



**POLITECNICO
DI TORINO**

POLITECNICO DI TORINO

Master Degree course in Mechatronics Engineering

Master Degree Thesis

Feature extraction using satellite images

Supervisors

Prof. Carlo NOVARA

Dr. Lorenzo FERUGLIO

Candidate

Omid TOUTIAN ESFAHANI

ACADEMIC YEAR 2018-2019

Acknowledgements

I would first like to thank my thesis advisor Professor Carlo Novara of the Department of Electronics and Telecommunications (DET) at Politecnico di Torino. I am very grateful for all of the things that you have done for me and I can not express how much I am thankful to you. The door to Prof. Novara office was always open whenever I ran into a trouble or had a question about my research.

I would also like to thank the experts in AIKO for giving me the chance to be a part of your team. Cooperating with talented young people working at AIKO was an honour for me. I learned a lot from you and it was one of best experiences in my life working with you on challenging problems that we faced during this period. I would like to express my gratitude to Dr. Lorenzo Feruglio for his unconditional support. You are my all time favourite entrepreneur and I wish you all the best for your future endeavours. Without your help, this thesis would not have been successful.

Finally, I must express my very profound gratitude to my parents and to all of my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Omid Toutian Esfahani

Abstract

Remote Sensing Technology (RST) mainly focuses on the gathering information about the Earth's surface and atmosphere using sensors onboard airborne or space-borne platforms. RST has been widely used in ground mapping, resource regulation, environmental protection, urban planning, geological research, disaster relief and emergency, military reconnaissance, and many other fields. RST has enabled us to have access to a large amount of data at a relatively low cost and it would be feasible to perform computational algorithms on gathered image data to extract useful information.

Availability, accessibility to data, the evolution of Artificial Intelligence and largely increased computing power has enabled the scientists to solve many of today's world problems which were considered unsolvable before. This improvement is not only limited to a specific field of science but also proved to be useful in many different fields, varying from economics and finance to robotic, space missions and image processing.

Artificial Intelligence, Machine Learning and Deep Learning solutions are being progressively researched and implemented in the field of space. Machine Learning and Deep Learning algorithms are used to analyse and process of high resolution satellite images to acquire an explicit representation. Machine Learning is considered as an priceless tool to analyse data gathered from remote sensing and telemetry. Beyond Earth Observation, for precise data analysis, Machine Learning is used to process immense amount data from deep space missions. Furthermore, Using AI would lead to reduce the workload of ground stations. By applying AI in space missions, the need for human intervention would decrease and in this manner it would cause to save a lot of time, money and effort.

Amongst the applications of RST, object detection plays a key role in detecting and identifying target objects. Target object detection and identification are usually achieved using a combination of signal and image processing techniques and statistical models.

Recent advances in Deep Learning architectures have shown promising results over statistical counterparts in target object detection and identification. Although such learning architectures are heavily dependent on computing resources, they are easy to use compared to sophisticated statistical models. Further, Deep Learning architectures are also able to render feature engineering as a part of their learning process which makes them extremely powerful in target object detection and identification process.

Nowadays object detection methods are maturing very rapidly and thanks to deep learning, new algorithms and models keep on outperforming the previous ones. For instance, methods such as SSD, YOLO, R-CNN, Fast R-CNN, Faster R-CNN are among the state of art models and they are able to deliver high accuracy and reasonable processing speed for a wide variety of applications. Although several breakthroughs have been witnessed in recent years in object detection, still there is room to improve.

Object detection algorithms have the potential to be applied to satellite images to detect the various object. One important application of object detection on satellite images is ship detection due to increasing of shipping traffic in recent years. There are more ships used in today's world and this phenomenon leads to increase the chances of breaches at sea like ship accidents, piracy, illegal fishing, drug trafficking, and illegal

cargo movement. This has urged many organizations such as environmental protection agencies, insurance companies, and national government authorities, to have a closer look and pay more attention over the open seas.

The main focus of this thesis is on ship detection based on satellite images by employing Deep Learning algorithms. For the specific case of ship detection, one of the biggest challenges is the lack of appropriate dataset. Therefore, a synthetic dataset is created to overcome the issue. For object detection, various methods are implemented and compared to achieve the best result.

Contents

1	Introduction	5
1.1	Artificial Intelligence	5
1.2	Machine Learning	6
1.2.1	Supervised learning	8
1.2.2	Unsupervised learning	8
1.2.3	Reinforcement learning	9
1.3	Artificial Neural Networks (ANN)	9
1.3.1	Feed-forward Neural Networks	13
1.3.2	Recurrent Neural Networks (RNN)	14
1.3.3	Convolutional Neural Network	14
1.4	Object detection	16
1.5	History of image processing and its relation to Deep Learning and CNN	18
1.6	Remote sensing and its importance	19
1.6.1	Related work	22
1.7	Thesis objectives and outline	23
2	State-of-the-art models for object detection	25
2.1	Region proposal based methods	26
2.1.1	R-CNN	26
2.1.2	Fast R-CNN	27
2.1.3	Faster R-CNN	29
2.2	Regression/Classification based methods	30
2.2.1	YOLO	30
2.2.2	Single Shot MultiBox Detector (SSD)	31
2.3	Some precision metrics definitions	32
2.3.1	True Positive, False Positive, False Negative and True Negative	32
2.3.2	Intersection Over Unit (IoU)	33
2.3.3	Precision and recall	33
2.3.4	Mean Average Precision (mAP)	33
3	Problem definition	35
3.1	Modern maritime piracy	35
3.2	Anti-piracy measures	35
3.3	Sentinel-2 mission	36

3.4	Synthetic dataset	37
4	Use Case Results	41
4.1	Google Tensorflow	41
4.2	Object detection by Tensorflow	43
4.3	Object detection results	59
5	Conclusions	65
5.1	Further work	65
	Bibliography	67

Chapter 1

Introduction

After passing the Industrial Revolution, nowadays human-beings are finding themselves in a new era called Information Age. The main highlight of this period is the availability of an immense amount of data to most people by the means of computer technology. [1] Likewise, the capacity of information storage has increased dramatically, from 2.6 exabytes¹ in 1986 to 296 exabytes¹ in 2007. It is approximately equivalent to 539 MB per person in 1986 to 44.5 GB of data per person in 2007. [2] Although in this period, we have suffered from two Artificial Intelligence (AI) winters from 1974 to 1980 and from 1987 to 1993 [3], but at the moment, with good investment in this field, the trend of AI is moving forward and it is one of the most growing fields of science if we do not consider it as the most growing field! It is worth mentioning that scientists believe there is a 50 percent chance that in around 45 years, Artificial Intelligence would be able to surpass human-beings in nearly every possible task. [4]

Availability, accessibility to data, the evolution of Artificial Intelligence and largely increased computing power has enabled the scientists to solve many of today's world problems which were considered unsolvable before. This improvement is not only limited to a specific field of science but also proved to be useful in many different fields, varying from economics and finance to robotic and image processing.

The goal of this thesis is to employ a state of the art Deep Learning algorithm to detect ships from satellite images. Since understanding, Deep Learning algorithm needs a foundation of Artificial Intelligence and Machine Learning, in the following sections an introduction to these concepts will be presented.

1.1 Artificial Intelligence

Artificial Intelligence (AI) or so-called Computational Intelligence (CI) is defined as the field of science which concentrates on studying of the intelligent agent: the system which acts appropriately based on the conditions and its purpose. It would be able to affect and change the environment and goals. The system learns from its experiences and makes

¹10¹⁸

the right decisions given some limitations and finite computations. [5]

John McCarthy who is known as the godfather of AI defined it as:

The science and engineering of making intelligent machines. Especially intelligent computer programs. [6]

In today's world, the huge progress in the field of AI has led to a massive impact on our day-to-day lives. Virtual Personal Assistance such as Siri, Alexa, and Google Assistant are able to understand the human's natural language and perform the tasks that they are asked to. Using navigator applications like Google map, it is possible to find the best route to the destination based on the real-time traffic. Spam detection is another field that AI is proving to be useful in recent years. All of the technologies mentioned earlier could be classified as Artificial Narrow Intelligence (ANI). As the name suggests, these types of technologies are able to perform a limited and narrow task well and they are not aiming to perform a full range of human cognitive skills. Meanwhile, we are experiencing fundamental improvement towards human-level Artificial General Intelligence (AGI), also known as strong AI or full AI. By definition, Strong AI is a type of intelligence of a machine which is able to successfully perform any type of intellectual task which can be done by any human-being such as learning, planning, making decisions under uncertain situations, using natural language to communicate, manipulating others and reprogramming itself. Likewise, the word "strong AI" is used by academics to indicate a machine with the ability to feel consciousness. [7] Another concept which is aiming towards the same goal is "ultraintelligent machine" which was first defined by I.J. Good in 1965:

Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any man however clever. Since the design of machines is one of these intellectual activities, an ultraintelligent machine could design even better machines; there would then unquestionably be an "intelligence explosion", and the intelligence of man would be left far behind. Thus the first ultraintelligent machine is the last invention that man need ever make, provided that the machine is docile enough to tell us how to keep it under control. [8]

As mentioned by Good, once we could create an AI which can improve itself, it will open up an indefinite recursive loop of self-improvement that could lead to a so-called intelligent explosion. Time to reach this intelligent explosion could vary from many years to a single day.

Although Artificial Intelligence has many branches, one of the most important subsets of AI is Machine Learning. Thus, all Machine Learning algorithms are part of AI but not vice-versa. In today's world, Machine Learning is playing a core role in the journey proceeding to Artificial General Intelligence (AGI) and meanwhile, it is improving every industry and has an immense effect on our day-to-day lives.

1.2 Machine Learning

Machine Learning is a task of designing an efficient and accurate algorithm which could generate precise predictions on unseen data and generalize from its experience. [9] In 1959,

A.L.Samuel, one of the Machine Learning pioneers of his time, defined Machine Learning (ML) as a field of study which provides the computers the ability to learn without being explicitly programmed for the desired task. [10] Also, Tom M. Mitchell stated Machine Learning as:

A computer program is said to learn from experience E with some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E . [11]

To clarify the above definition, consider a board game like Chess. Experience E refers to the experience that the machine gained by playing and observing many games of chess. T is the task of playing chess and P indicates the win probability of the machine to win the next game. The word "experience" in the above definition usually indicates the good amount of data. Hence, the availability of data is a keystone for every Machine Learning problem. This data is usually referred as "dataset" by Machine Learning experts and it is divided into two parts called training data and test data. Since Machine Learning has a close relation with optimization, most of the Machine Learning problems are formulated to minimize a set of loss function on training data by means of changing the model parameters. Loss function denotes the difference between the predictions of the algorithm being trained and the true instances of the problem. [12] In the next step, the model is tested on the unseen test data to evaluate its performance and the level of generalization.

Machine Learning consists of different types of algorithms. The selection of the right algorithm depends on the type of input and output data and also the type of problem that is intended to solve.

Broadly speaking, Machine Learning could be classified into three sub-categories: Supervised learning, Unsupervised learning and Reinforcement learning.

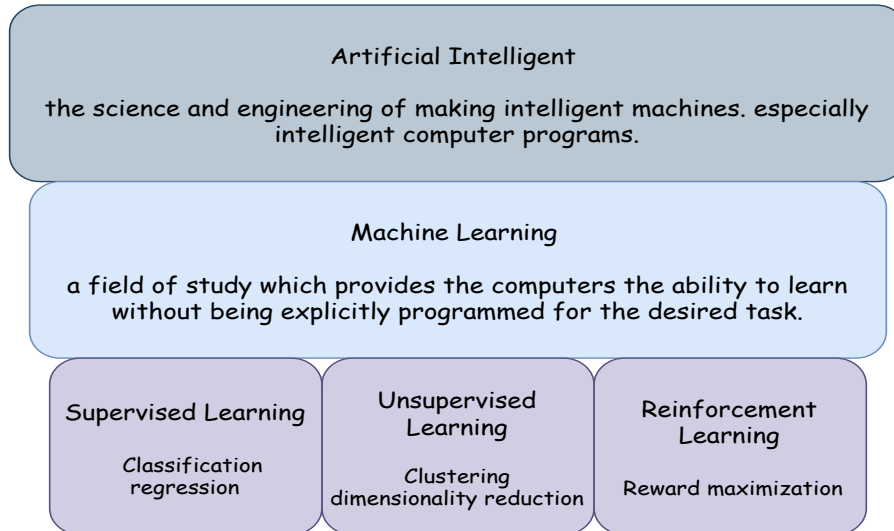


Figure 1.1: Artificial Intelligent hierarchy

1.2.1 Supervised learning

Supervised learning is a subcategory of Machine Learning which builds a mathematical model based on a data-set that contains a set of training data. In the case of supervised learning, each member in training data consists of at least one input and the desired output. The model tries to minimize a lost function to find the best mapping between input and output data. [9]

Each supervised learning problem could be categorized as a classification problem or a regression problem. [13] In the case of classification, outputs provided to the model are discrete and limited to a set of known values. For instance, consider a classifier which as input takes a set of images and as output, it should be able to detect if that image is a dog or a cat. On the other hand, in the regression problem, the algorithm should predict the continuous values as the output. An example would be, estimating the price of a house based on its area, numbers of rooms and its location.

There exist a wide range of algorithms to solve a supervised learning problem. some of the most important are:

- Support Vector Machines (SVM)
- K-nearest Neighbour
- Linear regression
- Logistic regression
- Naive Bayes
- Decision trees
- Neural Networks

1.2.2 Unsupervised learning

In contrast with Supervised learning, no specific output is given to an unsupervised learning algorithm. Unsupervised learning algorithms aim to find the structures and relationships in the provided data. Therefore, unsupervised learning algorithms are searching for groups or clusters of data which share the common characteristics. Another application of unsupervised learning is dimensionality reduction in which the algorithm tries to reduce the complexity of data while maintains its structure and quality.

Some of the most common unsupervised algorithms are:

- Clustering:
 - k-means clustering
 - Hierarchical clustering
 - Neural Networks

- Dimensionality reduction:
 - Principal component analysis (PCA)
 - Singular value decomposition (SVD)

1.2.3 Reinforcement learning

The basic concept of reinforcement learning is that feedback is provided to the agent and it indicates how good or bad was the decision that it took. In the context of reinforcement learning, This feedback is usually represented as a reward or reinforcement. Based on the reward given to the agent, it is able to understand the right decision in a specific situation. Reinforcement learning methods could be seen as a type of trial-and-error approach which aims to maximize the long-term reward. Because of the generality of this topic, it is studied in many other fields, such as operational research, control theory, game theory and etc. In reinforcement learning, it is assumed that the exact mathematical representation of the model and environment is unknown and it is used when the exact models are infeasible. [14] Nowadays, reinforcement learning algorithms are used in a variety of applications including autonomous vehicles and when an agent is trained to play a game against a human opponent. Figure 1.1 shows the summery of Artificial Intelligence, Machine Learning and sub-categories of Machine Learning as described before.

1.3 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) are frameworks motivated by the biological natural neural systems that comprise animal brains. The neural network itself is not considered as an algorithm, yet rather a structured framework for some Machine Learning algorithms to cooperate and process complex information. Such frameworks "learn" to perform assignments by seeing enough amount of examples, without being programmed to follow any explicit guidelines. For example in image recognition, the artificial neural network is used to identify the images of dogs by showing the manually labeled images of dogs to the network. This task does not ask the network to have any type of prior knowledge about dogs such as their visual representations.

As it is shown in 1.2, biological neuron is composed of three basic elements: dendrites, cell body and axon. Dendrites are responsible to receive the signals and send them to the cell body. When signals arrive at the cell body, they are added together and if this summation is greater than a certain threshold, then they are transmitted through the axon to the next neuron. This functionality was the inspiration for the scientists to develop an Artificial Neural Network architecture. The schematic of this network is shown in 1.5. In this figure, the network has n inputs. In neural networks, each input channel is associated with a specific weight which represents the "importance" of that input in comparison with other inputs. After weighing the inputs, function g is acting on each of them. g in most cases is the summation function. In the next step a non-linear primitive function $f(x)$, called activation function, will be calculated and the output will be produced.

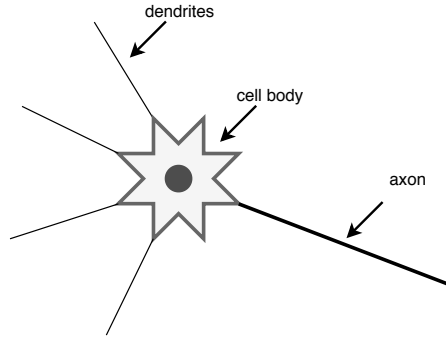


Figure 1.2: Biological neuron

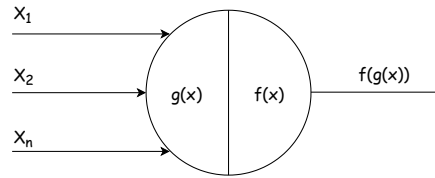


Figure 1.3: Artificial Neural Network

Activation functions are used to allow a seamless transition as input values alter. Thus small change in inputs causes a small change in outputs. Most used activations functions are:

- Rectified Linear Unit (Relu)
- Sigmoid
- Tanh

in figure (will be added) the above mentioned functions are plotted.

Artificial Neural Networks could be classified into two sub-categories based on the depth of connected networks layers:

- Shallow neural networks
- Deep neural networks

Deep Neural Networks (DNN)

Deep Learning neural network is an artificial neural network architecture with multiple layers between inputs and outputs. On the other side, in shallow networks, there is only one neural layer between inputs and outputs. The comparison of two architectures is shown in [1.4](#).

Deep Learning is defined as a class of Machine Learning algorithms which: [\[15\]](#)

- Is produced by cascading multiple layers of non-linear processing units.

- Could be used in both supervised and unsupervised learning.
- Learn various degrees of representation that match to different levels of abstraction.

In Deep Learning networks, each level learns to translate its input data into a slightly more abstract and composite representation. For example in case of an image recognition, the input is a matrix of pixels which could be three dimensions if we are considering color images or two dimensions for black and white images; the first layer might reckon the pixels and understand edges; the second layer may compose and encode combination of edges; the next layer may encode more meaningful objects such as eyes; and the last layer would be able to realize that the image contains a face. significantly, a Deep Learning algorithm could learn on its own which features are better to be placed in which level.

At the first phase, the DNN initialize its "weights" to some random numerical values. The weights and inputs are multiplied together and as an output, the network outputs a numerical value which is usually between 0 and 1. After having this output, the algorithm compares the computed output to the true output which was given to the network. By using the defined loss function, a value for the error will be computed. Afterward, Stochastic Gradient Descent (SGD) algorithm tries to adjust the weights of each layer. In SGD, some input vector for a small portion of instances of data is shown to the network and then outputs and the errors are computed. Next, by calculating the average gradient for those examples, weights are tuned consequently. In each loop of this procedure, we are moving one step closer to the optimal minimum of the loss function. The length of the step is determined by the hyper-parameter called "learning rate". Finding the optimal learning rate is yet a challenging problem in Machine Learning field and it is usually done by the trial-error approach. The mentioned process will be repeated until the defined loss function does not decrease anymore and starts to reach a constant number with a small tolerance. This method is called stochastic due to the fact that each set of instances which are feed to the network provide a noisy estimate of the average gradient and not the exact gradient because they are selected randomly.

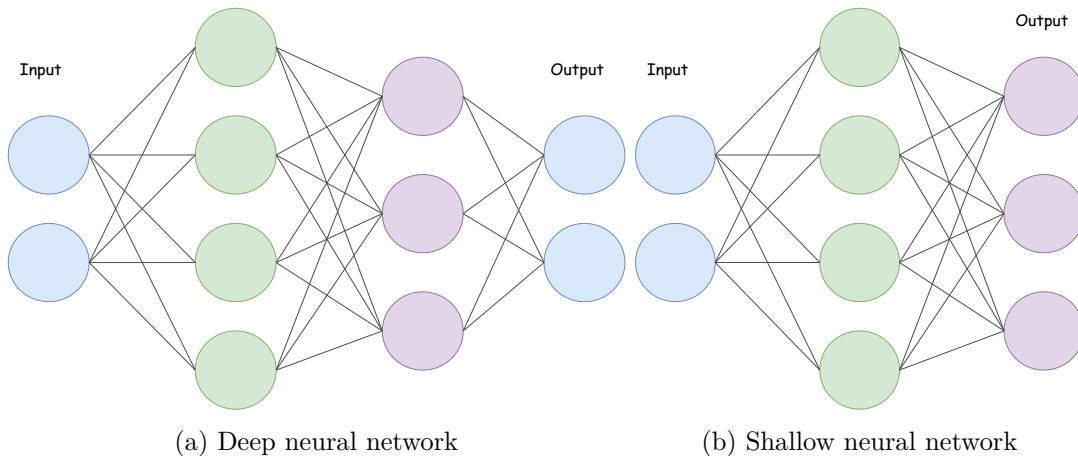


Figure 1.4: Comparison of two artificial neural network architectures

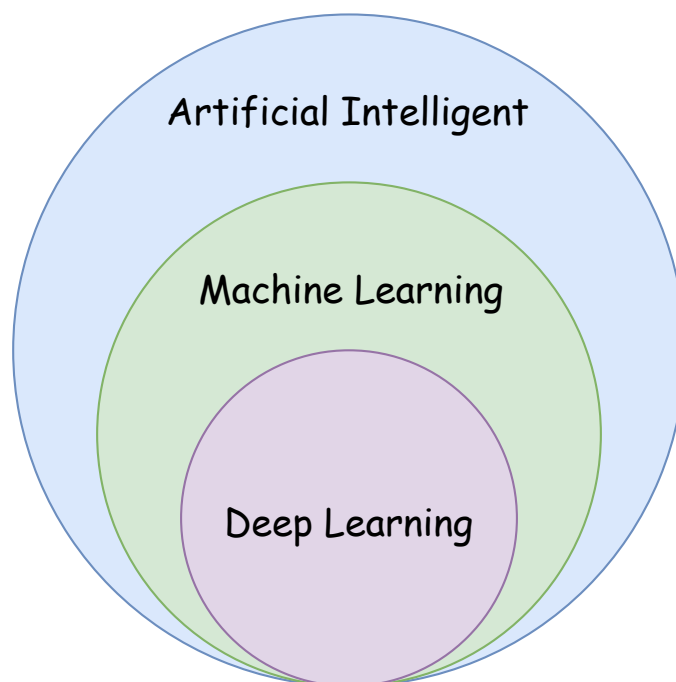


Figure 1.5: Deep Learning as a subset of Machine Learning

This elementary technique commonly finds a good set of weights in less time comparing to other optimization techniques. [16]

The methods of computing the gradient to correct the weights to train the network are called backpropagation. The concept of backpropagation was discovered in the mid-1970s by P.Werbus. [17] The concept of backpropagation which computes the gradient of a loss function with respect to the weights of a multi-layer neural network is nothing more than a simple usage of the chain rule for derivatives in algebra. As the name "backpropagation" indicates, The crucial factor in this theorem is that the derivative of the loss function is computed by moving backward from gradient with respect to the output. Backpropagation is applied throughout each layer of an artificial neural network from the output to the input corrects the weights from the very end of the network and uses the If after this first attempt the network was not able correctly to understand a particular pattern, an algorithm would adjust the weights.

Like any other algorithm, deep neural networks have their own drawbacks. Two of the most important issues are:

- Overfitting
- Computation time

In Machine Learning, overfitting occurs when the trained network performs well on the training data and poorly on the test data. In other words, training loss is comparably less than the validation loss. It could be said that instead of generalizing the problem which

is the main goal of the network, the algorithm memorized the inputs which were given to it. To overcome this problem, the main solution is to provide the network with more data so it could generalize better by receiving more samples. Also when the input data is in form of images, it is possible to use data augmentation methods such as cropping, rotating, zooming and etc. to increase the number of available samples. Another solution is to use regularization methods to limit the numerical value of network weights. [18] Alternatively, a dropout layer can be added to the network. By utilizing this layer, some nodes in the hidden layer are arbitrarily dropped during the training which helps the network to generalize better. [19]

Due to the vast number of parameters in the complex neural network architectures, performing training usually takes a considerable amount of time. To solve this problem techniques such as batching could be used. In batching, the gradient is computed on a cluster of training data at once rather than individual training samples. [20] Besides, in recent years utilizing multi-core GPUs provides an immense improvement in reducing the training time, Due to the fact that they are optimized for matrix and vector computations. It is evident that matrix computations are the keystone of the neural networks by their nature. [21]

The most important architectures of Deep Neural Networks (DNN) are:

- Feed-forward Neural Networks
- Recurrent Neural Networks (RNN)
- Convolution Neural Networks (CNN)

1.3.1 Feed-forward Neural Networks

Deep neural networks, in most cases, are Feed-forward Networks. In this type of network data flows from the input layers to the output layer without looping back. Therefore the information never reaches a node twice. Feed Forward Neural Networks, are considered as memory less networks and thus, they are not good in predicting the future behaviour of a phenomenon. Due to the fact that a Feed-forward Network is able to only receive the current input, it has no notion of time. They simply can not remember anything about the past, except their training.

As mentioned before, the leftmost layer in this network architecture is called the input layer and the rightmost layer is called output layer. The middle layer is called a hidden layer, because the neurons in this layer are neither inputs nor outputs. Each neuron has its own activation function which helps to grasp the non-linearities presented in the dataset.

The design of the input and output layers of Feed-forward Neural Network is straightforward, the design of the hidden layers could be a challenging task. In practice, in most of the times, trial and error method is used to find out the best network architecture which has the best results and lowest loss. Feed-forward Neural Networks are only able to map one input to one output at a time.

1.3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) are considered as an important and robust type of Neural Networks due to their ability to process the information received from the past. Having an internal memory enables them to remember important information about the input that they have received.

In a RNN, the information cycles back through the network and creates a type of feedback. This network architecture has the ability to take into account the current input and also the inputs it received in previous steps. Due to this characteristic, they are good choices to process and analyse the sequential data such as time series, text, natural language, speech, financial data, audio, video and etc. The difference in the flow of information between Feed-forward Networks and RNN is illustrated in figure 1.6.

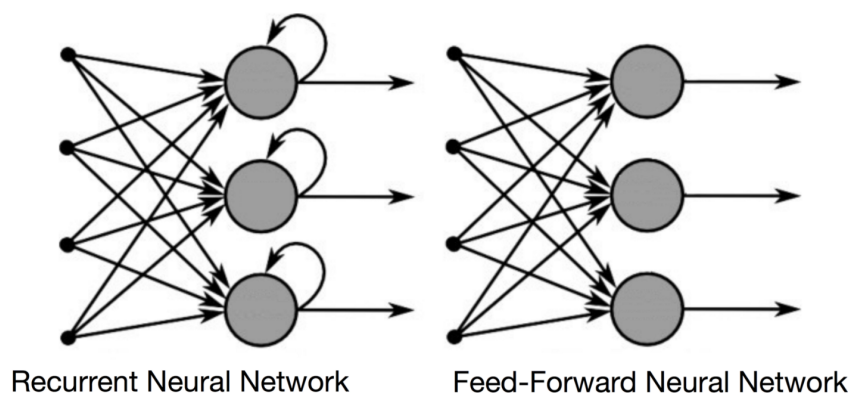


Figure 1.6: Feed-forward Networks and RNN architectures comparison

1.3.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is considered as a subcategory of deep neural networks and it is widely used in image processing and natural language processing applications. CNN is composed of four main building blocks:

- Convolution layer
- Non-linear activation function
- Pooling layer
- Fully connected layer

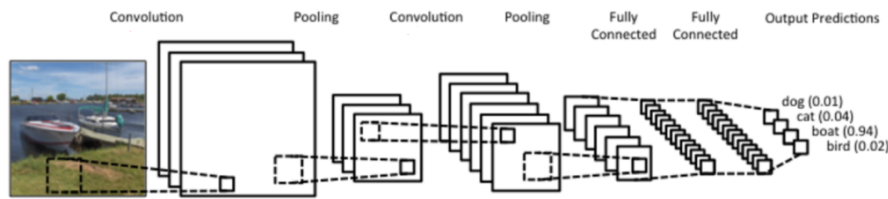


Figure 1.7: Convolutional Neural Network structure

Convolution layer

As the name "Convolution" suggests, the main functionality of this layer is to extract features from the input layers. If we consider images as a matrix of pixel values, they could be feed into the network as the input. Convolution layer is a small size matrix which will be slid over the input image for every position. An element-wise multiplication between two matrices is performed and they are added together to form the output matrix. In CNN terminology, the small matrix which slides through the image is called "filter" or "kernel". The output matrix created by sliding the kernel over the original image is referred to as "feature map". In practice, a convolutional neural network learns the optimized values of kernels throughout the training phase.

The size of the feature map is determined by three characteristics of the kernel:

- **Depth:** Depth defines how many convolution layers are put on top of each other.
- **Stride:** Stride is the number of pixels by which kernel is slid over the input matrix. For example, when stride is defined as 2, it means that the kernel is sliding through the input by two pixels at each step.
- **Zero-padding:** Since convolution operation decreases the dimension of the input matrix, zeros could be added to the input matrix to solve this dimensionality reduction. This process is called zero-padding.

Activation function

As mentioned in section 1.3, activations are used to allow a seamless transition from inputs to outputs. Additionally, most of the real world event is non-linear by their nature and adding a non-linearity factor to the network proved to be useful and helps the network to generalize better. It is worth mentioning that the non-linear function operates in a bit-wise matter.

Pooling layer

Pooling layer (also called downsampling or subsampling) is used to decrease the dimensionality and complexity of each feature map while keeping the most important pieces of information. In each pooling layer, it is necessary to define a neighborhood in which some function will be performed to reduce complexity. Pooling has many different types:

- Max pooling:
Takes the maximum value from a neighbourhood in a matrix and pass it to the next layer.
- Average:
Calculates the average in a neighbourhood of a matrix and pass it to the next layer.
- Sum:
Performs summation in a neighbourhood and sends the value to the next layer.

The effect of two types of pooling layers are shown in figure 1.8.

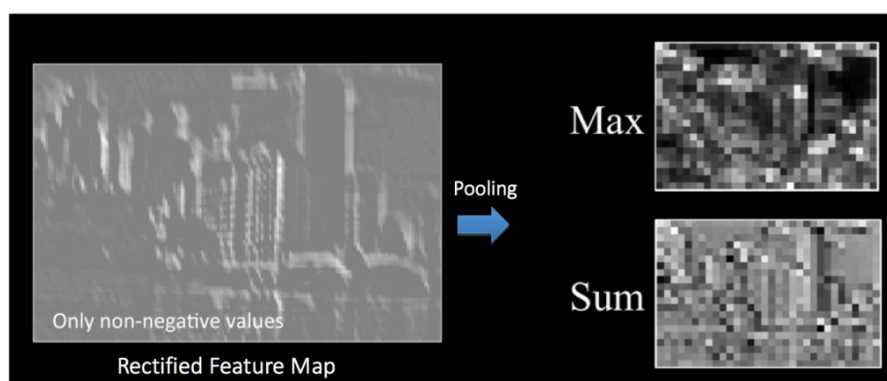


Figure 1.8: Max pooling and sum pooling on feature map

Fully connected layer

Fully connected layer is similar to the traditional neural network. It is a combination of neural nodes fully connected to each other. Adding these layer enables the network to consider the combinations of the feature maps created in previous layers and improves the performance of the network. As shown in figure 1.7 the output of the network indicates that which picture belongs to which class. Also, it is possible to generate a float number as output. These problems are called classification and regression respectively. The architecture of this network only enables us to have a unique output from the network whether it is a real or float and since our purpose is object detection from the images, this simple architecture does not fit for the object detection problem and further investigation is needed to adopt this network to fit the object detection problem.

1.4 Object detection

Object detection is a technology associated with computer vision and image processing which aims to detect instances of objects and their corresponding location in digital images and videos. Object detection methods fall into two main categories, Machine

Learning based approach and Deep Learning based approach. In Machine Learning approach, after implementing the techniques below it is necessary to use a type of classical Machine Learning approach such as SVM to make the classification. On the other side, solutions based on Deep Learning are able to do the end-to-end object detection without the requirement to define features. Deep Learning methods are utilizing convolutional neural networks as a backbone.

- Machine Learning approaches:
 - Viola-Jones object detection framework based on Haar features
 - Scale-invariant feature transform (SIFT)
 - Histogram of oriented gradients (HOG) features
- Deep Learning approaches:
 - Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
 - Single Shot MultiBox Detector (SSD)
 - You Only Look Once (YOLO)

Further explanation of Deep Learning approaches in object detection will be given in the next chapter.



Figure 1.9: An example of object detection with two classes

1.5 History of image processing and its relation to Deep Learning and CNN

The amount of visual data exploded in recent years. In 2015 Cisco estimated that by 2017, 80 percent of all traffic on the net will be video. Moreover, every second, 5 hours of video is uploaded on Youtube. It is critical to developing an algorithm capable of visualizing and understanding those data. The problem is that this type of data is hard to understand. Computer vision is an interdisciplinary field and touches many areas of science. One of the first cameras created by humans was made by Leonardo da Vinci in the 16th century. Computer vision history starts around 1960s. Block world is the paper published by Larry Robert as his Ph.D. thesis and is considered to be the first publication in the computer vision field. He tried to translate the visual world into simplified structures to recognize them and recreate them. In 1966 there was an MIT summer project called summer vision project and the goal was an attempt to use summer workers to create a part of a visual system. David Marr was an MIT vision scientist who has written an influential book around the early 80s about his perspective and ideas about vision and how we should process them. According to him, in order to reach the full 3D representation of the image it is needed to go to several steps: [22]

- Primal Sketch: In this step mostly edges, bars, curves and boundaries are represented. This step is inspired by neuroscientist idea which indicates early stages of visual processing is responsible to detect simple structures such as edges and curves
- $2\frac{1}{2}$ -D Sketch: In this step the surfaces and depth of the image should be extracted
- 3-D Model: Based on the previous steps, we extract the full 3-D model out of the picture

This way of thinking has dominated computer vision for several decades. In 1987, David Lowe tried to detect Razors by constructing lines and edges. [23] Up to this point, those researches remained as just examples and not breakthroughs and they are not considered as practical solutions for real-world problems.

The general focus changed from this point on from image recognition to image segmentation. It was believed that image segmentation is an easier problem to solve. Image segmentation is the task of grouping the pixels into meaningful parts and areas. One of the first attempts on image segmentation was done using graph theory algorithm by Malik in 1997. [24]

At the beginning of 2000, statistical Machine Learning approaches like SVM and boosting started to gain momentum. As a result, Viola and Jones developed a real-time face detection framework using the AdaBoost learning algorithm. [25] David Lowe developed a technique called Scale Invariant Feature Transform (SIFT) to detect objects. The main idea was that to match an object to the same object in a different image, only some parts of image or features are needed and they remain constant. [26]

Using the same idea of David Lowe which was extracting features from the images and concentrate on those features instead of the whole image was the inspiration for most of the computer vision researches. In 2005, Histogram of Oriented Gradient (HOG)

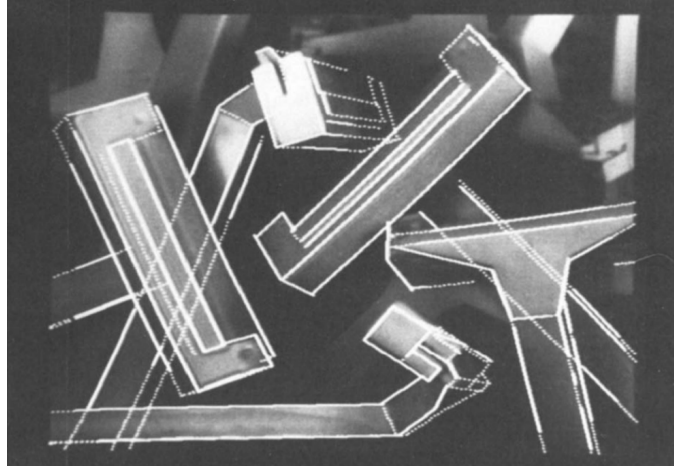


Figure 1.10: Razor detection done by David Lowe [23]

technique was used to detect humans. In this method, it endeavors to catch the shape of structures in the district by computing the gradient. [27] In the early 2000s, a good amount of visual data was available and enables scientists to measure the amount of progress that they make. One of the earliest benchmark datasets was PASCAL composed are 20 object classes and it has around several thousand images per category. ImageNet is another type of dataset for image recognition which was created in 2009. It consists of 14 million images and 22 thousand categories. By having this large dataset, overfitting was not a concern anymore for visual object recognition problem. Also, ImageNet team created a challenge called Large Scale Visual Recognition Challenge and asked the scientists from around the world to develop algorithms which are able to recognize images in their dataset. The results of ImageNet competition is shown in figure 1.12.

The most important part of this graph is the year 2012 in which a breakthrough happened and the error rate decreased significantly about 10 percent. The winner of that year used **Convolutional Neural Network** to beat other algorithms. From that year on, all other competitors used CNN as a basis for their network architecture and since then CNN became the state of art model for image recognition and image processing. Each year algorithms outperformed the previous networks and in 2015 RestNet was able to surpass the humans in image recognition.

1.6 Remote sensing and its importance

Remote sensing is the act gathering information about an object by using a device able to receive information without any physical contact with the desired object. Today the term is mostly used to describe the process of collecting and processing information based on satellite images. The first applications of remote sensing were wartime reconnaissance, disaster assessment and etc. Mostly during world war one. Actual evolution of remote sensing started around the second half of the twentieth century when artificial satellites



Figure 1.11: Object detection using SIFT

were developed and expanded the boundaries of remote sensing to global scale.

In Remote Sensing, two types of sensors are used more than others. Passive sensors and active sensors. Passive sensors receive information by gathering the radiation which is emitted by the object such as photography and infrared. On the other hand, active sensors send energy to scan an object and collect the received signal. RADAR and LIDAR are examples of active remote sensing.

Machine Learning and Deep Learning solutions are being progressively researched and implemented in various fields of science. They are used to analyse and process of satellite images to acquire a explicit representation. Machine Learning is considered as an priceless tool to analyse data gathered from remote sensing and telemetry. Recent advances in Deep Learning architectures have shown promising results over statistical counterparts in target object detection and identification. Although such learning architectures are

heavily dependent on computing resources, they are easy to use compared to sophisticated statistical models. Further, Deep Learning architectures are also able to render feature engineering as a part of their learning process which makes them extremely powerful in target object detection and identification process.

One of the most important applications of remote sensing is object detection which enables us to track and detect a specific object from the image taken from a satellite. The main challenges in the field of object detection for remote sensing is the large changes in the appearance of an object due to variations in the amount of light in the environment and also the low resolution of the images taken from the satellite.

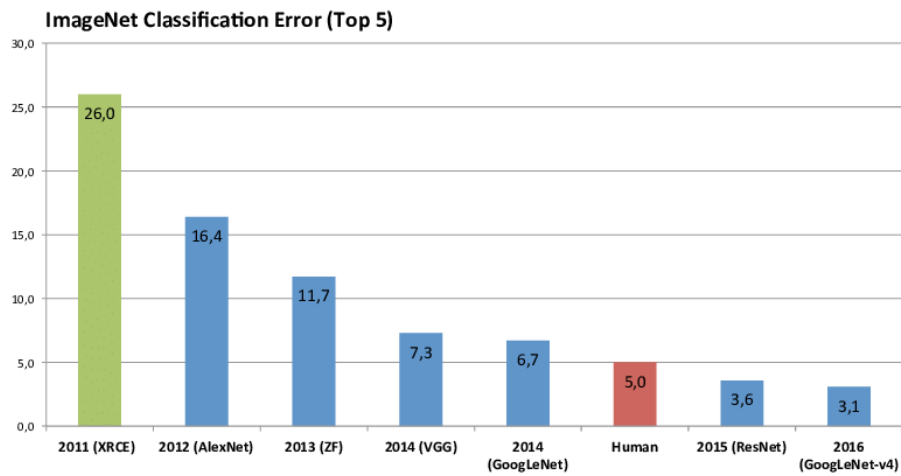


Figure 1.12: ImageNet results

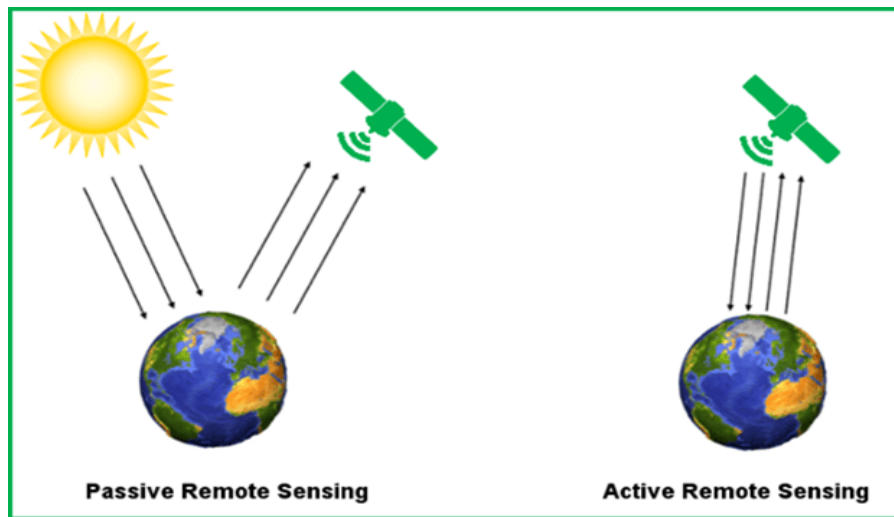


Figure 1.13: Active and passive remote sensing sensors

1.6.1 Related work

In the last years, a large number of methods and algorithms have been developed for object detection from satellite images. Because of the powerful capability of learning high-level and more complex and meaningful feature descriptions, deep CNNs are investigated in object-detection methods in opposition to the more traditional methods developed by a classifier based on handcrafted features. [28] [29] Here, some related work using CNN will be reviewed for both exclusive and non-exclusive object detection. A vector-guided vehicle detection method was introduced by Jin and Davis for IKONOS satellite. In this method, a shared-weight neural network is used to learn the absolute model of the vehicle. The shared-weight NN combines both spatial and spectral properties and classifies pixels into vehicles and non-vehicles. [30]

To solve the problem of the large-scale variance of objects, a hybrid deep CNN model was recommended to detect vehicles from satellite images. In the suggested model, feature maps of the last convolutional layer are divided into different blocks of variable-receptive field size to extract multi-scale features. this model divides all feature maps of the last convolutional and max-pooling layer of the CNN into multiple blocks of variable-receptive field size to extract multi-scale features. [31] Another solution to vehicle detection was using a graph-based superpixel segmentation to obtain image patches and then, by training a CNN model, it would be determined whether the extracted patch contains a vehicle or not. [32]

In other methods, instead of creating the network from scratch, a pre-trained network is used for object detection. A supervised learning framework is utilized to train an object detector. At the first phase, a pre-trained network is used as a transfer learning method to obtain high-level features. Afterward, the negative bootstrapping scheme is added to the training process to reach a faster convergence of the detector. [33] Another method which is used to detect oil tanks is to use the combinations of deep features extracted from the pre-trained convolutional network and local features. In the next step, an ellipse and line segment detector are used to detect candidate parts of the image. [34]

The sliding window is another method which was used to detect objects. It is a brute force approach in which a small portion of the image is selected and given to the network for classification. This part of the image is called the window and is sliding on the whole image with different ratios. Sliding window approach was used after training the pre-trained GoogLeNet network twice with two different fine-tuning options to create a framework for object detection. [35]

One of the challenges in satellite object detection is the orientation of the object. To solve this issue, a set of pre-trained features are extracted from combinations of network layers to create an orientation-robust framework for object detection. The selective search method is implemented to find the proposed regions and a linear SVM is used for final classification. [36] Another effective approach to solve the orientation problem is to enforce a new objective function in the problem and trying to optimize it. Additionally, a regularization constant is added to make sure that before and after the rotation, training samples have similar features. [37]

1.7 Thesis objectives and outline

The main goal of this thesis is to develop and implement an algorithm to detect ships based on satellite images by employing Deep Learning algorithms.

Object detection algorithms have the potential to be applied to satellite images to detect the various object. One important application of object detection on satellite images is ship detection due to increasing of shipping traffic in recent years. There are more ships used in today's world and this phenomenon leads to increase the chances of breaches at sea like ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. This has urged many organizations such as environmental protection agencies, insurance companies, and national government authorities, to have a closer look and pay more attention over the open seas.

For object detection, various methods are implemented and compared to achieve the best result. Nowadays object detection methods are maturing very rapidly and thanks to deep learning, new algorithms and models keep on outperforming the previous ones. For instance, methods such as SSD, YOLO, R-CNN, Fast R-CNN, Faster R-CNN are among the state of art models and they are able to deliver high accuracy and reasonable processing speed for a wide variety of applications. For the specific task of ship detection, Faster R-CNN and SSD method are implemented and compared due to their reasonable execution speed and high accuracy in comparison with other architectures.

The biggest challenge is the lack of appropriate dataset for ship detection and to overcome this issue, a synthetic dataset was created. The dataset was generated using Python programming language and OpenCV and PIL libraries were employed. Our synthetic dataset was divided into two parts for training and testing. 80 percent of data is used as training dataset and the remaining 20 percent is used as a test dataset.

After training both Faster R-CNN and SSD architectures on the synthetic dataset, we arrived to a conclusion that SSD method struggles in detecting small objects because of utilizing a limited set of default boxes in its architecture. On the other side, Region Proposal Network in Faster RCNN allows for better performance regarding small object detection. Based on the results mentioned, Faster RCNN is having a better accuracy and performance. Thus it is selected as a baseline for ship detection.

The developed algorithm was applied to real images taken from satellite images. Even though our object detector was trained on a synthetic dataset, it was able to detect ships with a good performance. Using the synthetic dataset approaches enables us to be able to perform deep learning solutions to the vast variety of problems in which a standard dataset is not available for training.

Chapter 2

State-of-the-art models for object detection

Object detection problem definition is to find the location of the objects in an image (object localization) and define the category that each image belongs to (object classification). In classical object detection models, the pipeline was divided into three stages:

- Region selection: Since object could appear in any position with different sizes in the image, the classical approach was to use a multi-scale sliding window which can cover all parts of the image. The drawback of this approach is that based on a large number of windows created, it is computationally expensive.
- Feature extraction: To be able to detect a various object, it was necessary to extract visual features from the image. These features allow for robust representation of the desired object.
- Classification: In the last step, a classifier will be used to distinguish between a goal object and other categories.

Those classical approaches based on the above architecture was not able to provide good results. The biggest challenges of those models were the computational cost of the sliding window approach and hand-engineering the low-level feature extractor.

When Deep Neural Networks (DNNs) started their dominance in the image processing field, they also have a massive effect on object detection. Since DNNs have a deeper model and architecture, they are able to extract more complex feature than traditional approaches. Additionally, Deep Learning training algorithms allow to learn the object features and representations automatically and without the need to design them manually. [38]

The state of the art object detection models are classified into two main categories:

- Region proposal based
- Regression/classification based

In region proposal based methods, a region of the image will be selected and it will be classified to the different categories. Most important region proposal based methods are:

- R-CNN
- Faster R-CNN
- Fastest R-CNN

On the other hand, in regression/classification based framework, object detection problem is seen as a regression or classification problem and the process is done in only one step which leads to having a faster response. The state of the art regression/classification based methods are:

- You Only Look Once (YOLO)
- Single Shot MultiBox Detector (SSD)

In this chapter, above mentioned object detection state of the art models will be discussed in more details.

2.1 Region proposal based methods

2.1.1 R-CNN

R-CNN method was used to improve the feature extraction and accuracy of finding regions of interest. When R-CNN was proposed and published in 2014 it was able to reach the mean average precision of 53.3% which was about 30% improvement with respect to the previous best result. [39]

The process done by R-CNN can be divided into three steps:

- Region proposal generation: The R-CNN uses selective search method to create two thousand region proposals for each image. In selective search, initial sub-segmentations are generated and similar parts are combined together to create a larger region.
- CNN based deep feature extraction: In this step stage, each proposed region is resized to have a fixed resolution. The Alexnet [40] convolutional neural network is used to create a 4096-dimensional feature as the final representation. Because of the structure of Deep Learning networks, this type of rich and robust representation is obtainable.
- Classification and localization: In the last step, region proposals are scored with a pre-trained linear SVM. Those scores are then adjusted by a bounding box regression. Afterward, a non-maximum suppression is utilized to generate the final bounding boxes for the specific object.

One of the most important drawbacks of this method is the high cost of computation. For instance, generating the two thousand regions could take around two seconds. Another problem of this network was that due to the existence of fully connected layers in the architecture, the input image should have a fixed size (in this case 227 by 227).

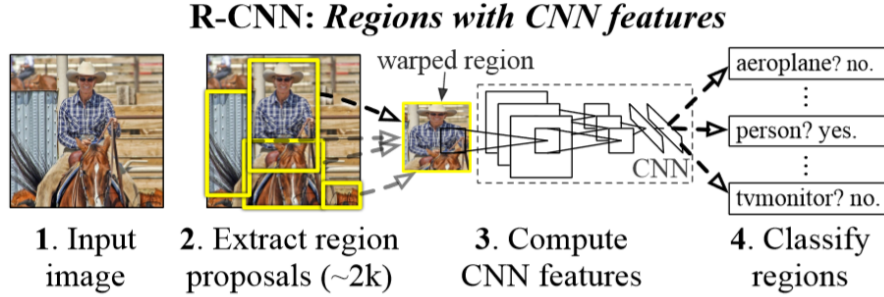


Figure 2.1: R-CNN flowchart [39]

2.1.2 Fast R-CNN

Basic R-CNN model has been rapidly improved. In this section, changes that have been introduced to R-CNN model to reach Fast R-CNN will be introduced. Because of the fully convolutional layers in the base CNN classifier of R-CNN, input image should have a fixed resolution of 224 by 224. Thus, after extraction the object proposals, they should be rescaled and this change of resolution can change the visual aspect of the object. To overcome this issue, the last pooling layer of any CNN network could be substituted by Spatial Pyramid Pooling (SPP).

Spatial Pyramid is built on top of the region of interest layer. The first level of the pyramid architecture is a region of interest itself. On the next level, the region is divided into four equal cells. On the last level, the region will be split into 16 parts on four by four grid. After these divisions, average pooling will be applied to each part. For instance, if the feature map has depth of 256, then by applying the pooling layer to each cell, we produce a vector which has the length of 256. After applying the pooling layer, all outputs are concatenated together and they are passed as the input to the fully convolutional layer. By utilizing this method, we obtained a feature representation which has a fixed length regardless of input image resolution. Using this method dramatically increases the processing speed.

In the Fast R-CNN paper, it was proposed to use the Region of Interest Pooling layer (ROI) layer instead of SPP layer. ROI layer is a simplified version of SPP layer, only the last layer of SPP is used and others are neglected. In addition to ROI layers, two more changes have been made in Fast-RCNN with respect to original R-CNN. First, instead of using SVM as classifier, softmax is used. Second, multi-task training is used to train classifier and bounding box regressor simultaneously. The input image and set of object proposals are supplied to the neural network. The neural network produces a convolutional feature map. From convolutional feature map, feature vectors are extracted

using Region of Interest Pooling layer. Then, the feature vectors affect into a sequence of fully convolutional layers. The output of fully convolutional layers are branched into softmax for classification, and a regressor to generate four real-valued bounding box coordinates output. Fast R-CNN can be trained to this multi-task loss. This multi-task loss is sum of classification loss, and bounding box regression loss. For classification, log loss is used. The smooth L1 loss is used for the bounding box regression. Because in this case we do not have a separate SVM training, Fast R-CNN can be trained end-to-end, which is faster than previous method. weakness of this method is that this method still rely on proposal generation methods such as Selective Search to find best regions and this process is time consuming and takes most of the test time. [41]

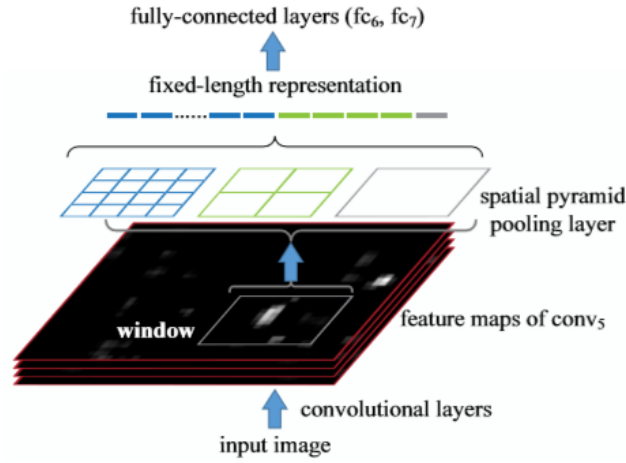


Figure 2.2: SPP architecture [42]

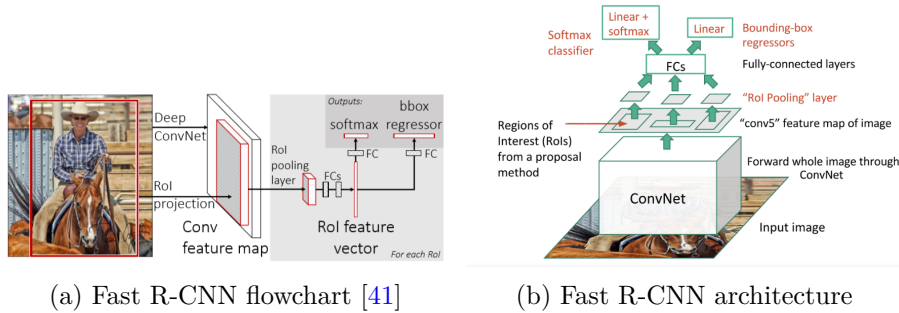


Figure 2.3: Fast R-CNN Network

2.1.3 Faster R-CNN

Next step towards the evolution of R-CNN model is Faster R-CNN. This method can be considered as the combination of Fast R-CNN model and Region Proposal Network (RPN). In Faster R-CNN, we are able to detect objects in a single pass with a single neural network which significantly improves the speed of object detection procedure. RPN is only a convolutional neural network and it is utilized to generate a proposal regions and replace classical region proposal methods such as Selective Search. The Region Proposal Network slides over convolutional feature map. RPN simultaneously classify the corresponding region as an object or non-object in the location of the bounding box. The position of the sliding window position gives us information about the location of the object with reference to the image. Box regression is used to fine-tune localization by utilizing the sliding window as a reference. At each sliding window position, a set of object proposals is generated. Each of the created proposals could be in different sizes and aspect ratios. Those proposals are called "anchors". Anchors help improving handling the objects with different sizes and aspect ratios and it could be seen as a sliding window that could have various sizes and aspect ratios.

During the training of RPN, the anchor is treated as a positive value if intersection over union (IoU) is greater than 0.7, or it reaches maximum possible value for all anchors for this specific ground truth example. It is negative if IoU is less than 0.3. Due to the new architecture of Faster R-CNN, it is trainable end-to-end as a unique network which has four separate losses:

- RPN classification loss: Defines the extent of bounding boxes created by RPN network which are accurately classified as foreground or background. The cross entropy loss function is implemented to calculate classification loss.
- RPN regression loss: The distant measure between the predicted bounding boxes and ground truth bounding boxes. The summation of all the regression losses are calculated by utilizing smooth L1 function.
- Fast R-CNN classification loss: Defines the extent of bounding boxes created by RPN network which are accurately classified as correct object class and not as foreground or background. Same as RPN classification loss, cross entropy is used but in this case it is calculated based on all available classes and not only foreground and background.
- Fast R-CNN regression loss: It is similar to RPN regression loss but in this case the loss is class specific.

Since in new architecture there is no need to have an external region proposal method, the execution speed is improved significantly.

Despite having the anchor boxes, it is difficult for RPN to handle objects with various scales and sized since RPN has a fixed receptive field. To solve this problem, a set of RPN could be trained for different scales so each RPN has dissimilar convolutional layers as input and receptive field has various sizes now. This method substantially improves the ability to detect a very small and very large object in Faster R-CNN.

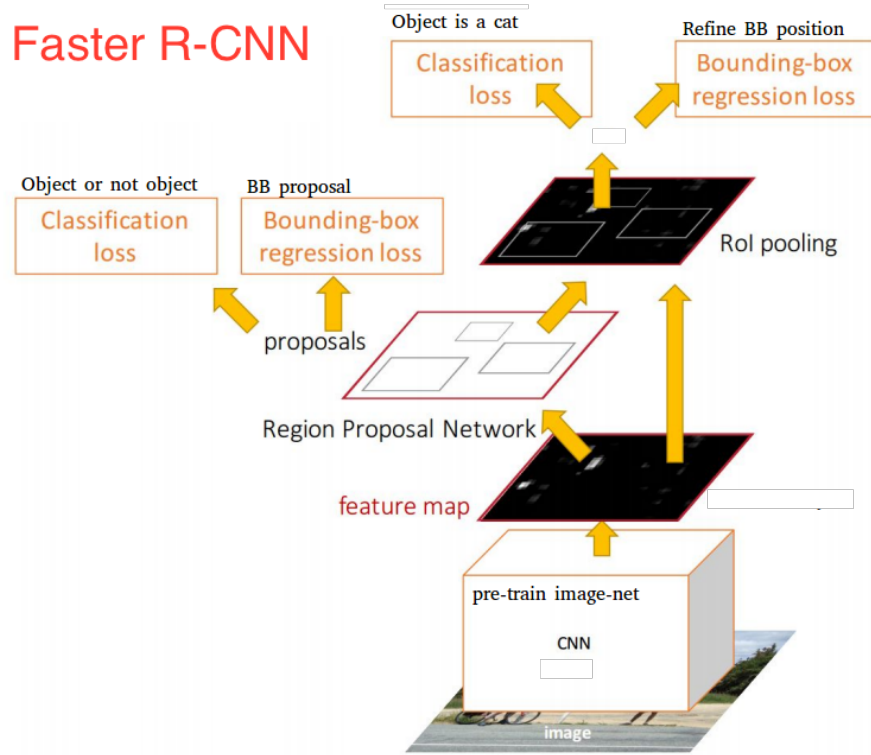


Figure 2.4: Faster R-CNN architecture

2.2 Regression/Classification based methods

Region proposal based frameworks are created by combining several stages together, region proposal generation, feature extraction with CNN, classification and bounding box regression. Despite recent end-to-end architecture of the Faster R-CNN, an alternative training is needed to fine-tune the shared convolution parameters between RPN and detection network. Respectively, the time taken to take care of diverse components becomes the challenge in real-time applications where speed is crucial. One-step frameworks based on regression/classification methods could be used to decrease time expense.

2.2.1 YOLO

The intention of the creators of YOLO was that they wanted to have one neural network to do object detection completely.

At the first step in SSD architecture, a grid is put on the top of the original image to divide it into equal parts. If the middle of an object (not all of it) falls into a specific grid cell, that grid cell is responsible to detect that object. each cell in the grid is responsible to predict a few different things. The first thing to predict is a bounding box and confidences related to it which indicates the probability of existing an object in the

cell. In the next step, all of those bounding boxes and their corresponding confidence are applied to the image and create a map of all the possible objects in the image. Then each cell is responsible to also predict a class probability.

One thing to notice is, the prediction done by the cell is "conditional" in a sense that if there exists an object in that cell of the image, what it would be. Now if we take these conditional probabilities and multiply them by the confidence values from bounding boxes, we have all of the bounding boxes weighted by their actual probability for containing the actual object. Lastly, by adding a threshold to these predictions we reach the final result. Also to avoid having multiple bounding boxes for the same object, non-maximum suppression is used. [43]

YOLO network architecture is inspired by the GoogLeNet model which was used for the task of image classification. the network has 24 convolutional layers followed by two fully connected layers. The network architecture is shown in figure 2.5.

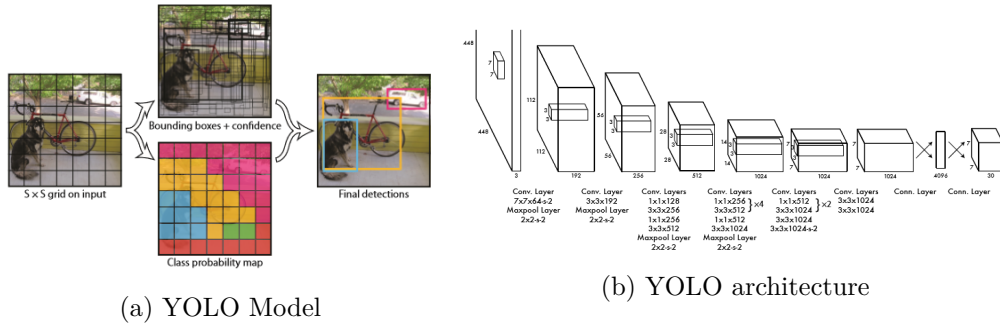


Figure 2.5: YOLO network [43]

2.2.2 Single Shot MultiBox Detector (SSD)

One of the biggest problems of YOLO is that this network struggles to detect small objects due to the spatial constraints forced to bounding box predictions. [43] Additionally, due to having several downsampling layers in YOLO architecture, it is having difficulty when it comes to detecting an object in new and unusual situations. Because of above-mentioned problems of YOLO, scientists suggested a new model called the Single Shot MultiBox Detector (SSD). In this model, instead of using the fixed grids suggested by YOLO, a set of anchor boxes with different scales and aspect ratios are utilized.

it is possible to reuse the computation already done for the problem of image classification to solve object detection problem by using activations from the last convolutional layer. In this point, we have the spatial information but represented on a smaller scale. at this point it is possible to use a convolutional layer to classify each cell and also from the same layer, we could attach another layer to find the four numbers corresponding bounding boxes. using this method it is possible to do the classification and localization utilizing one network architecture.

SSD matches the objects by using default boxes which have various aspect ratios. each element in the feature map has a certain number of bounding boxes associated with

it and any of those default boxes which has an Intersection over Unit(IoU) of 50 percent or more is considered as a match. In the SSD architecture, six feature maps are presented and each is responsible for a different object scale. This allows the network to detect the objects across a large range of scales. As mentioned for each default box network outputs a vector of length C which represents the probabilities of each box containing an object belongs to each class C . Also an offset vector is generated which has four numbers representing the predicted offset from the default box to the actual position of the object.

In the SSD architecture, VGG16 is used as the backbone network and several other feature layers are added to the end of the networks. Those layers are responsible for predicting the offset of each default box having various scales and aspect ratios. Also, their corresponding confidence should be calculated. Finally, the network is trained with the weighted sum of localization loss and confidence loss. [44]

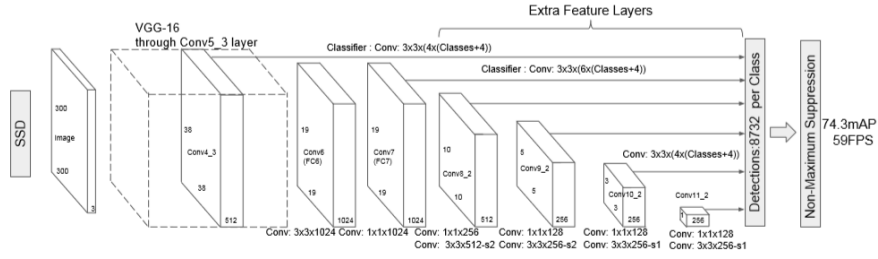


Figure 2.6: SSD architecture

2.3 Some precision metrics definitions

It is necessary to comprehend some basic criteria related to statistics, classification and object detection to fully understand the accuracy metrics utilized in object detection.

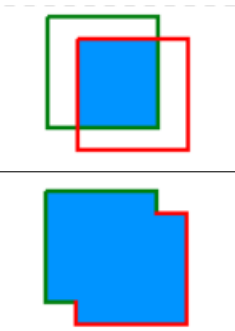
2.3.1 True Positive, False Positive, False Negative and True Negative

To evaluate the performance and accuracy of a object detection architecture, some primary metrics are defined as follows:

- True Positive (TP): A correct detection. Detection with $IOU \geq \text{threshold}$.
- False Positive (FP): A wrong detection. Detection with $IOU < \text{threshold}$.
- False Negative (FN): A ground truth not detected.
- True Negative (TN): Does not apply in object detection. It indicates the bounding boxes which should not be detected and since there is so many of them in each image, it is ignored.

2.3.2 Intersection Over Unit (IoU)

Intersection Over Union (IoU) is calculated based on Jaccard Index evaluating the overlap between two bounding boxes. It requires a ground truth bounding box and a predicted bounding box are needed to find IoU. Using this measure, I could be stated if detection is valid (True Positive) or not (False Positive). IoU is defined by the overlapping the area between the predicted bounding box and the ground truth bounding box divided by the area of union between them. The definition is shown in the equation below:

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}} \quad (2.1)$$


2.3.3 Precision and recall

Precision measures how accurate is your predictions. i.e. How much the predictions made by the predictor is correct. How many of the detected boats are actually boats and not ships?

Recall indicates how good you find all the positives. For example, how many of the actual desired objects presented in the picture are detected by the system.

The mathematical definition for recall and precision is shown in equation 2.2:

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \\ Recall &= \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truth}} \end{aligned} \quad (2.2)$$

2.3.4 Mean Average Precision (mAP)

The mAP is the metric utilized to measure the accuracy of object detectors. It is the average of the maximum precisions at several values of recall. A way to perform a comparison between different object detectors is to find the area under the curve (AUC) of the Precision and Recall curve. Since the curves are mostly having zigzag behavior, comparing different curves is a difficult task because the curves cross each other frequently. The idea of Average Precision (AP) is defined as approximating the area under the Precision and Recall curve and then finding the area under it.

The interpolation of the curve is calculated based on the equation below:

$$\sum_{r=0}^1 (r_{n+1} - r_n) P_{interp}(r_{n+1})$$

$$P_{interp}(r_{n+1}) = \max_{\tilde{r}: \tilde{r} \geq r_{n+1}} P(\tilde{r}) \quad (2.3)$$

$P(\tilde{r})$ is measured Precision at Recall value of \tilde{r} . AP is now calculated by interpolating the precision at each level of r , taking the maximum precision value whose its corresponding recall value is greater or equal than $r + 1$. Hence, the estimated area under the curve is determined. The mean Average Precision (mAP) score is computed by taking the mean of AP over all classes for PASCAL VOC2007 dataset or over all IoU thresholds for Microsoft COCO dataset.

Chapter 3

Problem definition

3.1 Modern maritime piracy

It might seem strange and unbelievable but seaborne piracy is still a significant international issue in the 21st century. It is estimated that billions of dollars are lost per year due to piracy worldwide. [45] Piracy of maritime reached its maximum in 2010 in which around 445 incidents were reported. The most dangerous regions with a great chance of pirate attacks include Indonesia, the Philippines, Nigeria, and Somalia. Strategic passages for oil transport such as Bab-el-Mandeb, near Somalia, or the Strait of Malacca off the Indonesian coast have become main targets for maritime crime in recent years. In 2009, around 13.6 million barrels of oil per day was passed through the Strait of Malacca. This amount of oil is more than the amount of oil imported to the European Union each day. Also, the number of incidents occurred in Vietnam was quadrupled between 2014 and 2015. [46] Pirates also use larger ships, often called "mother ships" which they are used to supply the smaller motorboats.

Instead of targetting the ship cargo, modern pirates are willing to target the personal belongings of the crew and the contents of the ship's safe, which could contain large amounts of cash. On the other hand, the pirates might force the crew to go out off the ship and after sailing it to a port and repainting the ship, they create a new identity for the ship using false papers and resell the ship. [47]

Also, in some other cases, pirates attacked some luxury resorts and kidnapped the people and bring them to their own hideouts. After capturing them for some time, pirates ask the families to pay a ransom to release them. [48] Since 2008, Somali pirates located in the Gulf of Aden earned about 120 million US dollar per year. This act of piracy reportedly costs the shipping industry between 900 million US dollar to 3.3 billion US dollar annually. [49]

3.2 Anti-piracy measures

In the 20th century, it was traditional for merchant vessels not to be armed and carry weapons, the U.S. Government has recently changed the rules in this matter and now it is considered as "best practice" for vessels to have a team of armed security guards. The

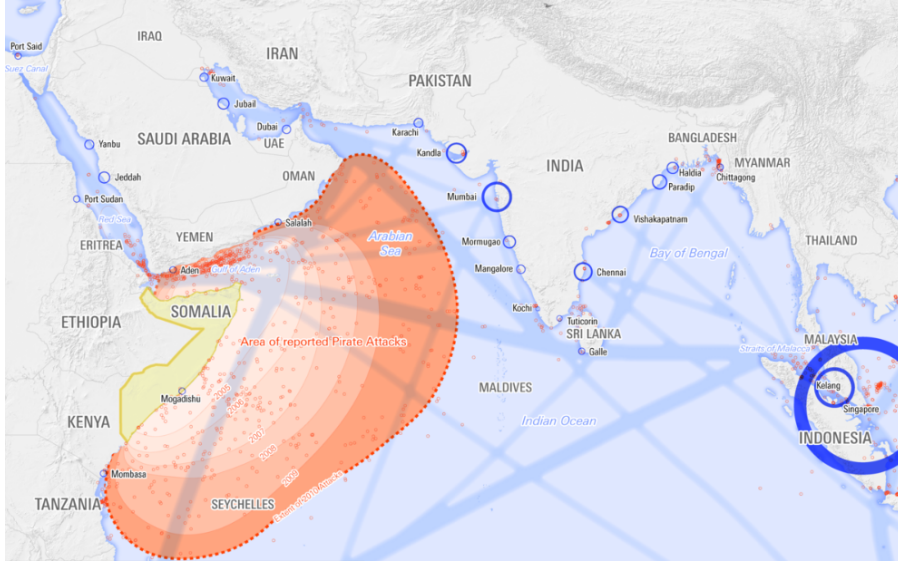


Figure 3.1: Piracy threat map between 2005 and 2010

crew can be given weapons training, and it is legal to fire warning shots in international waters to keep the pirates away. [50]

Also, the Automatic Identification System (AIS) is developed by the European Space Agency (ESA) to help the ship protect themselves. AIS is being used by the ships which weight over 300 tons. AIS transmits information about the ship position and its movements. If any unexpected change happens in this information, it could indicate a potential thread for the ship. Initially, this data was able to be received by if there was a ship nearby. But recently specific satellites have been launched which are able to receive and transmit the data sent from ships. [51]

The above-mentioned solutions are useful when the pirate attack already happened and does not help to prevent the risk of attack. We proposed a ship detection system which is able to detect the ships by processing the images received from the satellite images. In this proposed method, the boat and cargo ship is detected and an alarm could be raised to warn the cargo ship and also the authorities. By utilizing this method the time of the response to the attack is reduced. In these types of attacks, every second count and could lead to saving the lives of crew members by performing the right countermeasures.

3.3 Sentinel-2 mission

Due to cost-free availability and ease of use and access to satellite images, Sentinel-2 satellites were chosen as the main source of aerial images for ship detection. Sentinel-2 is a mission from the EU Copernicus Programme for Earth observation. Optical images are acquired at high spatial resolution (10m to 60m). A broad range of services and applications are supported by this mission including, agricultural monitoring, emergencies management, land cover classification or water quality. Sentinel-2 has been developed by

European Space Agency (ESA), and the satellites were manufactured by a consortium led by Airbus DS.

Sentinel-2 satellites support multi-spectral data which has 13 bands: four of them are bands at 10 m, six bands at 20 m and three bands at 60 m spatial resolution. It covers land surfaces from 56°S to 84°N, coastal waters and the Mediterranean Sea. Revisiting happens every 5 days with the same viewing angles. Field of view is 290 Kilometers. To have high mission availability two identical satellites called Sentinel-2A and Sentinel-2B are operating together. The planned orbit for this mission is a Sun-synchronous orbit at 786 km altitude and 14.3 revolutions per day, with a descending node at 10:30 a.m. This orbit is chosen to have the best trade-off between minimizing the cloud cover and having good sun illumination. [52]

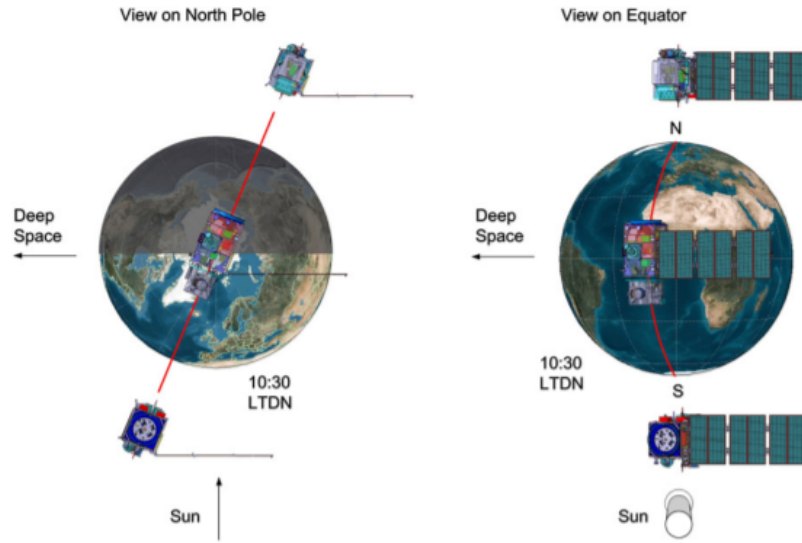


Figure 3.2: Sentinel-2 satellite orbital configuration [52]

3.4 Synthetic dataset

As mentioned in chapter 1, one of the prerequisites of solving any Machine Learning problem is having a suitable dataset. For the case of ship detection, this type of dataset was not publicly available by the time of exploiting this project. Thereby, a synthetic and artificial dataset was created to solve the issue.

The dataset was created by employing OpenCV and PIL open-source libraries. Around five hundred images were generated. In each image, the random number of boats and cargo ships are placed on different sea backgrounds. The location and orientation of boats and cargo ships are chosen randomly. Furthermore, the angle between the wakes and the length of the wake for each boat are chosen arbitrarily. Some samples of the generated images are shown in 3.5.

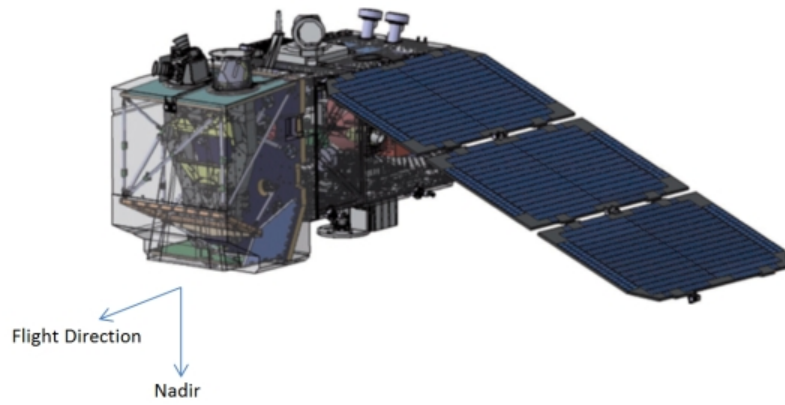


Figure 3.3: Schematic View of the Deployed SENTINEL-2 Spacecraft (image credit: EADS Astrium)

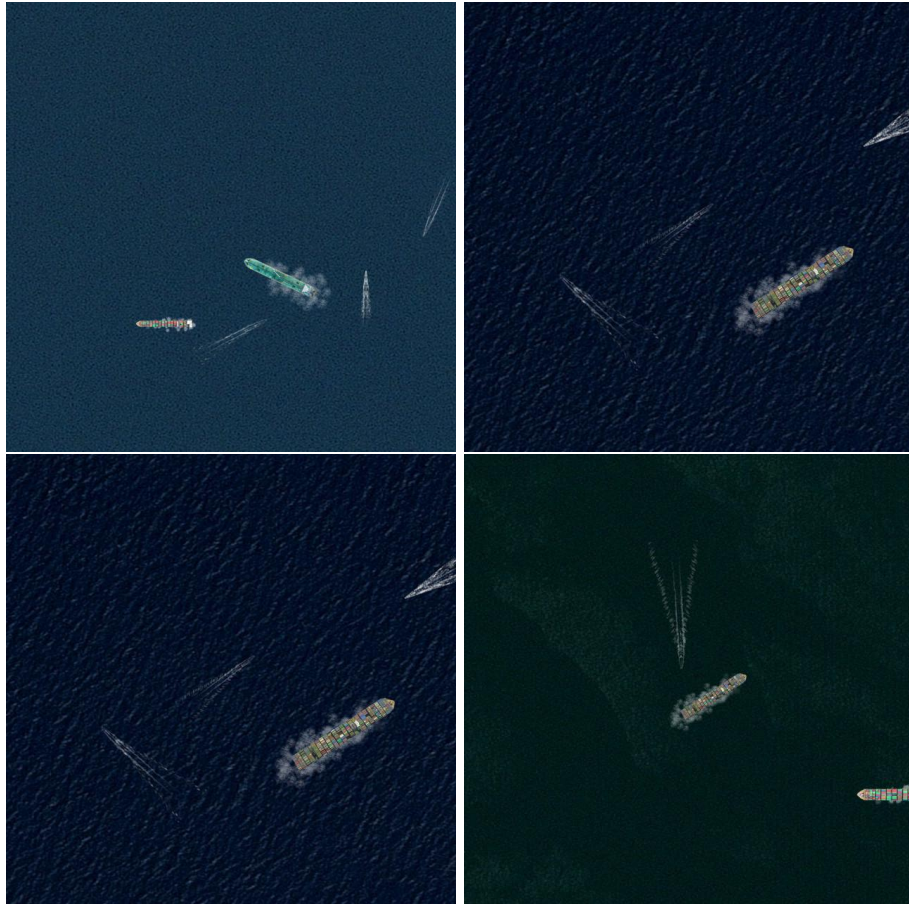


Figure 3.4: Generated images from synthetic dataset



Figure 3.5: Generated images from synthetic dataset

The output of the developed code is the generated images of boats and cargoes alongside with their ground truth data indicating the name of each image, class of the detected object (boat or cargo in this case) and the corresponding coordinates of each bounding box. The synthetic dataset was divided into two parts for training and testing. 80 percent of data is used as training dataset and the remaining 20 percent is used as a test dataset.

1	filename	class	xmin	ymin	xmax	ymax
16	102.jpg	cargo	188	84	211	96
17	103.jpg	boat	450	234	474	257
18	103.jpg	boat	152	168	170	185
19	103.jpg	cargo	500	482	519	492
20	103.jpg	cargo	548	218	580	235
21	104.jpg	boat	457	530	487	558
22	104.jpg	boat	491	312	523	342

Figure 3.6: Some samples of the ground truth data generated in CSV format

Chapter 4

Use Case Results

Up to this point, we gathered all of the required prerequisites including the dataset to tackle our object detection problem. There are many Deep Learning frameworks already available which are designed to help the user easily develop a Deep Learning algorithm without any need to hard-code all the mathematical concepts related to Deep Learning such as backpropagation, gradient descent and etc. Mentioned frameworks facilitate the process of gathering data, training, making predictions, and extracting future results. One of the most popular Deep Learning frameworks is Tensorflow which is used in this project for object detection.

4.1 Google Tensorflow

Tensorflow was originally developed by Google Brain Team as an open-source library initially made to handle complex numerical computations in a large scale. Thereafter, the framework is implemented to Machine Learning and Deep Learning concepts. Tensorflow employs Python programming language to produce the front-end API for application development while the developed applications are executed in C++ which is significantly faster than Python. It can be used in a wide variety of Machine Learning and Deep Learning applications including image recognition, natural language processing, regression problems and many more. One of the advantages of using Tensorflow is the support for production prediction at large scale applying the same model used for training.

Due to massive investment and support by Google, an application developed by TensorFlow is executable on a wide variety of targets including a local machine, a cluster in the cloud, iOS and Android devices and modern browsers. Likewise, training on the GPU is supported by Tensorflow which dramatically increases the computation speed and decreases the amount of time required for training the model. An additional benefit of using Tensorflow for this project is its native support for state of the art models used for object detection. Some of the supported architectures are:

- SSD
- Faster RCNN

- Mask RCNN

The models mentioned above are available in different versions and they vary in the base network architecture. As mentioned before, all of the object detection models are using an image classification network as their backbone. For instance, Faster RCNN models are available with ResNet50, ResNet101, InceptionV2, and NASnet.

The backbone of Tensorflow is built on Tensors and Computational Graphs. In mathematics, a Tensor is an N-dimensional vector used to describe an N-dimensional space. figure 4.1 illustrates the concept of Tensors having minimal dimensions.

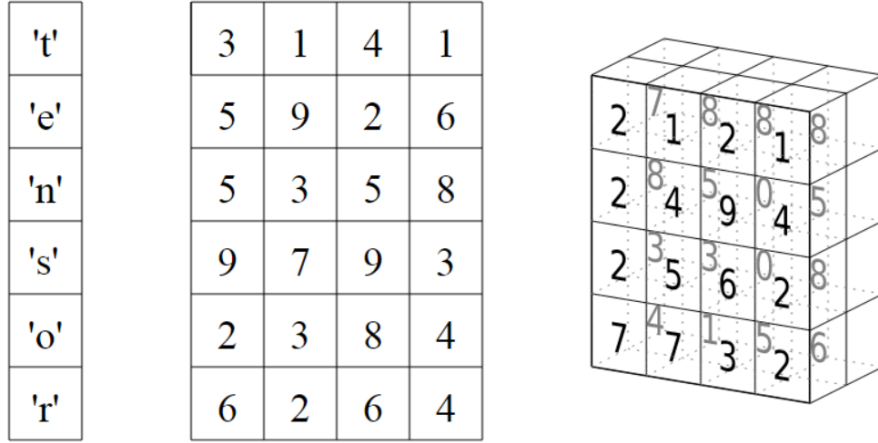


Figure 4.1: Some examples of Tensors

A Computational graph is defined as a set of interconnected entities, called nodes. The connection between those nodes is established via edges. In a dataflow graph, the edges allow data represented in Tensors to flow between the nodes in a specific manner. To give an example, consider the equation of $a = (b + c) * (c + 2)$. The corresponding computational graph is shown in figure 4.2.

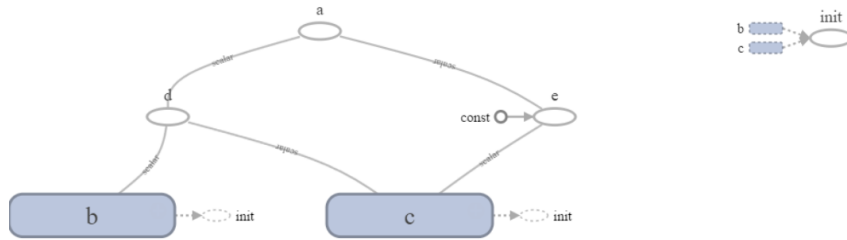


Figure 4.2: Computational graph example

In the Tensorflow environment, the written code is able to only generate the graph

and initialize the sizes of Tensors to be used and operations to be executed on them. It is not responsible to assign any numerical value to any of the Tensors. The graph is not executed unless the session is run. Hence, to assign the values and force them to flow through the graph, a session should be created. Although in the second version of Tensorflow API, eager execution is enabled which allows the user to execute Tensorflow kernels immediately. This feature makes debugging easier and more straightforward. Additionally, it is now possible to make use of Python built-in control flow structures such as loops and conditional function within Tensorflow API. The architecture of Tensorflow API is shown in figure 4.3.

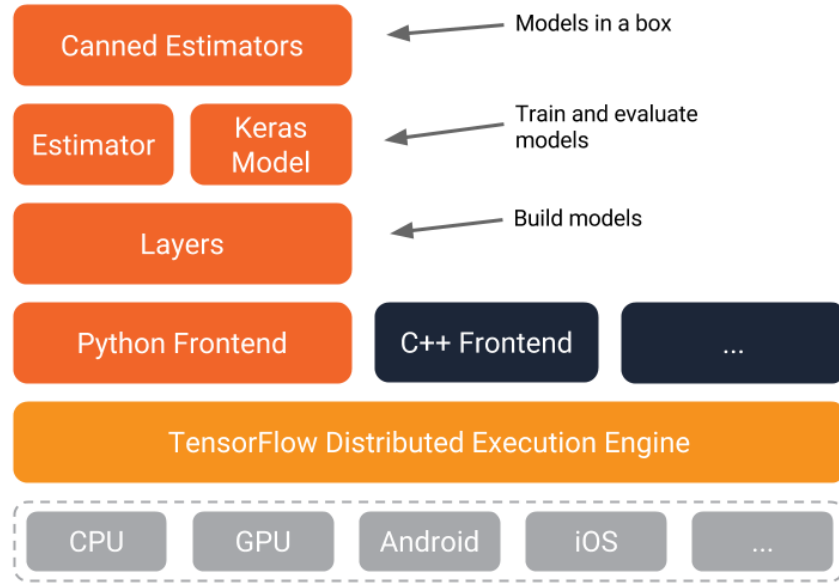


Figure 4.3: Tensorflow API Architecture

4.2 Object detection by Tensorflow

After having the dataset ready and choosing the suitable Deep Learning framework to use, the input data should be converted in a format which is understandable by our framework. The format accepted by Tensorflow frame-work for object detection is called TFRecord file format. Before converting our data to TFRecord format, they should be divided into two parts for training and testing. 80 percent of data is used as training dataset and the remaining 20 percent is used as a test dataset. According to Tensorflow documentation, the input images should be converted to numerical values before creating the TFRecord file. Moreover, all of the input data including encoded images, file names, bounding box coordinates, image resolution and corresponding classes have to be employed to generate a TF.Example proto. After TF.Example proto is generated, we are able to create TFRecord files needed for training. TFRecord files are considered the binary storage format to be

used by Tensorflow. using a binary file format leads to a significant improvement in the performance of the import pipeline and consequently reduce the training time of the model. Binary formatted data takes less space on disk, accordingly takes less time to copy and could be read much more efficiently from disk. Another advantage of using TFRecord format is that instead of loading all the data on the memory, only the data which is needed at the time (e.g. a batch) is loaded from disk and then processed. This allows for more efficient use of available hardware, especially when working with a large amount of data which is usually the case when dealing with object detection problems.

As mentioned before, Tensorflow library supports built-in state-of-the-art object detection models. Most of the object detection models in Tensorflow are pretrained based on Microsoft COCO dataset. Microsoft COCO dataset is designed specifically to tackle object detection, image segmentation, and image captioning tasks. In total COCO dataset has 328,000 images including 2,500,000 labeled instances and 80 objects. categories. [53]

The Task of reusing a pre-trained network architecture as a starting point for a second assignment is called transfer learning. It is the improvement of learning in a new task by means of the transfer of knowledge from a related task which has already been learned. In transfer learning, an initial network is trained on a base dataset which is COCO in our case. Afterward, we transfer the learned features to a second network to be trained on a final dataset. By employing transfer learning we would save the time needed to train the network from scratch. Moreover, it enables us to use a comparably smaller dataset to solve our problem.

In Tensorflow object detection environment, various options and configurations for the training and evaluation process are stored in a file so-called config file. The *configfile* make use of protobuf files to configure the network training and evaluation parameters. At the highest level, the config file could be split into 5 parts:

- The model configuration: The type of model to be trained is determined in this part.
- The train_config: Contains the parameters used for model training such as SGD parameters, feature extractor initialization values of feature extractor and input preprocessing.
- eval_config Defines the sets of the metrics to be used and reported during evaluation process
- train_input_config: It is used to define the training dataset.
- eval_input_config: It is used to define the dataset in which the model will be evaluated on.

The main structure of configuration file is shown in listing 4.1.

Listing 4.1: General structure of config file

```
model {  
  (... Add model config here...)  
}  
  
train_config : {  
  (... Add train_config here...)  
}  
  
train_input_reader: {  
  (... Add train_input configuration here...)  
}  
  
eval_config: {  
}  
  
eval_input_reader: {  
  (... Add eval_input configuration here...)  
}
```

There are various model parameters to be configured. The optimal settings are highly dependent on the desired application. As mentioned in chapter 2, Faster R-CNN models perform better where high accuracy is desired. On the other hand, if processing time is considered as the most important factor, SSD models are recommended. In ship detection case, both network architectures are tested and compared to find the best choice for our specific problem. The parameters defined in this part depends on the back-bone architecture of the network. A noteworthy parameter to be mentioned in this section is `num_classes` which states the number of classes to be detected by the architecture. Although, one of the most important fields in this section is `num_classes` which indicates how many classes we have in our training and evaluation data.

Inputs accepted by The Tensorflow Object Detection API should be in the TFRecord file format. Users must specify the locations of both the training and evaluation TFRecord files in the configuration file. Furthermore, users should also create a label map file, which corresponds to the mapping between a unique class id and class name. Mentioned label map file should be saved in .pbtxt format. It should be noted that the defined label map file should be identical for both training and evaluation. Label map file used for ship detection is defined in listing 4.2. Since we have two object classes, boat, and cargo, two entities are defined alongside with their corresponding ids.

Listing 4.2: Defined label map file

```
item {  
  id: 1  
  name: 'boat'  
}  
item {  
  id: 2  
  name: 'cargo'
```



```
}
```

The structure of input configuration is shown below:

Listing 4.3: Input configuration structure

```
tf_record_input_reader {  
input_path: "/usr/home/username/data/train.record"  
}  
label_map_path: "/usr/home/username/data/label_map.pbtxt"
```

The `train_config` section is responsible to determine various parts of the training process:

- Initialization of model parameters.
- Preprocessing. the inputs.
- SGD parameters.

In model parameters initialization, a pre-trained object detection network should be defined. the field called `fine_tune_checkpoint` is used to provide the path to the pre-defined checkpoint. Mentioned checkpoints are downloadable from Tensorflow object detection GitHub repository.

Input preprocessing part is utilized to augment the provided dataset by applying function including but not limited to:

- Image normalizing.
- Randomly flip the images horizontally.
- Randomly flip the images vertically.
- Randomly scale the value of each pixel in the images.
- Randomly scale the images.
- Randomly crop the images.
- Rotate random images by 90 degree.
- Randomly adjust the brightness, contrast and hue of images.

SGD parameters are hyper-parameters for gradient descent including the type of the optimizer, learning rate value and optimizer learning rate configurations.

Optimizers supported by Tensorflow are:

- RMSProp optimizer.
- Momentum optimizer.
- ADAM optimizer.

Configurations available for optimizer learning rate include:

- Constant learning rate.
- Manual step learning rate.
- Exponential decay learning rate.
- Cosine decay learning rate.

An example of `train_config` section is shown in listing 4.4.

Listing 4.4: `train_config` sample

```
batch_size: 1
optimizer {
  momentum_optimizer: {
    learning_rate: {
      manual_step_learning_rate {
        initial_learning_rate: 0.0002
        schedule {
          step: 0
          learning_rate: .0002
        }
        schedule {
          step: 900000
          learning_rate: .00002
        }
        schedule {
          step: 1200000
          learning_rate: .000002
        }
      }
    }
    momentum_optimizer_value: 0.9
  }
  use_moving_average: false
}
fine_tune_checkpoint: "/usr/home/username/tmp/model.ckpt-#####"
from_detection_checkpoint: true
load_all_detection_checkpoint_vars: true
gradient_clipping_by_norm: 10.0
data_augmentation_options {
  random_horizontal_flip {
  }
}
```

One last part of the config file corresponds to `eval_config`. The crucial components to set in this section are `num_examples` and `metrics_set`. The `num_examples` parameter is the number of image instances included in evaluation dataset. The parameter `metrics_set` specifies the metrics to be run during the evaluation process and it is often set to `coco_detection_metrics` to have the same metrics as Microsoft COCO dataset.

For our case of ship detection, two config files are defined for Faster RCNN and SSD architectures.

For Faster RCNN config file, the model section is divided into two parts. The first part represents the configurations related to Region Proposal Network (RPN) and the second part is used to configure the stage box classifier. The prefixes "first_stage_" and "second_stage_" are used to differentiate those parts.

After defining the network architecture as Faster RCNN with inception version 2 backbone, the number of classes is declared. Images are resized to have a minimum of 600 pixels in their smaller dimension and maximum of 1024 pixel in their larger dimension. The aspect ratio is kept constant during the conversion. The resizing process is done by configuring "image_resizer" parameter. The feature extractor is chosen as "faster_rcnn_inception_v2" which acts as the backbone for the architecture.

"first_stage_anchor_generator" parameter defines grid anchors to be used in Faster RCNN. The scale, aspect ratio, and stride are defined for the anchors. Type of the layers used in the first stage is defined to be convolutional layers. Furthermore, an L2 regularization layer is used to decrease the chance of over-fitting. weights of the first stage layer are initialized in the form of a truncated normal distribution having 0.01 standard deviation. Truncated normal distribution behavior is similar to random normal initializer except that in this case values which are more than two standard derivations from the mean is discarded and regenerated. To avoid having overlapping bounding boxes, a Non-Maximum Suppression (NMS) method is used. NMS initially discards the bounding boxes having possibility less than 70 percent, takes the bounding box with maximum probability as the reference and finally discards any remaining cell which has more than 50 percent intersection with the reference cell. A max pooling layer is then added to the first stage with a stride of 2 and kernel size equals to 2. The maximum number of proposals in each image is set as 300.

For the second stage, a set of fully connected layers are used alongside with an L2 regularization layer. For initializing the weight of the second stage, `variance_scaling_initializer` is employed which keeps the input variance scale constant. Using this initializer leads to preventing the network to explode or diminish. The parameters set in this section correspond to usage if Xavier initializer [54]. Another Non Maximum Suppression (NMS) is implemented in this section with intersection threshold of 60 percent. A softmax layer is utilized to normalize the probability distribution of the output. Localization loss weight and classification loss weight are set to 2 and 1 respectively.

In the training configuration section, an SGD optimizer with momentum is used. Normal SGD algorithm has some troubles when the surface curve is much more steeply in one dimension with respect to the other dimension. this phenomenon is common, especially around local optima points. In SGD with momentum algorithm, this problem is solved by adding a fraction of the update vector of past time step to current update vector. [55]

The added vector is to be defined in `momentum_optimizer_value` parameter and it is usually set to 0.9 by Machine Learning community. The learning rate is chosen 0.002 and it will remain constant during the training.

In the next sections of the config file, TFRecord file addresses and path to the pre-trained model are given alongside with the number of steps for the training process. Additionally, images are flipped horizontally in a random sequence to augment the available data and help the model to generalize better.

The complete config file for Faster RCNN architecture is shown in listing [4.5](#).

Listing 4.5: FasterRCNN config file

```
model {
  faster_rcnn {
    num_classes: 2
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
  }
  first_stage_nms_score_threshold: 0.0
  first_stage_nms_iou_threshold: 0.7
  first_stage_max_proposals: 300
  first_stage_localization_loss_weight: 2.0
  first_stage_objectness_loss_weight: 1.0
  initial_crop_size: 14
  maxpool_kernel_size: 2
  maxpool_stride: 2
  second_stage_box_predictor {
    mask_rcnn_box_predictor {
      use_dropout: false
      dropout_keep_probability: 1.0
      fc_hyperparams {
        op: FC
        regularizer {
```

```
        l2_regularizer {
            weight: 0.0
        }
    }
    initializer {
        variance_scaling_initializer {
            factor: 1.0
            uniform: true
            mode: FAN_AVG
        }
    }
}
}
}
second_stage_post_processing {
    batch_non_max_suppression {
        score_threshold: 0.0
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 300
    }
    score_converter: SOFTMAX
}
second_stage_localization_loss_weight: 2.0
second_stage_classification_loss_weight: 1.0
}
}

train_config: {
    batch_size: 1
    optimizer {
        momentum_optimizer {
            learning_rate: {
                constant_learning_rate {
                    learning_rate: 0.002
                }
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}
gradient_clipping_by_norm: 10.0
fine_tune_checkpoint:
"gs://cargoshipbucket/faster_rcnn_inception_v2_newDSv2/data/model.ckpt"
from_detection_checkpoint: true
load_all_detection_checkpoint_vars: true
num_steps: 200000
data_augmentation_options {
    random_horizontal_flip {
    }
}
```

```

}

train_input_reader: {
  tf_record_input_reader {
    input_path:
      "gs://cargoshipbucket/faster_rcnn_inception_v2_newDSv2/data/train.record"
  }
  label_map_path:
    "gs://cargoshipbucket/faster_rcnn_inception_v2_newDSv2/data/label_map.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 100
}

eval_input_reader: {
  tf_record_input_reader {
    input_path:
      "gs://cargoshipbucket/faster_rcnn_inception_v2_newDSv2/data/test.record"
  }
  label_map_path:
    "gs://cargoshipbucket/faster_rcnn_inception_v2_newDSv2/data/label_map.pbtxt"
  shuffle: false
  num_readers: 1
}

```

For the SSD case, in the model section of the config file, the type of architecture is defined as SSD with version 2 of inception classification network as a backbone structure. Input images are resized to have 300 pixels by 300 pixels resolution. Similar to Faster RCNN scenario, L2 regularization layer is implemented but with a weight equals to $4e-5$. Network weights are initialized by a truncated normal distribution having 0 mean and 0.03 standard deviation. For activation function, ReLU6 is utilized which is shown in equation 4.1.

$$y = \min(\max(x, 0), 6) \quad (4.1)$$

Implementing this type of ReLU activation enables the sparse feature learning earlier and faster. [56]

A batch normalization layer is implemented to normalize the inputs of each layer within the network. This technique enables faster training and allows higher values of learning rate to be used during the network training. Convolution layer kernel size is set to 3. Grid anchors are generated by setting the `ssd_anchor_generator` parameter. Number of layers are defined to be 6, minimum scale and maximum scale are 0.2 and 0.95 respectively. Aspect ratios are set to 0.5, 0.333, 1 and 2. Non Max Suppression (NMS) is implemented on each batch of input data intersection threshold of 0.6 and maximum detection of 100. Detection scores are achieved by applying a sigmoid function to input scores. For the losses, the weighted smooth L1 loss is applied for localization loss and

weighted sigmoid loss is applied for classification loss. Furthermore, online hard example algorithm is exploited to reduce the number of false negative detections by simply using backpropagation on examples with the highest loss while keeping the ratio between true positive and false negative samples [57].

In the training configuration section, batch size is set to 16. Randomized horizontal flip is used for data augmentation. Additionally, the input images are cropped randomly to enhance the data augmentation. RMSProp optimizer is used with a constant learning rate of 0.002, Momentum optimizer value of 0.9 and epsilon value of 1.

In the remaining part of the config file, path to pre-trained network and TFRecord files are defined. Similar to Faster RCNN configuration, COCO detection metrics are chosen for the evaluation.

The complete config file for SSD architecture is shown in listing 4.6.

Listing 4.6: SSD config file

```
model {
  ssd {
    num_classes: 2
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
    feature_extractor {
      type: "ssd_inception_v2"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 4e-05
          }
        }
        initializer {
          truncated_normal_initializer {
            mean: 0.0
            stddev: 0.03
          }
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.9997000009823
        center: true
        scale: true
        epsilon: 0.00100000000475
        train: true
      }
    }
    use_depthwise: true
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
    }
  }
}
```

```
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
    }
}
similarity_calculator {
    iou_similarity {
    }
}
box_predictor {
    convolutional_box_predictor {
        conv_hyperparams {
            regularizer {
                l2_regularizer {
                    weight: 3.99999989895e-05
                }
            }
            initializer {
                truncated_normal_initializer {
                    mean: 0.0
                    stddev: 0.03
                }
            }
            activation: RELU_6
            batch_norm {
                decay: 0.999700009823
                center: true
                scale: true
                epsilon: 0.0010000000475
                train: true
            }
        }
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        use_dropout: false
        dropout_keep_probability: 0.800000011921
        kernel_size: 3
        box_code_size: 4
        apply_sigmoid_to_scores: false
        use_depthwise: true
    }
}
anchor_generator {
    ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.20
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
    }
}
```

```
        aspect_ratios: 3.0
        aspect_ratios: 0.333
    }
}
post_processing {
  batch_non_max_suppression {
    score_threshold: 0.30
    iou_threshold: 0.60
    max_detections_per_class: 100
    max_total_detections: 100
  }
  score_converter: SIGMOID
}
normalize_loss_by_num_matches: true
loss {
  localization_loss {
    weighted_smooth_l1 {
    }
  }
}
  classification_loss {
    weighted_sigmoid {
    }
  }
}
  hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 3
  }
  classification_weight: 1.0
  localization_weight: 1.0
}
}
}
train_config {
  batch_size: 16
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
  data_augmentation_options {
    ssd_random_crop {
    }
  }
}
optimizer {
  rms_prop_optimizer {
    learning_rate {
      constant_learning_rate {
        learning_rate: 0.002
      }
    }
  }
}
```

```
    }  
  }  
  momentum_optimizer_value: 0.899999976158  
  decay: 0.89  
  epsilon: 1.0  
}  
}  
fine_tune_checkpoint:  
  
  "gs://cargoshipbucket/model-ssdinceptionv2-2/data/model.ckpt"  
  num_steps: 200000  
  fine_tune_checkpoint_type: "detection"  
}  
train_input_reader {  
  label_map_path:  
    "gs://cargoshipbucket/model-ssdinceptionv2-2/data/label_map.pbtxt"  
  tf_record_input_reader {  
    input_path:  
      "gs://cargoshipbucket/model-ssdinceptionv2-2/data/train.record"  
  }  
}  
eval_config {  
  num_examples: 200  
  metrics_set: "coco_detection_metrics"  
}  
eval_input_reader {  
  label_map_path:  
    "gs://cargoshipbucket/model-ssdinceptionv2-2/data/label_map.pbtxt"  
  shuffle: false  
  num_readers: 1  
  tf_record_input_reader {  
    input_path:  
      "gs://cargoshipbucket/model-ssdinceptionv2-2/data/test.record"  
  }  
}  
}
```

Since object detection training requires heavy computational power, it is highly recommended to perform the training in a high-end machine. Google provides a cloud-based infrastructure for developers to build, test and deploy their applications called Google Cloud. It is used by 4 million applications worldwide. Google cloud also enables the user to perform the Machine Learning training task via the Cloud Machine Learning Engine. Performing the training on the cloud takes a significantly shorter amount of time with respect to local machines due to the high-end hardware specifications available in the platform. Because of the cloud-based nature of the platform, projects implemented on Google Cloud are easily scalable. To store the required data (TFRecord files, configuration file, label map and model checkpoint), online storage is allocated by the Google Cloud platform. Another advantage of On Google Cloud is that it is feasible to make the prediction online. Moreover, Google offers better pricing options with respect to other competitors like Amazon or Microsoft.

Before running the training job on Google cloud, the number and types of the machine should be specified in a YAML configuration file. For both object detection architectures, the same YAML configuration is used and it is shown in listing 4.7.

Listing 4.7: Training Machine configuration

```
trainingInput:
  runtimeVersion: "1.9"
  scaleTier: CUSTOM
  masterType: standard_gpu
  workerCount: 2
  workerType: standard_gpu
  parameterServerCount: 1
  parameterServerType: standard
```

A custom machine configuration is defined for object detection. Two workers are utilized to improve the training speed on one server cluster. Machine type is set to standard and The type of GPU is standard_gpu. According to the Google Cloud website, a standard machine has 4 virtual CPUs and 15 gigabytes of memory. standard_gpu represents a single NVIDIA Tesla K80 GPU. The mentioned GPU has 24 gigabytes of memory and clock speed of 562 MHz with the possibility to boost up to 875 MHz. Full specifications of this GPU is shown on table 4.1.

Stream Processors	2 x 2496
Core Clock	562MHz
Boost Clock	875MHz
Memory Clock	5GHz GDDR5
Memory Bus Width	2 x 384-bit
VRAM	2 x 12GB
Single Precision	8.74 TFLOPS
Double Precision	2.91 TFLOPS
Transistor Count	2 x 7.1B
TDP	300W
Cooling	Passive
Manufacturing Process	TSMC 28nm
Architecture	Kepler

Table 4.1: NVIDIA Tesla K80 specifications

After all the configuration is done for the models and cloud machines, all of the required files such as datasets, configuration files, and label map must be uploaded on Google Cloud storages. Although many different solutions are available for online data storage on Google cloud platform, for object detection it is compulsory to use Google "bucket" which would be considered as a virtual hard drive.

To start the training process on cloud the command shown on listing 4.8 is executed on the local machine:

Listing 4.8: Tensorboard initialization

```
# From tensorflow/models/research/
gcloud ml-engine jobs submit training
object_detection_ 'date +%m%d_%Y_%H%M%S' \
--runtime-version 1.9 \
--job-dir=gs://{MODEL_DIR} \
--packages dist/object_detection-0.1.tar.gz, slim/dist/slim-0.1.tar.gz,
/tmp/pycocotools/pycocotools-2.0.tar.gz \
--module-name object_detection.model_main \
--region europe-west1 \
--config ${PATH_TO_LOCAL_YAML_FILE} \
-- \
--model_dir=gs://{MODEL_DIR} \
--pipeline_config_path=gs://{PIPELINE_CONFIG_PATH}
```

Since each training should have a unique name, the exact time of command execution is adopted for naming conventions. `${PATH_TO_LOCAL_YAML_FILE}` is the local path to the YAML configuration, `gs://{MODEL_DIR}` specifies the directory on Google Cloud Storage where the training checkpoints and events will be written to and `gs://{PIPELINE_CONFIG_PATH}` points to the model configuration stored on the Google Cloud Storage.

To visualize the train and evaluation progress, we use the built-in Tensorflow visualization framework named Tensorboard. Having a built-in visualization framework is a massive advantage of using Tensorflow. The following command initializes the Tensorboard.

Listing 4.9: Tensorboard initialization

```
tensorboard --logdir= gs://{MODEL_DIR}
```

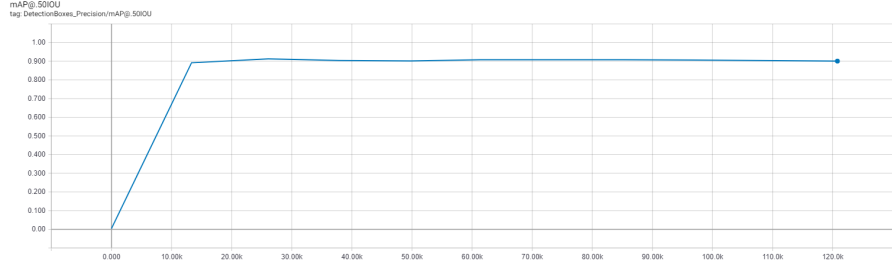
`gs://{MODEL_DIR}` specifies the directory on Google Cloud Storage where the training checkpoints and events are written. Commands used to train the network

4.3 Object detection results

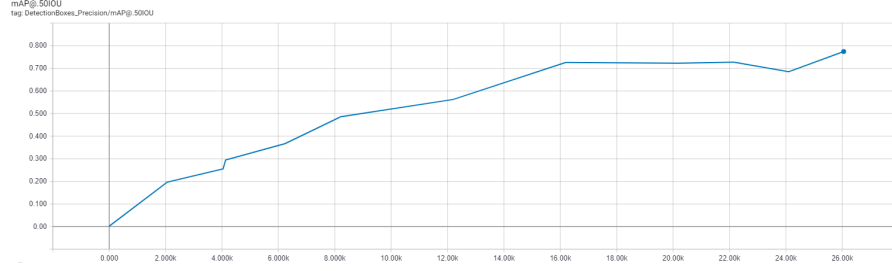
In the previous section, all the required preprocessing is done and configurations are set. In this section, the evaluation graphs will be presented based on COCO dataset metrics. Models will be compared and the best one will be chosen to be implemented for this project. To have a fair comparison, both models are trained on the same machine with the same hardware configuration implemented in the YAML file. Since boats and ships are taking a small portion of our images, mean Average Precision (mAP) at 0.5 Intersection over Unit (IoU) is selected as an accuracy measure.

Mean Average Precision (mAP) of SSD and faster RCNN architectures are shown in 4.4.

In the case of Faster RCNN, final mAP value is 0.9125 and for SSD architecture, it is 0.7743. It is clear from the numbers that Faster RCNN has better accuracy. This result



(a) Faster RCNN



(b) SSD

Figure 4.4: Mean Average Precision for both architectures

is in accordance with a well-known issue of SSD network. As in chapter 2, SSD network struggles in detecting small object because of utilizing a limited set of default boxes in its architecture. On the other side, Region Proposal Network in Faster RCNN allows for better performance regarding small object detection.

In figure 4.5 four losses of Faster RCNN model are illustrated. As said in chapter 2, two losses related to RPN network are objectness loss and localization loss. Two other losses are related to stage box classifier section of Faster RCNN and they are classification and localization loss.

As shown losses are decreased from their initial value and plateaued after 15 thousands of steps. The respected values for losses are 0.1853 for classification loss, 0.08785 for localization loss, 0.07369 for RPN localization loss and 0.06459 for RPN objectness loss.

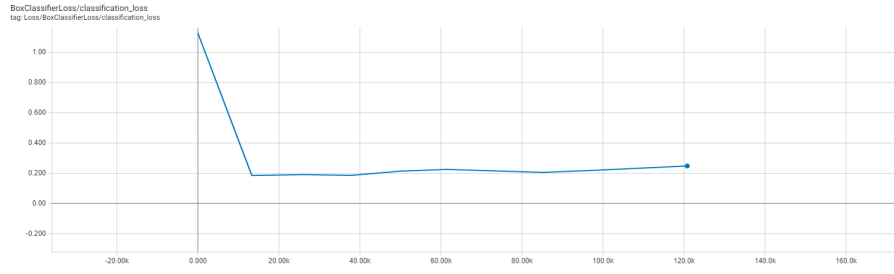
The total loss which is the addition of above-mentioned losses is displayed in figure 4.6 In this figure, the loss trend is decreasing and reaches the minimum value of 0.4286

In SSD network architecture, two types of losses are defined: classification loss and localization loss. They are shown in 4.7. Localization loss is reducing up to the point that it reaches its minima in 5.08. Classification loss is decreasing up to 5.08 and afterward, it starts to increase slightly. Moreover, since the total loss is the addition of two losses, it follows the same pattern and reaches its minimum value at 6.6654. Total loss is expressed in figure 4.8.

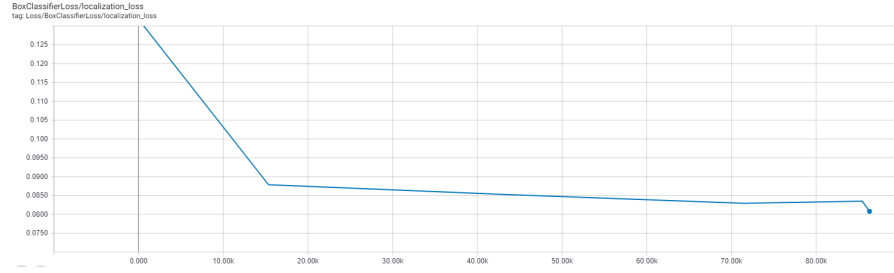
Based on the results mentioned, Faster RCNN is having a better accuracy and performance. Thus it is selected as a baseline for ship detection. Some samples of ship detection implementing Faster RCNN is given in figure 4.9. The ground truth data are presented on the right side of the image and actual detection is presented on the left side.

Furthermore, the developed algorithm was applied to real images taken from satellite

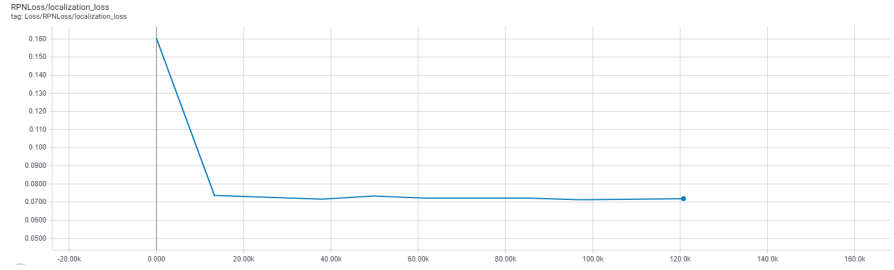
4.3 – Object detection results



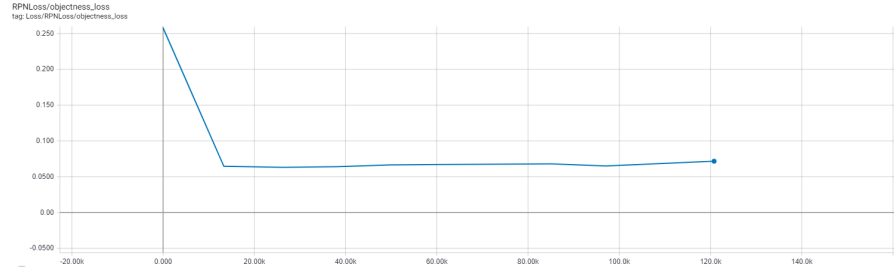
(a) Classification loss



(b) Localization loss



(c) RPN localization loss



(d) RPN objectness loss

Figure 4.5: Faster RCNN losses

images. Even though our object detector was trained on a synthetic dataset, it was able to detect ships with a good performance. Using the synthetic dataset approaches enables us to be able to perform deep learning solutions to the vast variety of problems in which a standard dataset is not available for training. Some samples of ship detection on real images are given in figure 4.10.

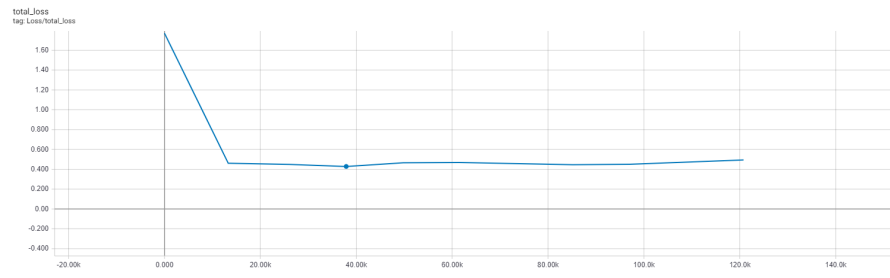
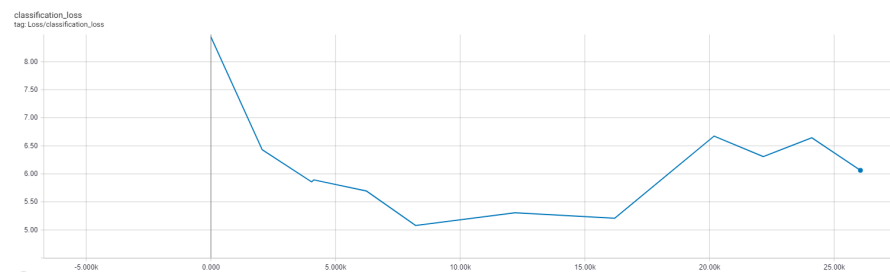
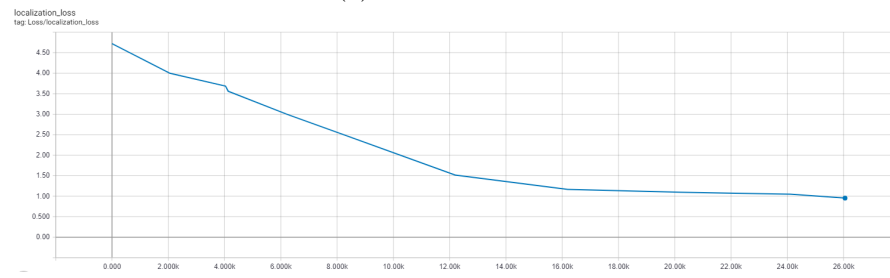


Figure 4.6: Faster RCNN total loss



(a) Classification loss



(b) Localization loss

Figure 4.7: SSD losses

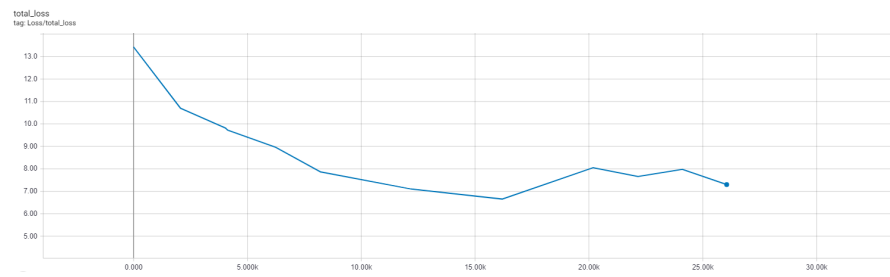


Figure 4.8: SSD total loss

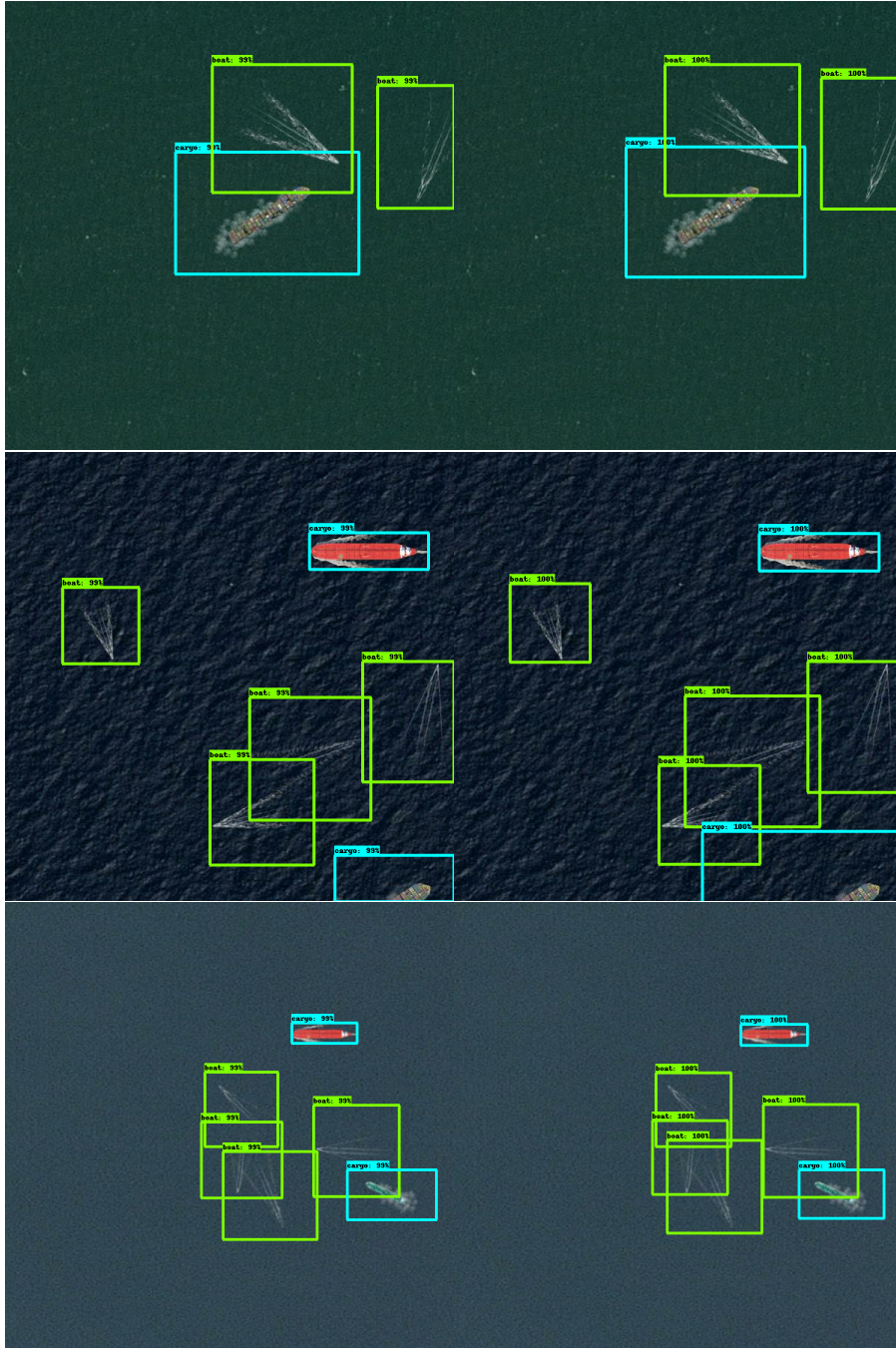


Figure 4.9: Ship detection by Faster RCNN



Figure 4.10: Ship detection done on real images using Faster RCNN

Chapter 5

Conclusions

After a brief introduction to artificial intelligence, Machine Learning, and Deep Learning and their applications in our world, two states of the art object detection named SSD and Faster RCNN were studied. Due to the lack of available dataset for ship detection, a synthetic dataset was developed to overcome the issue. Both networks were trained with a synthetic dataset containing cargo ships and boats. Because of better accuracy and ability to detect smaller ships, Faster RCNN was chosen for the special case of ship detection. The whole system has the potential to be implemented on satellites to detect the ships and boats on images produced by satellites. By implementing the proposed solution, it would be possible to detect the boats having suspicious behavior and propose a boat having a high probability of being a pirate. In this fashion, a warning signal could be sent to ship and authorities. Therefore initial countermeasures could be done to prevent the attack or reinforcement could be sent to aid in case of attack.

5.1 Further work

In today's world, no method is 100 percent perfect and this proposed method is not an exception. There are many parts which could be improved in our solution.

First and foremost, the proposed system could have a significantly better performance if it would be trained on a dataset containing real-world images from ships and boats and not a synthetic dataset.

In each and every Machine Learning problem, hyperparameters play an important role and the same goes for our problem of ship detection. After training the network on real world data, optimized hyperparameters could be found to enhance the performance of the system.

Since our project is developed in Tensorflow framework, it would be attainable to implement it on an embedded platform to be utilized on satellites to perform online detection.

An alternative to the above solution would be implementing the system on cloud platforms to increase the execution speed and security of the system. In this manner, images taken from the satellite will be transmitted to the cloud for detection. Due to the protected infrastructure of the cloud computing platform, the security is improved

compared to the implementation of the system directly on satellites. A drawback of this solution is the time spent to transfer the images between satellites and cloud servers.

Furthermore, a recently published paper proposed a novel Deep Learning architecture called NASNet. Indicated network shows promising results when implemented as a Faster RCNN backbone on COCO dataset and has the best results by the time of writing this thesis. [58]

This project is a simple endeavor to investigate the potentials and applications of Deep Learning algorithms to solve object detection problems utilizing aerial images taken from satellites.

the developed algorithm was applied to real images taken from satellite images. Even though our object detector was trained on a synthetic dataset, it was able to detect ships with a good performance. Using the synthetic dataset approaches enables us to be able to perform deep learning solutions to the vast variety of problems in which a standard dataset is not available for training.

Bibliography

- [1] “Merriam-webster.com,” 2018.
- [2] M. Hilbert and P. López, “The world’s technological capacity to store, communicate, and compute information,” *Science*, vol. 332, no. 6025, pp. 60–65, 2011.
- [3] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence*. New York, NY, USA: Basic Books, Inc., 1993.
- [4] K. Grace, J. Salvatier, A. Dafoe, B. Zhang, and O. Evans, “When will AI exceed human performance? evidence from AI experts,” *CoRR*, vol. abs/1705.08807, 2017.
- [5] D. L. Poole, A. K. Mackworth, and R. Goebel, *Computational intelligence: a logical approach*, vol. 1. Oxford University Press New York, 1998.
- [6] J. McCarthy, “Definition of ai,” 2007.
- [7] R. Kurzweil, *The Singularity Is Near: When Humans Transcend Biology*. Viking Press, 2005.
- [8] I. J. Good, “Speculations concerning the first ultraintelligent machine,” in *Advances in computers*, vol. 6, pp. 31–88, Elsevier, 1966.
- [9] A. T. Mehryar Mohri, Afshin Rostamizadeh, *Foundations of Machine Learning*. MIT Press, 2012.
- [10] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of Research and Development*, vol. 44, pp. 206–226, Jan 2000.
- [11] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.
- [12] N. Le Roux, Y. Bengio, and A. Fitzgibbon, “15 improving first and second-order methods by modeling uncertainty,” *Optimization for Machine Learning*, p. 404, 2011.
- [13] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2010.
- [14] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, vol. 2. Athena Scientific, 2012.
- [15] L. Deng, D. Yu, *et al.*, “Deep learning: methods and applications,” *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.
- [16] L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” in *Advances in neural information processing systems*, pp. 161–168, 2008.
- [17] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.
- [18] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, “Advances in optimizing recurrent networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8624–8628, May 2013.
- [19] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for

- lvcsr using rectified linear units and dropout,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8609–8613, IEEE, 2013.
- [20] I. Aleksander, M. De Gregorio, F. M. G. França, P. M. V. Lima, and H. Morton, “A brief introduction to weightless neural systems,” in *ESANN*, pp. 299–305, Citeseer, 2009.
- [21] Y. You, B. Aydin, and D. James, “Scaling deep learning on gpu and knights landing clusters,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 9, ACM, 2017.
- [22] D. Marr, *Vision: A computational investigation into*. WH Freeman, 1982.
- [23] D. G. Lowe, “Three-dimensional object recognition from single two-dimensional images,” *Artificial intelligence*, vol. 31, no. 3, pp. 355–395, 1987.
- [24] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [25] P. Viola and M. J. Jones, “Robust real-time face detection,” *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [26] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.
- [27] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.
- [28] N. Yokoya and A. Iwasaki, “Object detection based on sparse representation and hough voting for optical remote sensing imagery,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 8, no. 5, pp. 2053–2062, 2015.
- [29] J. Han, P. Zhou, D. Zhang, G. Cheng, L. Guo, Z. Liu, S. Bu, and J. Wu, “Efficient, simultaneous detection of multi-class geospatial targets based on visual saliency modeling and discriminative learning of sparse coding,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 89, pp. 37–48, 2014.
- [30] X. Jin and C. H. Davis, “Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks,” *Image and Vision Computing*, vol. 25, no. 9, pp. 1422–1431, 2007.
- [31] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, “Vehicle detection in satellite images by hybrid deep convolutional neural networks,” *IEEE Geoscience and remote sensing letters*, vol. 11, no. 10, pp. 1797–1801, 2014.
- [32] Q. Jiang, L. Cao, M. Cheng, C. Wang, and J. Li, “Deep neural networks-based vehicle detection in satellite images,” in *Bioelectronics and Bioinformatics (ISBB), 2015 International Symposium on*, pp. 184–187, IEEE, 2015.
- [33] P. Zhou, G. Cheng, Z. Liu, S. Bu, and X. Hu, “Weakly supervised target detection in remote sensing images based on transferred deep features and negative bootstrapping,” *Multidimensional Systems and Signal Processing*, vol. 27, no. 4, pp. 925–944, 2016.
- [34] L. Zhang, Z. Shi, and J. Wu, “A hierarchical oil tank detector with deep surrounding

- features for high-resolution optical satellite imagery,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4895–4909, 2015.
- [35] I. Ševo and A. Avramović, “Convolutional neural network based automatic object detection on aerial images,” *IEEE geoscience and remote sensing letters*, vol. 13, no. 5, pp. 740–744, 2016.
 - [36] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, “Orientation robust object detection in aerial images using deep convolutional neural network,” in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 3735–3739, IEEE, 2015.
 - [37] G. Cheng, P. Zhou, and J. Han, “Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, 2016.
 - [38] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
 - [39] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
 - [40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
 - [41] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
 - [42] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *European conference on computer vision*, pp. 346–361, Springer, 2014.
 - [43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
 - [44] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, pp. 21–37, Springer, 2016.
 - [45] S. Re, “Global insured losses caused by maritime disasters from 2007 to 2017 (in million u.s. dollars).”
 - [46] H. S. News, “Number of pirate attacks against ships worldwide from 2009 to 2017.”
 - [47] W. Langewiesche, “Anarchy at sea,” *The Atlantic Monthly*, vol. 292, no. 2, pp. 50–70, 2003.
 - [48] D. Damon, “Judith tebbutt: My six months held hostage by somali pirates,” *BBC News Magazine*, vol. 26, 2013.
 - [49] T. Besley, T. Fetzer, and H. Mueller, “The welfare cost of lawlessness: evidence from somali piracy,” *Journal of the European Economic Association*, vol. 13, no. 2, pp. 203–239, 2015.
 - [50] J. W. Miller, “Loaded: Freighters ready to shoot across pirate bow.”
 - [51] T. Eriksen, A. N. Skauen, B. Narheim, Ø. Hellere, Ø. Olsen, and R. B. Olsen, “Tracking ship traffic with space-based ais: Experience gained in first months of

- operations,” in *Waterside Security Conference (WSS), 2010 International*, pp. 1–8, IEEE, 2010.
- [52] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, *et al.*, “Sentinel-2: Esa’s optical high-resolution mission for gmes operational services,” *Remote Sensing of Environment*, vol. 120, pp. 25–36, 2012.
- [53] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, pp. 740–755, Springer, 2014.
- [54] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- [55] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.
- [56] A. Krizhevsky and G. Hinton, “Convolutional deep belief networks on cifar-10,” *Unpublished manuscript*, vol. 40, no. 7, 2010.
- [57] A. Shrivastava, A. Gupta, and R. Girshick, “Training region-based object detectors with online hard example mining,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016.
- [58] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.