the formula:

$$\text{Sørensen-Dice coefficient} = \frac{2TP}{2TP+FP+FN}$$

Table 5: Confusion matrix created after testing the segmentation model.

		Actual Value	
		Positive	Negative
Predicted Value	Positive	124229.1	8827.54
	Negative	5554.06	123533.34

The average value of dice coefficients for the 50 test images is 0.9171. This value indicates that on an average the predicted mask and the annotated image are 91.71% similar. The accuracy for each image was also calculated, using the values from confusion matrix in Table 5. The average accuracy is 94.52 % which is close to the validation accuracy.

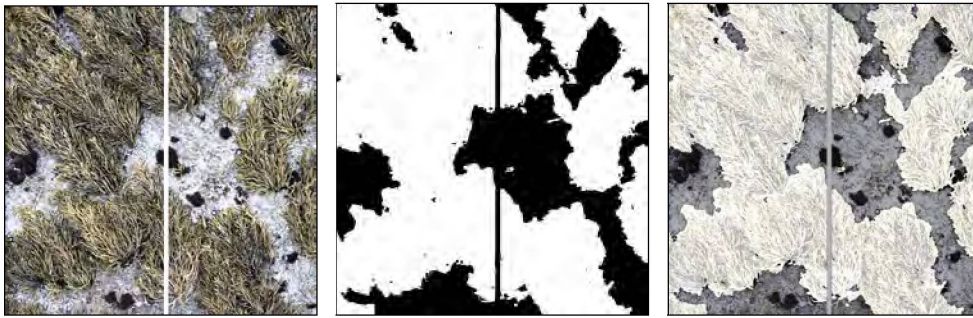


Figure 31: The predicted mask overlapping the original image for researchers to analyze. The left image is original test image, the center image is the predicted mask and the right image is the overlapped image.

The goal of the present study was to perform ecological monitoring on the assemblage of *Silvetia Compressa*. So, a CSV file was created that stores the percentage of *Silvetia Compressa* present in each predicted mask image. Also, an image with predicted mask overlapping the original image was created so that the researchers monitoring the assemblage can see the falsely segmented and unsegmented areas in the image. Figure 31 shows the example of predicted mask overlapping the original image.

8 Conclusion and Future Work

This chapter summarizes the present study and provides suggestions for future improvements.

Conclusion: The major motivation of undertaking this work was to gain understanding of designing, implementing and evaluating a convolutional neural network. This was defined as an image classification and semantic segmentation problem, to identify a specific algae species - *Silvetia Compressa*. The present study investigated the image classification problems by designing a CNN. Then implementing it using Keras and evaluating it with different metrics. It also investigated the problem of semantic segmentation by fine-tuning an existing model and achieved a good accuracy.

By investigating these problems, an in-depth understanding of various machine learning techniques was achieved. It helped to understand how to tune hyperparameters, how addition of dropout layers can help reducing overfitting, how selecting a proper loss function can affect the accuracy and loss of the model. The main goal was to reduce the time required to analyze the images for ecological monitoring and to make it less prone to human error. It was achieved by creating an automated tool to find the assemblage of *Silvetia Compressa* by implementing semantic segmentation.

Future Work: The image classification model created was not as accurate as it should be. So an improvement can be made to the image classification model by using more training data, preprocessing it well for proper identification by the machine. Improving the image quality can help the machine to pick features specific to a particular species and not confusing two different species as one.

The semantic segmentation model was able to provide good results. But there are a few images where it segments few pixels that do not belong to *Silvetia Compressa*. Figure 30 shows one such example. Improvements in training and preprocessing can be done to achieve more accurate results.

As mentioned in Chapter 4 there are other species that are present in the two rocky intertidal zones of Santa Rosa Island. The created models should be tested on the new species and a multiclass classification and segmentation model should be implemented to identify these species.

References

- [1] Nigel Gilles Yoccoz. *Ecological Monitoring*. American Cancer Society, 2012.
- [2] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2009.
- [3] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.
- [4] Maureen Caudill. Neural Networks Primer, Part I. *AI Expert*, 2(12):46–52, December 1987.
- [5] Andrew Ng. Machine Learning - Online Course. <https://www.coursera.org/lecture/machine-learning/the-problem-of-overfitting-ACpTQ>. Accessed: Dec-2017.
- [6] Hafidz Zulkifli. Understanding Learning Rates and How It Improves Performance in Deep Learning. <https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10>, January 2018. Accessed: Feb-2018.
- [7] Andrej Karpathy. Convolutional Neural Networks for Visual Recognition (Course Notes). <http://cs231n.github.io/neural-networks-3/#loss>. Accessed: Nov-2017.
- [8] Andrew Ng. Machine Learning - Online Course. <https://www.coursera.org/lecture/machine-learning/gradient-descent-8SpIM>. Accessed: Dec-2017.
- [9] Petar Veličković. Deep Learning for Complete Beginners: Convolutional Neural Networks with Keras. <https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html>, March 2017. Accessed: Nov-2017.
- [10] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [12] Andrej Karpathy. Convolutional Neural Networks for Visual Recognition (Course Notes). <http://cs231n.github.io/convolutional-networks/#pool>. Accessed: Nov-2017.
- [13] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [14] Peter Bhlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated, First edition, 2011.
- [15] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.*, pages 248–255. IEEE, 2009.
- [18] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

- [20] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.
- [21] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.
- [22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in Network. *arXiv preprint arXiv:1312.4400*, 2013.
- [23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going Deeper with Convolutions. *Cvpr*, 2015.
- [24] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [25] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. *arXiv preprint arXiv:1709.01507*, 7, 2017.
- [26] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008*, pages 1–8. IEEE, 2008.
- [27] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011*, pages 1297–1304. Ieee, 2011.
- [28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [29] 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.

- [30] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.
- [31] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [32] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.
- [33] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [34] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable are Features in Deep Neural Networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press.
- [35] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught Learning: Transfer Learning from Unlabeled Data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 759–766. ACM, 2007.
- [36] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer Learning for Image Classification with Sparse Prototype Representations. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on 2008*, pages 1–8. IEEE, 2008.
- [37] Clément Douarre, Richard Schielein, Carole Frindel, Stefan Gerth, and David Rousseau. Transfer Learning from Synthetic Data Applied to Soil-Root Segmentation in X-Ray Tomography Images. *Journal of Imaging*, 4(5):65, 2018.
- [38] F. Chollet. Keras. <https://github.com/keras-team/keras>, 2015.

- [39] Adit Deshpande. A Beginner's Guide to Understanding Convolutional Neural Networks. <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>, July 2016. Accessed: Oct-2017.
- [40] Sebastian Ruder. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747*, 2016.