ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AIRPLANE DETECTION AND IDENTIFICATION BASED ON MASK REGION CONVOLUTION NEURAL NETWORK

M.Sc. THESIS

Waleed AL-SHAIBANI

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Dr. Mustafa HELVACI

July 2020

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AIRPLANE DETECTION AND IDENTIFICATION BASED ON MASK REGION CONVOLUTION NEURAL NETWORK

M.Sc. THESIS

Waleed AL-SHAIBANI
(705181019)

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Dr. Mustafa HELVACI

July 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

ÜZERİNE GÖRE UÇAK TESPİT VE TİP TANIMLAMA MASK REGION CONVOLUTION NEURAL NETWORK

YÜKSEK LİSANS TEZİ

Waleed AL-SHAIBANI
(705181019)

İletişim Sistemleri Anabilim Dalı

Uydu Haberleşmesi ve Uzaktan Algılama Programı

Tez Danışmanı: Dr. Mustafa HELVACI

Temmuz 2020

Waleed AL-SHAIBANI, a M.Sc. student of ITU Informatics Institute student ID 705181019, successfully defended the thesis entitled “AIRPLANE DETECTION AND IDENTIFICATION BASED ON MASK REGION CONVOLUTION NEURAL NETWORK”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : Dr. Öğr. Üyesi. Mustafa HELVACI .....

Istanbul Technical University

Jury Members : Prof.Dr. Ertuğrul Karaçuha .....

Istanbul Technical University

:Prof.Dr. Bahadir K. GUNTURK .....

Medipol University

Date of Submission : 10/06/2020

Date of Defense : 06/07/2020

To my family,

FOREWORD

I am writing these words based on the literal and general meaning of "FOREWORD" word; as the author's closest friend.

I am accomplishing a master thesis, which considered an advanced step in my plan. The ultimate goal of this plan is reaching to researchers', academics' and scientists' area in the technology filed.

Doing a master thesis in the most repeated technological term in the research area like "object detection" and "deep learning" considered an enjoyable big challenge for me. This enjoyment led me to accomplish good results in this field.

With every single step through obtaining results related to this work, there were my first guiders, first teachers, the inspiration sources, and my life models; they are my parents, TAWFIK and FATIMA, who support me not just through this work. Still, they do that every moment in my whole life. Big thanks to them.

Many thanks to him to my supervisor Dr. MUSTAFA HELVACI, who guided me with respect, helpful, friendly guidance and dealing. He gave me his support without putting extra pressure. He gave me a lot of convincing and encouraging words.

To my family, friends, teachers, and every person who is around me, thanks a lot for your support, really your kind words are pushing me forward.

July 2020 WALEED TAWFIK AL-SHAIBNAI

TABLE OF CONTENTS

Page

ABBREVIATIONS

ANN : Artificial Neural Network
ATR : Automatic Target Recognition
BP : Backpropagation
Ce : Cross-entropy
CNN : Convolutional Neural Network
COCO : Common Objects in Context
CPU : Central Processing Unit
CUDA : Computer Unified Device Architecture
CUDNN : The NVIDIA CUDA Deep Neural Network Library
DL : Deep Learning
Faster RCNN: Faster Region-based Convolutional Network
GPU : Graphics Processing Unit
MSE : Mean Square Error
RAM : Random Access Memory
RBM : Restricted Boltzmann Machines
ReLU : Rectified Linear Unit
RGB : Red, Green, and Blue
RMSE : Root Mean Square Error
RNN : Recurrent Neural Network
RS : Remote Sensing
SGD : Stochastic Gradient Descent
Sig(x) : Sigmoid function applied on x
SVM : Support Vector Machine

LIST OF TABLES

Page

LIST OF FIGURES

Page

AIRPLANE DETECTION AND IDENTIFICATION BASED ON MASK REGION CONVOLUTION NEURAL NETWORK

SUMMARY

Traveling is a critical need for humans. Throughout their lives, humans need to travel for many reasons such as improving their lives, escaping from imminent danger, visiting their relatives, stuff transporting, and emergency cases such as giving someone help or rescue others. There are different ways for traveling and stuff transportation; air, land, and sea, but when comparing them through safety, cost, and speed, the best choice for most people is flying.

The number of flights performed all-inclusive by the aircraft business increased consistently since the mid-2000s, and it is expected to reach 40.3 million this year. So to handle such dramatical growth, a lot of new fight routes are generated, which puts massive pressure on airports and causes high traffic jams at airports.

To point this status, let us take Turkey as a case of study. Turkey has more than 57 airports; most of them are international airports. To be more specific, let us take the most critical and modern airport, Istanbul airport. This airport served more than 50 million passengers last year, which is a considerable number that requires a huge effort in organizing the air traffic. The point which is essential here is that this number is continually increasing and is expected to reach 60 million in the future, which seems that it could be handled according to modern technology used in this airport, but what if this number keeps increasing!

The increasing number of flights performed per day puts excessive pressure on the air traffic control system. Thus, providing a method to support ground traffic control is desirable.

.

One of the methods used to solve this issue is airplane detection using satellite images with a deep learning approach where the dataset is collected from public satellite sources[14]. Using such methods is not a universal solution because they are using satellites as a source of their data. Also, preparing satellite images takes a long time to prepare them using image processing techniques. Thus, these approaches have a high cost and a long operating time. In the method used here, the drones dataset is used, which is cheap to be acquired and fast to be processed, so any country in the world, especially those that do not have modern technology to own a satellite, can use this solution to help them control the traffic jam.

MASK RCNN algorithm is the technique used here. It has a faster R-CNN as its building base with simple critical modifications on its head neural network constructions. Then it adds masks estimation, which allows us to detect whether there is an airplane or not, and helps in the estimation of the surface area and length of each plane so that the traffic control will be more accurate.

In terms of image visualizations, there are perfect results of detection within scores of 100%, 90s%, and 80s%. And there are some with low score detections, such as 50s%. The evaluation gave promised values in terms of COCO metrics compared with other researchers' results.

ÜZERİNE GÖRE UÇAK TESPİT VE TİP TANIMLAMA MASK REGION CONVOLUTION NEURAL NETWORK

ÖZET

Seyahat etmek insanlar için kritik bir ihtiyaçtır. İnsanlar yaşamlarını geliştirmek, tehlikelerden kaçmak, akrabalarını ziyaret etmek, eşyalarını taşımak ve acil durumlarda birilerine yardım ulaştırmak veya onları kurtarmak gibi birçok nedenle yaşamları boyunca seyahat etmeye ihtiyaç duyarlar. Seyahat etmek ve eşya taşımacılığı için hava, kara ve deniz gibi farklı yollar vardır; ancak bu yolların güvenlik, fiyat ve hız konusunda değerlendirilmesi söz konusu olduğunda, insanlar için en iyi seçenek havacılıktır.

2000'li yılların ortasından itibaren, hava taşıtı işletmeleri tarafından uygulanan her şey dahil uçak seferleri de devamlı olarakartmıştır ve bu uygulamaların 40.3 milyona varması beklenmektedir. Uçak seferlerindeki bu çarpıcı büyüme, yeni varış noktaları ortaya çıkarır. Bu, ayrıca havalimanlarına büyük baskı yükler ve uçak seferlerinde trafik sıkışıklığına neden olur.

Türkiye'yi bu durumun önemine işaret etmek için çalışma örneği olarak ele alalım. Türkiye çoğunluğu uluslararası olmak üzere 57'den fazla havalimanına sahiptir. Daha kesin ve açık olması için en kritik ve modern havalimanı olan İstanbul Havalimanı'ndan bahsedelim. İstanbul Havalimanı geçen yıl 50 milyon yolcuya hizmet vermiştir, ki bu büyük bir sayıdır ve hava trafiği organizasyonunda büyük bir çaba gerektirir. Buradaki asıl önemli nokta ise bu sayı hala artmaktadır. Araştırmalara göre, bu sayı 60 milyonlara varabilir, ve görülen o ki havalimanında kullanılan modern teknoloji de bu durumla baş edebilir; ancak ya bu sayı artmaya devam ederse, ne olur!

Günden güne artan uçak seferleri sayısı hava trafik kontrol sistemleri üzerinde baş edilmesi güç, büyük bir baskı yaratır, bu yakın gelecekte de artacağı tahmin edilen bir sayıdır, nitekim hava trafik kontrolü için farklı bir method temin etmek bu günlerde istek uyandırmaktadır.

Bu sorunu çözmek için kullanılan yaklaşımlardan biri, uydu görüntülerini derin öğrenme yaklaşımıyla ve kamu uydu kaynaklarından toplanan veri kümeleriyle birlikte değerlendiren uçak algılama yöntemidir [14]. Bu tür çözüm yöntemleri verilerinin kaynağı olarak uydu kullandıklarından dolayı evrensel değildir. Bununla birlikte, uydu görüntülerini hazırlamak görüntü işleme teknikleri kullanımı yüzünden çok zaman alır, dolayısıyla bu çözüm maliyette ve zamanda kayıp içerir. Burada, insansız uçak veri kümesini kullanan yöntem yerine, dünya üzerindeki kendi uydusu için modern teknolojiye sahip olmayan herhangi bir ülke bu çözümü uçuşlardaki trafik sıkışıklığında kendilerini güvenli hale getirmek için kullanabilir.

Burada kullanılan teknik daha hızlı RCNN'yi baş sinir ağ yapılarında basit, kritik değişikliklerle yapı tabanı olarak alır ve uçak olup olmadığını tespit etmeye izin veren maske kestirimi ekler, yüzey alanı ve her uçağın uzunluğunu kestirir, buMASK RCNN algoritmasıdır, böylece trafik kontrol doğru ve kesin olur.

Görüntü görüntüleme açısından 100%, 90% ve 80% skorlarına varan mükemmel algılama sonuçları vardır; ancak %50 gibi düşük bazı skorlar da mevcuttur. Değerlendirme COCO metrik açısından, uydu görüntüleri kullanan diğer araştırmacıların sonuçları ile kıyaslandığında vaat edilen değerleri vermektedir.

Araştırmanın benzersiz yönü, burada kullanılan tekniğin dünyanın herhangi bir havaalanında uygulanabilmesidir. Havalimanının ihtiyaçları doğrultusunda belirlenen, her zaman diliminde sisteme sabit görüntüler sağlayan insansız hava araçlarına havalimanının belirli yerlerinde gerek duyulmaktadır. Diğer teknikler bunu yapabilmek için maliyet ve karmaşıklık açısından büyük farka neden olan uydulara ihtiyaç duyarlarken, insansız hava araçları herkes tarafından her yerde kullanılarak bu görüntüleri elde edebilir. Uydu ile görüntüleri elde etmek, farklı gereksinimlere cevap vermek ve çalışmak daha zordur.

Burada kullanılan teknik, havaalanındaki her bir uçağın yüzey alanını kesin olarak belirlemede önemli bir avantaj sağlar. Bu avantaj, son derece hassas uçuş kontrolüne

imkan verir. Havaalanı görevlileri uçağın yüzey alanı hakkında bilgi sahibi olarak, uçak tipini, üreticisini ve özelliklerini belirleyebilirler.

Bu çalışmada esas olarak insansız hava uçak görüntüleri veri kümesi olarak kullanılmıştır. Dolayısıyla, doğrudan görüntülerle ilgili bir çalışma yoktur, ek olarak bir kıyaslama noktası oluşturmak için uydu görüntüleri de değerlendirilmiştir. Bu çalışma, uydu görüntüleri kullanılarak yapılan çalışmalara kıyasla yakın sonuçlar vermiştir.

1. INTRODUCTION

Object detection is a process that aims to differentiate between objects based on data extracted from images or videos. Different objects have different patterns that make them different. For frames that contain random patterns, they considered as noise, so the ability to separate these patterns and identify the object is the ultimate goal for target detection.

Specialists and image processing experts always want to identify whether an object exists or not. For example, petrol engineers look for oil in the same area on earth. Laboratory guys search for disease, and weather guys work to find actions or conditions to determine or predict the future weather situation.

The basic idea is comparing data with a threshold so that they can determine an object's existence or action's happening If the limit is exceeded. The basic clear example is the detection process that occurs in the human visual system. This process looks for signatures that make objects different than background. And this difference between background and objects must be more significant than the threshold to be detectable. In technology, a clear example of that is a Facebook automatic tagging; it could recognize and identify faces directly from images.

Going deeply into image processing and understand its needs is not just concerning classification different images, but also put great attention on finding a precise estimation of objects' locations and their concepts. This complicated task is the function of object detection. That's why this function always divided into different subtasks like face detection, pedestrian detection, and skeleton detection. Object detection gives us high accurate information for semantic understanding of images, videos, or any such kind of data.

Working in this field develops indeed its building block, which is the neural network (NN). The neural network considered a building block for different kinds of learning systems, including deep learning (DL). This development in neural network effects a positive impact on object detection techniques.

In fact, from object detection's function requirements, there is a specification for two problems. The first one is determining where objects are located in an image, which is called object localization, and the second one is object classification. These two problems construct DL's network and configure its pipeline model.

The pipeline constructed from three stages. Informative region selection is the first stage, which focuses on different objects with different sizes and aspect ratios. It is done by scan the whole image with a sliding window. The second stage is feature extraction to recognize different objects to extract visual features. These because these features can produce representations associated with complex cells in the human brain. The last stage is classification to distinguish the target object from all other categories[1].

Considering milestones that make object detection DL-based more clear by focusing on most significant publications, which led to surprising results nowadays. As mentioned above, there is a pipeline for DL's model. The first stage in this pipeline is the region proposals, regions of interest(RoI), which aim to generate region on an image that could belong to an object. They use a selective research algorithm that generates sub-segmentations of images based on color, texture, size, and shape.

The next step is feature extraction from region proposals, which uses a convolution neural network (CNN) to generate a multidimensional feature vector for each region proposal for every image. The training of CNN passes through supervised pre-training now called (AlexNet) which has five convolutional two fully connected layers. It has been trained through image classification among a large number of images so it can learn basic features. Then the CNN passed through fine-tuned to

learn to identify classes belonging to the detection task from region proposals. Then the final classification from pre-training replaced with the N+1 SOFTMAX classification layer, here N represents object classes, and one is for the background.

Each image input converts into a fixed-size by fixed aspect ratio. And the labels for training are done by authors map each object proposal to a ground-truth instance with which has max intersection over union (IoU) overlap. The IoU has a threshold, so if this threshold exceeded one box, then the rest boxes treated as background class. The training of pipelines done by using stochastic gradient descent (SGD) with an approximately one-tenth learning rate. It is doing iteration for every positive and negative window over all classes. The negative ones are the background classes.

The second stage is object classification. Some of the most popular publications use a support vector machine classifier which used to detect whether the object exists or not. This stage consists of feature vector input that produced from the first stage of the region proposal. It labels each IoU overlap less than the least threshold with negative and any IoU overlap grater than that threshold positive. Finally, the third stage is the bounding box regression to improve localization performance [1][2][3].

Based on what mentioned above, it is fair enough to summarize the most building unit techniques work as following: Region-based CNN (RCNN), it takes an input image then does extraction around 2000 bottom-up region proposals after that, it computes features for each proposal using a vast convolutional neural network (CNN), finally classifies each region using SVMs. R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010 [2].

Fast R-CNN is no longer using SVMs just uses CNNs (single-stage). It does one single feature extraction per image instead of doing it per region, making it much faster 9x faster at training, 213x quicker at test[3]. Faster RCNN introduces RPNs (Region Proposal Networks), which are one type of fully convolutional network. They take an image of any size as input and output a set of rectangular object

proposals, each with an objectness score. Using an RPN instead of SS (selective search) slightly improved mAP from 66.9% to 69.9% on Pascal VOC[4].

MASK RCNN continues from where faster RCNN left and added some augmentations aim to make-instance segmentation. It has been designed for pixel-to-pixel alignment between network inputs and outputs by replacing RoIPool in faster RCNN by RoIAlign.Its result got AP 37.1 on COCO[5].

Based on this introduction about DL so far, this thesis aims to use DL technologies to introduce an efficient contribution to support flight control to decrease flight jams at the airport. Through using airplane detection with surface area estimation using drones images as a dataset.

1.1 Purpose of the Thesis

Due to fast development in technology, several facilities are increasing in each field of modern life which depend on technology. One of these facilities is fly traveling since it is the favored way to go for most people due to several things such as safety and speed.

Applicants' numbers increase dramatically to this service lead to an increase in airplanes used; consequently, these lead to flight traffic Jams. The flight ground control has difficulties once the traffic increases, So providing a method of traffic control to support current traffic control is desirable nowadays.

One of the methods used to solve this issue is airplane detection using satellite images with the DL approach with a dataset collected from public satellite sources[6]. Using such kind of methods is not a universal solution, because they are using satellite as a source of their data. Instead, This thesis uses drones dataset, so any country in the world, especially those countries which do not have modern technology or do not own satellite, can use this approach to solve fly traffic jam.

The technique used here is not just to detect whether there is an airplane or not. Still, also it determines the surface area of each plane, so the traffic control will be more accurate and could be joined to the database of airplane companies to know which type of aircraft depending on its surface area.

1.2 Unique Aspect

The technique used here could be applied at any airport in the world. It needs just drones at specific locations in airports to supply the system with fixed images every period determined by the airport's need. Whereas other techniques need satellites, so there is a vast difference in terms of cost and complexity. Drones are everywhere, and anyone can obtain them, but for satellite, it's hard somehow to purchase and operate due to complicated requirements and modern technology.

The technique used here also provides an essential advantage for precisely determines the surface area of each plane at the airport. This advantage will leads to highly accurate flight control. Drone images do not need extra pre-processing; they are typical images. These images could be forwarded directly into the system. In contrast, satellite images need pre-processing steps such as removing noises or graphical corrections, so this solution has less time to perform airplane detection task. By knowing the surface area and length of the airplane, the airport officers will be able to determine airplane type, name, manufacturer, and its specifications.

2. LITERATURE REVIEW

This chapter provides a comprehensive summary of concepts, methods, and selected studies in object detection. The goal is to provide reliable base information and gain powerful knowledge that allows reaching the ultimate purpose of this work.

2.1 Neural Network

They are computing systems vaguely inspired by the biological neural networks that constitute human brains. Such systems learn to perform tasks by considering examples, generally without being programmed with task-specific rules.

2.1.1 Perceptrons

It is a type of neurons that takes several inputs and produces a single binary output. It is the first and oldest building unit block of the neural network. The easiest way to compute its output had done by Rosenblatt, who proposed a simple rule to compute the output. He introduced weights ***(w_1, w_2 etc)*** and these weights are associated with inputs ***(x_1, x_2..etc).*** Weights represent the importance of such input to affect the output result.

Outputs have two values either ***"1"*** or ***"0"***, it could be calculated from a weighted sum $\sum_j \boldsymbol{w_j}\, \boldsymbol{x_j}$ concerning a threshold value. If the weighted sum is higher than the threshold value, then the output is ***"1"*** else output is ***"0"***.

It is better to change the summation notation $\sum_j \boldsymbol{w_j}\, \boldsymbol{x_j}$ to dot product notation $\boldsymbol{w}.\boldsymbol{x}$ where $\boldsymbol{w}$ and $\boldsymbol{x}$ are vectors, for simplicity purposes. Besides moving the threshold to the other side. Then dealing with it as bias instead of a threshold [7].

$$output = w.x + b \tag{2.1}$$

.

2.1.2 Sigmoid neurons

Learning's algorithm, in simple words, means small changes in weights and biases lead to small changes in output. This idea could not be addressed with perceptron, because small changes in weights or bias of any perceptron in the network can cause a complete change situation for perceptron's output; it may completely flip from ***"0"*** to ***"1"*** or reversed.

So there is a need for a smart way that enables small change in the weighted input to make corresponding small changes in output. The sigmoid neuron offers a solution to do that. It is the same as the perceptron except for its data and outputs; they are not just two values. Its inputs may be any value in the range **(0-1)** and also its outputs.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \textbf{(2.2)}$$

Sigmoid neurons could be considered as a smooth face for perceptron. In mathematics language, it could be represented in terms of graphs and equations, as shown below.

$$\nabla output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \qquad \textbf{(2.2)}$$

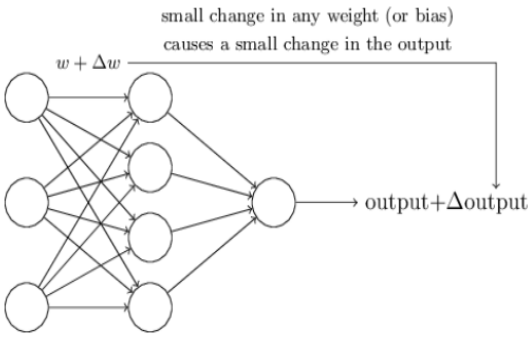


Figure 2.1: Learning algorithm [7].

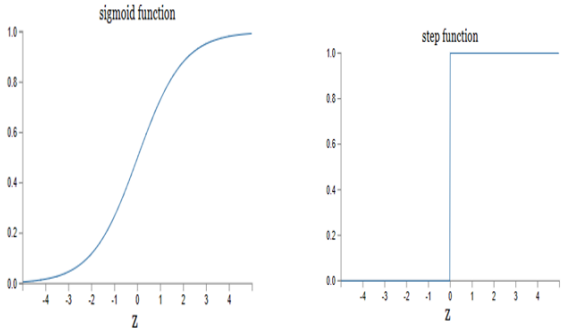


Figure 2.2: Sigmoid vs. perceptron neurons [7].

2.2 Methodology and Structure

The methodology and structure of a neural network could be best described through a handwritten digit's recognition example. The figure shown below is a handwritten number for ***3***, in a low-resolution picture of ***28 x 28*** pixels. However, the human brain has no problem recognizing it as a **3,** even if there are different types of writing styles. Although different writing styles mean different values for each pixel, the brain could consider this number is ***3*** easily.

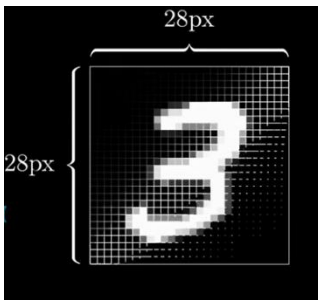


Figure 2.3: Handwritten digit with a low-resolution image [40].

Brain's functionality inspires engineers in developing neural networks. Like the brain, a neural network consists of neurons connecting. The neuron here defined as a thing that holds a decimal number from ***0-1*** to represents the grayscale value of the

pixels, then values **1s** are assigned to white pixels and **0s** for black pixels. These values called activations.

In a handwritten digit example, the image has ***28 x 28***-pixel resolution. In total, this image has ***784*** pixels, which, in turn, represent the input neurons for the neural network, and they construct the first layer, the input layer. The ultimate goal of the neural network is to distinguish numbers from ***0-9***, so the final layer formed by ten neurons, the output layer. Between the input layer and the output layer, there are essential layers called hidden layers.

The key idea which enables neural network to distinguish between these ten digits is that the activations in one layer determine the activation in the next layer, so a group of neurons which have fired cause the next group of neural to be fired.

The brightness values of the image fire the input neurons; each pixel has its brightness value. This step forms a firing pattern, then each pattern of firing or activation causes another trend in the next layer for activations, which causes another type of activations in the next layer, and so on until reaching the output layer. Finally, according to the values of activations, the maximum value represents the neural network decision about which digit is that.

The structure of the above neural network could be constructed from two hidden layers. According to this structure, the second hidden layer may recognize patterns of loops and sticks, which they build the digits. These patterns could be broken into basic ones. Such as circles could be grouped of different curves, lines could be formed from more than two small lines. Neurons of the first hidden layer may recognize these smaller curves and lines.

Figure 2.4: Neural network methodology and structure [40].

As a result, It could be generalized neural networkability of recognizing simple handwritten images to do recognition of complicated images. This conclusion relays on the fact of an image; each image consists of specific patterns; once recognizing these patterns, it acknowledges the whole image.

Figure 2.5: Recognizing a complicated image [40].

2.3 Learning Techniques

Learning operation in neural network aims to adjust neuron's weights of hidden layers. It means learning without being programmed, it learns from information examples and last data then processed current data in precision accuracy.

2.3.1 Unsupervised learning techniques

This technique does not use labeled data, so it has a save time advantage. It uses clustering ways to group samples according to how much they close to each other.K-means clustering, and Restricted Boltzmann Machines (RBMs) are well-known techniques [8]. Taking k-means clustering as a clear example to show how does these

techniques work, It creates k-number of centroids randomly with multiple iterations to find the best ones. Then any data close to the particular cluster centers, centroid, will be assigned to that cluster.

$$c^{(i)} = argmin_j \left\| x^2 - \mu_j \right\|^2 \quad \textbf{(2.4)}$$

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \quad \textbf{(2.5)}$$

Where $\boldsymbol{x^{(i)}}$ is the training set $\in \mathbf{R^{(n)}}$, $\mathbf{c^{(i)}}$ is a cluster label, and $\boldsymbol{\mu_j}$ the centroid.

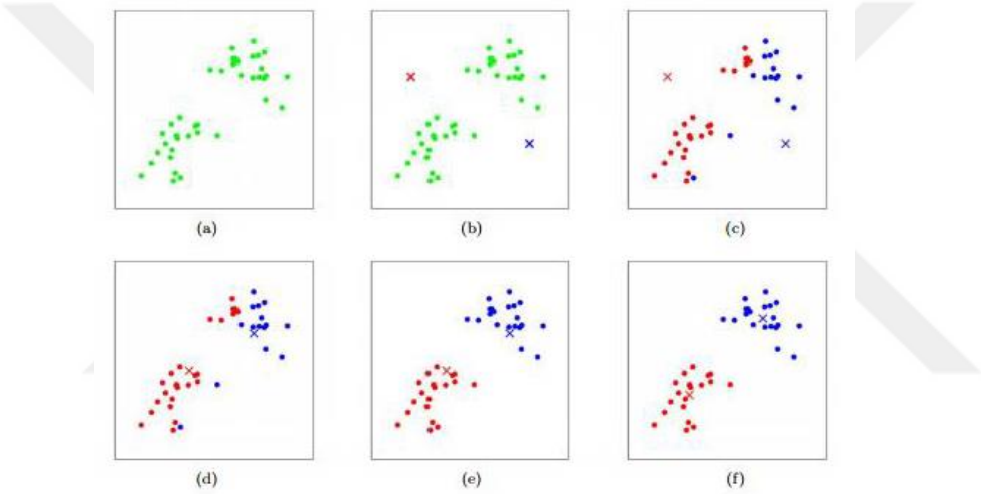


Figure 2.6: K-means cluster workflow [37].

Applications in remote sensing (RS) use these techniques, such as airplane detection using Deep Belief Network (DBN), which has been constructed by RBMs stacking [9]. Aircraft and car detection using the k-means clustering method through the Scale-Invariant Feature Transform (SIFT) features [10]. Unsupervised convolution neural network used for extracting spectral and spatial features [11]. Ships detection using optical satellite images with deep autoencoder structure [12].

2.3.2 Supervised learning techniques

This technique used training over manually labeled data, which is the main reason that makes supervised methods give more accurate results in the estimation of the new sample. The support vector machine (SVM) is the most popular technique. It used for classification tasks by developing a function that separates sample data into different categories. SVM aims to define a hyperplane between feature vectors extracted from samples and considering it as a margin that needs to be maximized.

$$f(x) = W^T x + b = y \tag{2.6}$$

If $y \geq 1 \rightarrow \forall x \in$ class 1, if $y \leq -1 \rightarrow \forall x \in$ class 2 :

$$Margin\ m = \frac{2}{\|m\|} \tag{2.7}$$

Convolution neural network (CNN) is an interesting supervised technique nowadays. It is a neural network that works with two-dimensional image data. The convolution process that takes place inside is the main reason for its naming. The convolution is a linear operation that implies multiplication of weights with input. These weights called a filter or kernel as usual.

Filters always are smaller than input in size. The dot product process takes place between filter and image. After multiplication, a summation process takes place to produce a single value. The filter goes through the entire image as a sliding window; thus, it discovers the feature anywhere in an image.

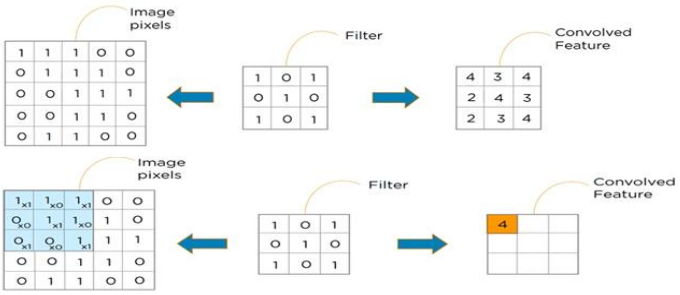


Figure 2.7: Convolution layer workflow [41].

There are different applications used in this field, like object detection in RS with Deep Network (HDNN) [13]. Using SVM to detect airports and airplanes [14]. As they did in ship detection, they used SIFT extracted previously to feed SVM using panchromatic images [15]. Using CNN for application on satellite images to aircraft and vehicle detection, like Zhong Yang, G.Hu, J.Han, L.Huang, they replaced the selective search by saliency. Due to many negative samples in the generated region proposal for RCNN, which will affect the model detection precision, saliency uses the human visual attention mechanism to achieve bottom-up object detection [16]. Lin Lei**,** T.tang, H.Zou, and Z. Deng treated with two problems in Faster RCNN. RPN layer has poor performance for small-sized objects localization, and a classifier is not reliable enough to distinguish vehicles and background, so they employed a hyper region proposal network (RPN) and replaced the classifier by a cascaded of boosted classifiers [17]. S.Zhang, R.Wu, K.Xu, and W.Sun, they introduced a fast region-based convolutional neural network (R-CNN) method to detect ships from high-resolution remote sensing imagery [18][19]. M.SOYDAŞ used a dataset collected from public satellite sources. He used SSD, faster RCNN, and YOLO object detection models, his results shown proper detection for faster RCNN and YOLO models according to COCO metrics [37] B. TRAORY used deep learning methods with high-resolution satellite images with two types of object detection models (SSD and faster RCNN). His dataset collected from public satellite sources such as WPU-RESISC45, WHURS19, and aerial image datasets. His results shown proper detection for SSD models according to COCO metrics. M.Khan, A.Yosef, and S.Nadeem used EdgeBoxes to produce proposals at the beginning, and then proposals have been filtered by geometric checking. Finally, CNN used for feature extraction and classification[38].

3. LEARNING ALGORITHMS

Neural networks have a lot of excitement, as they are quickly proving to be a promising and practical form of machine intelligence. This chapter simplifies the fundamentals of learning algorithms to get the idea behind the learning process.

3.1 Details on Detection Edges in The First Layer

Suppose that a neuron in the next layer after the input has to recognize edges from the input layer. The idea is assigning weights connecting each neuron in the input layer with the specified neuron. Then taking all activations of the input layer and computing their weighted sum.

$$X = w_1a_1 + w_2a_2 + w_3a_3 + \cdots w_na_n \quad \textbf{(3.1)}$$

where $\boldsymbol{w_n}$ represents weights and $\boldsymbol{a_n}$ represents activations. The input image is a pixels-based grid. Each pixel assigns its value to the input neuron. Some of these values are positives, and others are negatives, then equalizing those negative values with zeros will produce edges detection.

Computing a weighted sum might be any number in the number line. There is a need to make this value in the range (0-1) in the neural network. So there is a need for a function to squeeze results between zero and one. A sigmoid function could be used to do that.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \textbf{(3.2)}$$

$$\sigma\left(w_1a_1 + w_2a_2 + w_3a_3 + \cdots w_na_n\right) = \sigma(x) \quad \textbf{(3.3)}$$

Assuming that neuron needs to be lighted up when the weighted sum is more significant than ***10***, then it needs adding a bias ***10*** to the weighted sum before plugging it to the squeeze function.

$$\sigma\left(w_1a_1 + w_2a_2 + w_3a_3 + \cdots w_na_n + 10\right) \quad \textbf{(3.4)}$$

These processes are just for one neuron. Other neurons have the same number of processes. Of course, these steps are complicated and confusing, thus if the first layer after the input has ***16*** neurons and each one has ***784*** connection weights and ***16*** biases, then in total

$$784 \text{ x } 16 + 16 = 12560 \quad \textbf{(3.5)}$$

for the previous neural network which contains ***784*** x ***16*** x ***16*** x ***10***

$$(784 \text{ x } 16 + 16) + (16 \text{ x } 16 + 16) + (16 \text{ x } 10 + 10) = 13002 \quad \textbf{(3.6)}$$

3.2 Learning Process Details

Using linear algebra, it could organize all activations in the previous layer as a column vector, and all weights as a matrix within each row represent the connection for each neuron in the current layer to the last layer. It means taking the weighted sums for activations in the previous layer and assigns them to neurons in the current layer by arrays multiplication process. The next step is organizing biases as a column vector and adding them to the result. Then finally, enter all of them to a sigmoid function.

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} \cdots & w_{1,n} \\ w_{k,0} & w_{k,1} \cdots & w_{k,n} \end{bmatrix} \cdot \begin{bmatrix} {a_0}^0 \\ {a_1}^0 \\ {a_n}^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_n \end{bmatrix}\right) = a^1 \quad \textbf{(3.7)}$$

$$a^1 = \sigma(W^1 . a^0 + b^1) \quad \textbf{(3.8)}$$

Where $\mathbf{a^1}$ represents the activation vector of the second layer, $\boldsymbol{W}$ weights matrix, and $\boldsymbol{b}$ biases. It is beneficial for coding and makes it easier and faster. So, it would be useful to consider a neuron as a function instead of dealing with it as a thing that holds a number. Finally, the overall neural network is a complicated function that takes the input as activations and transforms it into output.

$$f(a_0, a_1 .. a_{783}) = \begin{bmatrix} y_0 \\ .. \\ y_9 \end{bmatrix} \quad \textbf{(3.9)}$$

It is essential to mention that sigmoid function not used frequently nowadays; because it makes the learning process slow. There is another function that fits the speed required in a neural network. This function called ***ReL;*** it does a rectification job with a linear behavior. It takes the maximum value between the zero and a specified amount.

$$ReLU(a) = \max(0, a) \quad \textbf{(3.10)}$$

So how does the neural network learn, what are the appropriate values for weights? At this moment, we think that each neuron is connected to all neurons in the previous layer. Its weights represent the strength of these connections, and the bias is indicated whether this neuron tends to be active or not.

The critical solution is initializing all these weights and biases randomly. This process, for sure, gives bad results at the output layer. Then it is needed to a way of telling the computer that this result is not correct. It could be addressed through a cost function. The basic form of a cost function is the squared error root; it adds the squared differences of random activation neurons and desired values. The value of this function is small when the network recognizes images correctly because the differences will be small.

Cost value has to be calculated for every single training example. Nowadays, they use the train batches approach; in consequence, the average cost value has to be

calculated every single step of training. This function is a substantial multidimensional function consists of multiple variables; weights and biases. The cost function of the previous neural network has ***13002*** variables; This gives a feel of how much is it big!

$$\text{Cost function: } C(w_1, w_2 \dots w_{13002}) \qquad \textbf{(3.11)}$$

Simplification of the complexity of cost function may be addressed by assuming it has just one input and one output, such as ***y(x) = x2 + 10***. The purpose of learning is minimizing cost value. In this case, it is trivial; it could be solved by making a cost function derivative equals to zero. Increasing inputs to two variables like the ***X-Y*** plane makes the cost function appears like a plane graph on the top of that ***X-Y*** plane. Here the complexity becomes higher than one input due to more than two directions to step into according to their slopes. For more complicated functions, it is not easy to find its minimum value as it had done above. But it could somehow get benefits from slopes and to make small steps, stepping to the left if the slop is positive and to the right, if it is negative. With repeating this process a few times, it could reach a local minimum of this function, but there is no grantee that this minimum is the smallest one.

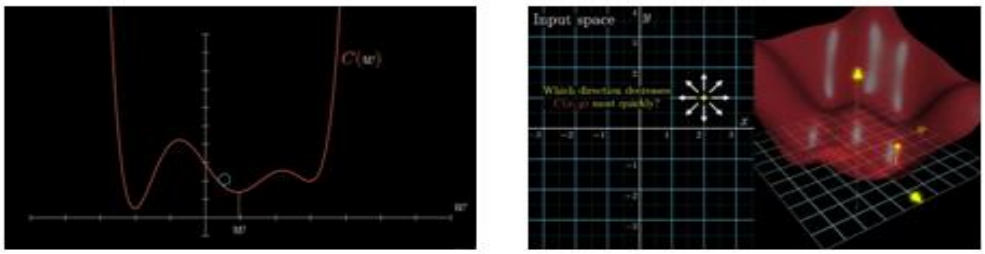


Figure 3.1: One variable vs. two variables cost [40].

In the calculus of multivariable, there is a term called Gradient $\boldsymbol{\nabla C}(\boldsymbol{x}, \boldsymbol{y})$, which gives the direction of the steepest increase. If we take the contrary direction$-\boldsymbol{\nabla C}$, it will lead to the most precipitous decrease, which may be applied to cost function to find its minimum value. Repeating this step by taking a small step in$-\boldsymbol{\nabla C}$ direction, over and over, the minimum of the cost function will be reached.

Backward to cost function, which consists of ***13002***; weights and biases as inputs. Organizing them in one vector $\overrightarrow{\boldsymbol{W}}$ then the negative gradient is a vector that contains ***13002*** values to adjust direction and amount of stepping. comparing the values of $\overrightarrow{\boldsymbol{W}}$ with $-\boldsymbol{\nabla}\overrightarrow{\boldsymbol{W}}$ indicates which weights have critical effects for decreasing cost rapidly

$$\overrightarrow{W} = \begin{bmatrix} 3.12 \\ 4.09 \\ -1.2 \\ 3.01 \\ \vdots \\ -0.5 \\ 1.40 \end{bmatrix} \qquad -\nabla C(\overrightarrow{W}) = \begin{bmatrix} -2.01 \\ -3.23 \\ 0{,}05 \\ -1.7 \\ \vdots \\ 0.20 \\ -0.82 \end{bmatrix} \tag{3.12}$$

Backpropagation is an algorithm to compute the gradient efficiently, which is effectively the heart of how the neural network learns. Taking a deep focus on the effect of $-\boldsymbol{\nabla C}(\overrightarrow{\boldsymbol{W}})$ on weights and biases. $-\boldsymbol{\nabla C}(\overrightarrow{\boldsymbol{W}})$ gives a lot of information about whether some weights have to decrease or increased in different amounts.

To clarify this point, by considering the two-dimensional cost example below, Then compute its gradient at point ***(1,1)***. The result indicates that the ***x*** component ***3*** has more effects on cost function comparing to the ***y*** component.

$$C(x, y) = \frac{3}{2}x^2 + \frac{1}{2}y^2 \tag{3.13}$$

$$\nabla C(1,1) = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \tag{3.14}$$

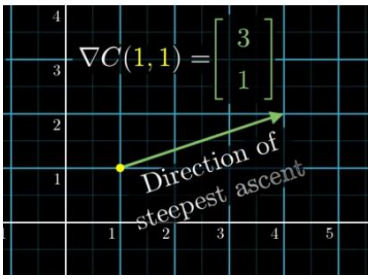


Figure 3.2: $\nabla C(\overrightarrow{W})$ effect.

Until this point, the neural network has been initialized with random weights and biases; after that, they have been modified and adjusted many times based on the gradient descent process. The question that jumps to mind is how it behaves with images that it has not seen before!

It behaves incredibly to identify them, yes, not all of them, but it reaches more than ***80%***. Yes, it is incredible since we do not tell the network which patterns that it looks for. For the previous system, the structure of two hidden layers with ***16*** neurons each, one input layer ***784*** neurons and one output layer with ten neurons. It is expected that the first layer after input recognizes edges, and the second one indicates patterns, but there is no guarantee wither this behavior occurs or not; in most cases, no.

Suppose the input is a random input image. As expected behavior of intelligence, the neural network must give uncertain results such as deactivate all output neurons or activating them all at the same time. But what happens sometimes, in reality, is offering a non-sense answer; It decides for an appropriate response as it is fed into the system. In conclusion, even if the neural network able to recognize images, ideally, it has no idea how they can draw them.

3.3 Back Propagation

As mentioned before, it is an algorithm that could compute the complicated gradient of neural networks. $-\nabla C$ (All weights and biases) could be considered as a direction in ***13002*** dimensions. But it is not easy to think in terms of ***13002*** dimensions; instead, we can think about it in terms of its amplitudes. As each amplitude in $-\nabla C$ gives information about how much it is sensitive to each weight and bias.

Cost function involves costs averaging per each example for tens of thousands of training examples. The way, which weights and biases adjusted for a single gradient descent step, depends on the example itself. Figures below demonstrate this process for one example. It is for recognizing digit ***2***.

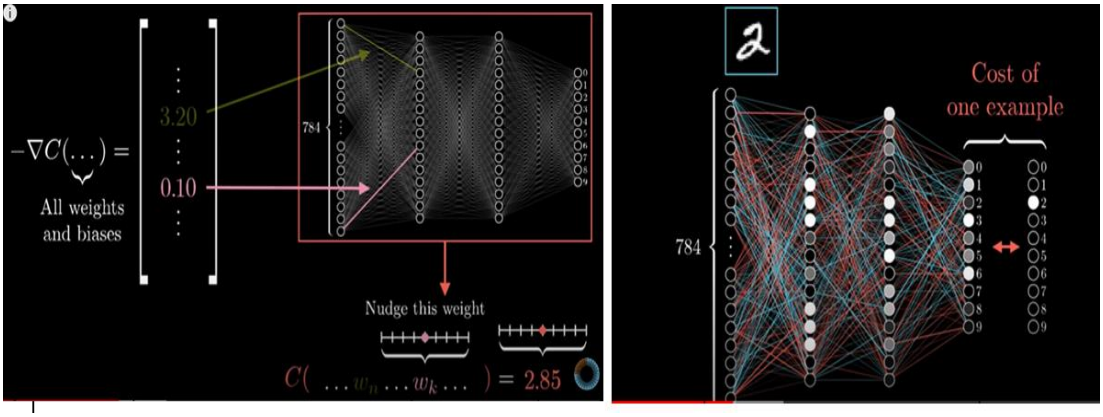


Figure 3.3: Backpropagation, for one example [40].

In the beginning, the network still not trained yet, so the activations in the output layer will be random, and it cannot directly change this activation. Instead, it is helpful to track which adjustment we wish should take to obtain the correct result at the output. In this example, the goal is to classify the image as **2.** It implies raising the activation of neuron ***2*** in the output layer to ***1*** while decreasing others to ***0***. The sizes of nudges should be proportional to how far away each value from the targeted values.

The value of activation for neuron ***2*** equals the weighted sum and biases of all previous layers plugged into the ***ReLU*** function. Three parameters affect the activation of a current neuron, which are weight, previous activation, and bias.

Let us take the weight as a case study; increasing weights joined neuron ***2*** to the previous neurons will help to increase the activation value current neuron. Taking into consideration that weights that are connected from high activated neurons from the previous neuron to the current neuron have much effect on increasing the current activation. For previously talking about **-∇C**, there was no need to talk about nudges. It just cares about which one gives the most effect.

Another way to increase the current neuron activation is by making every previous neuron connected with negative weight got dimmer and vice versa. And doing that backward to uninterested neurons. It is the point where the idea of propagation

backward comes in. By adding all desired effects together, it produces a list of nudges that suppose to happen in the previous layer. The previous explanation is just for the single neuron in the output layer, Neuron 2. Other neurons have the same processes.

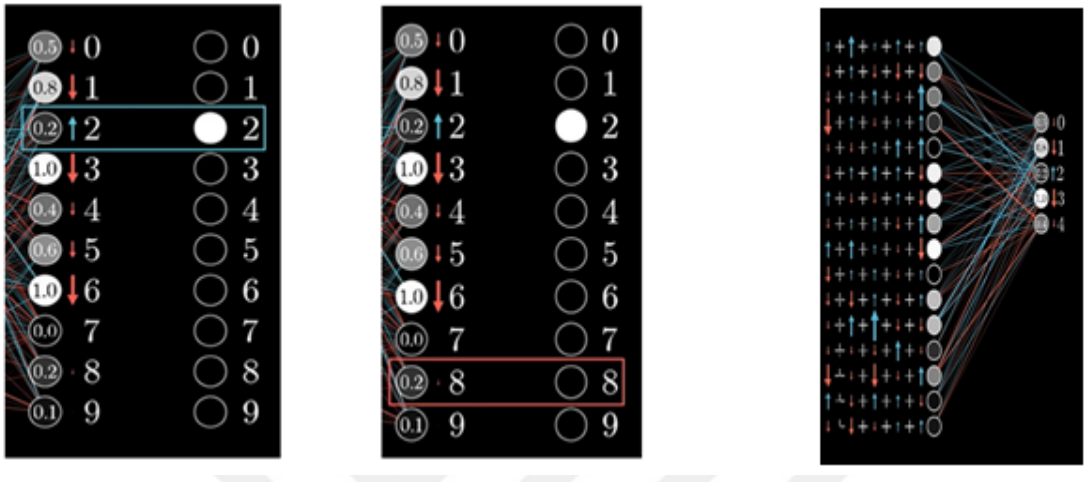


Figure 3.4: Nudges increasing and decreasing –backpropagation [40].

In practice, it takes an extremely long time to add up the influence of each training example every single gradient descent step. Instead, a random shuffle process applied for training data and then divided it into a whole bunch of mini-batches, let us say that each mini-batch has ten training examples. It is then computing a step for each mini-batch instead of doing it for each image. This will not give the actual gradient of a cost function, which depends on all training data, not on the subset so, it does not produce the most steps downhill, but it is a good approximation for that.

The previous idea is not ***100%*** accurate, but it gives us a significant computational speed up. So if we plot the trajectory of our neural network under the relevant cost surface, it will be like a drunk man stumbling downhill with taking quick steps. Rather than a carefully calculating man determining the exact downhill direction of each step before taking a prolonged and careful step in that direction, this gives the fast speed advantage to this technique. It is called stochastic gradient descent (SGD).

In summary, backpropagation is a process to determine how a single training example behaves to nudge the weights and biases. It is not just in terms of whether they should go up or down, but also, what relative proportions to those changes cause the most rapid decrease to the cost function. The correct gradient descent step will involve doing this process for all tens of thousands of training examples and averaging the desired changes that resulted, but this process is slow. Instead, a random subdivide process for data into mini-batches and compute each step concerning mini-batch. Then repeatedly calculating all steps for these mini-batches and tracking their adjustments until reaching a local minimum for the cost function.

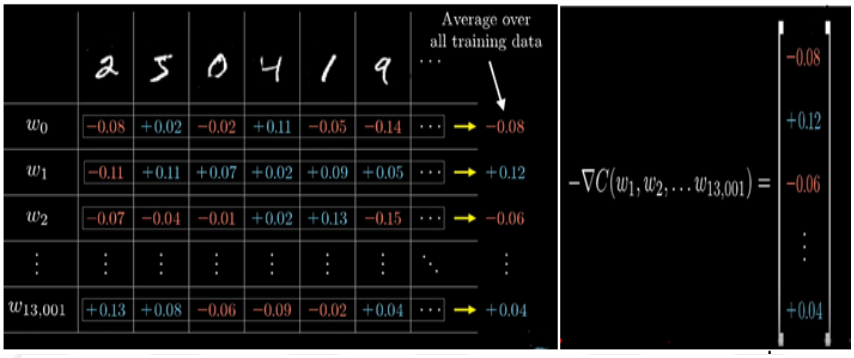


Figure 3.5: Backpropagation with SGD technique [40].

3.4 Back Propagation Calculus

It is preferred to simplify the neural network into a simple one so that each layer has just a single neuron, to make calculus easy. Three weights and three biases would determine the simplified network. The goal is to understand how sensitive the cost function is to these variables, then knowing which adjustment to these variables is going to cause the most decrement in the cost function. Here the focusing is on the last two neurons, which are the output neuron and the neuron before it.

Let the label of activation for the last neuron be $\boldsymbol{a^{(L)}}$ then the activation of previous layer neuron is $\boldsymbol{a^{(L-1)}}$, and let the desired value for a given training example is y then the cost function is

$$C_0(..) = (a^{(L)} - y)^2 \quad \textbf{(3.15)}$$

As mentioned earlier, activation of the next layer depends on the activation of the previous layer multiplied by weight connection and additive bias, then apply ***ReLU*** function.

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \quad \textbf{(3.16)}$$

$$a^{(L)} = \sigma(w^{(L)} a^{(L-1)} + b^{(L)}) \quad \textbf{(3.17)}$$

$$a^{(L)} = \sigma(z^{(L)}) \quad \textbf{(3.18)}$$

In the end, all variables w,z, a and c number, and the goal is understanding how sensitive the cost function is to a small change in weight $\boldsymbol{w^{(L)}}$, in mathematics $\frac{\partial C_0}{\partial w^{(L)}}$. It could be translated to nudge actions, so any tiny nudge in $\boldsymbol{w^{(L)}}$causes some nudge to $\boldsymbol{z^{(L)}}$, consequently causes change to $\boldsymbol{a^{(L)}}$,which influences the cost directly $\boldsymbol{C_0}$.

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \quad \textbf{(3.19)}$$

This a chain rule; multiplication of these three ratios equals the sensitivity of $\boldsymbol{C_0}$ to a small change in $\boldsymbol{w^{(L)}}$.

$$\frac{\partial C_0}{\partial a^{(L)}} = \frac{\partial (a^{(L)} - y)^2}{\partial a^{(L)}} = 2(a^{(L)} - y) \quad \textbf{(3.20)}$$

Nudge size is proportional to the difference between output value and the desired value. Note that if the output's value is so small, there will a significant impact on the cost function.

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \frac{\partial \sigma(z^{(L)})}{\partial z^{(L)}} = \sigma'(z^{(L)}) \quad \textbf{(3.21)}$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = \frac{\partial (w^{(L)} a^{(L-1)} + b^{(L)})}{\partial w^{(L)}} = a^{(L-1)} \tag{3.22}$$

Which means that any amount of nudge weight changing that influences the last layer, depends on how much previous neuron strong is. All of these effects are just for the cost derivative concerning $\boldsymbol{w^{(L)}}$ per training example. The cost function involves averaging together all those costs across many training examples. It is just for a single component of the gradient vector.

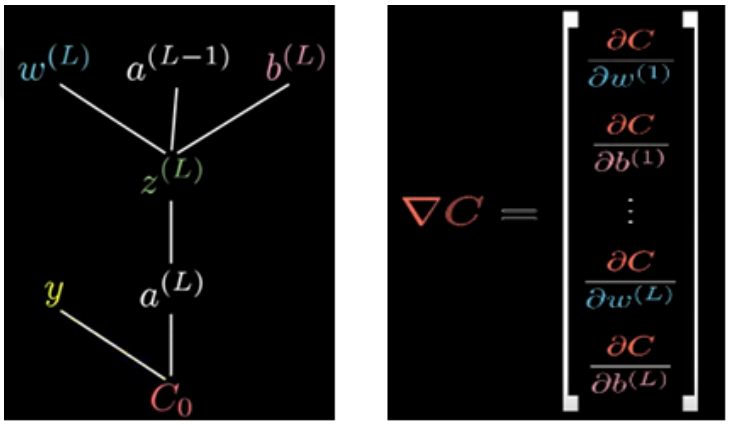


Figure 3.6: Parameters of gradient vector[40].

For sensitivity to the bias,

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{3.23}$$

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = \frac{\partial (w^{(L)} a^{(L-1)} + b^{(L)})}{\partial b^{(L)}} = 1 \tag{3.24}$$

$$\frac{\partial C_0}{\partial b^{(L)}} = \sigma'(z^{(L)})\, 2(a^{(L)} - y) \tag{3.25}$$

For sensitivity to the activation of the previous layer,

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = w^{(L)} \sigma'(z^{(L)})\, 2(a^{(L)} - y) \tag{3.26}$$

There is no ability to influence the activation directly. It's helpful to keep track of using chain relation backward. For networks that have more than one neuron, there is no significant difference differs from increasing indexes under those symbols. For example, instead of just $\boldsymbol{a^{(L-1)}}$ & $\boldsymbol{a^{(L)}}$, there will be $\boldsymbol{a_0}^{(L-1)}, \boldsymbol{a_1}^{(L-1)}$…etc & $\boldsymbol{a_0}^{(L)}, \boldsymbol{a_1}^{(L)}$,etc. It may be easy to index the layer ***(L-1)*** by ***k*** and ***J*** to the index layer ***L***.Summing up squares of differences between the last layer activations and the desired output.

$$C_0 = \sum_{j=0}^{n_L - 1} (\, a_j{}^{(L)} - y_j)^2 \tag{3.27}$$

Let a weight that connects the $\boldsymbol{K^{th}}$ neuron to the $\boldsymbol{J^{th}}$ neuron be $\boldsymbol{w_{jk}}^{(L)}$. This indexing might feel a little backward at first, but it lines up with how it would index the weight matrix. It is good to give a name to a relevant weighted sum like ***z***, so that the activation of the last layer is just a particular function, like sigmoid applied to ***z***.

$$z_j{}^{(L)} = w_{j0}{}^{(L)} a_0{}^{(L-1)} + w_{j1}{}^{(L)} a_1{}^{(L-1)} w_{j2}{}^{(L)} a_2{}^{(L-1)} b_j{}^{(L)} \tag{3.28}$$

$$a_j{}^{(L)} = \sigma(z_j{}^{(L)}) \tag{3.29}$$

At the final, the chain rule is the same as seen before except it has a complicated indexes

$$\frac{\partial C_0}{\partial w_{jk}{}^{(L)}} = \frac{\partial z_j{}^{(L)}}{\partial w_{jk}{}^{(L)}} \frac{\partial a_j{}^{(L)}}{\partial z_j{}^{(L)}} \frac{\partial C_0}{\partial a_j{}^{(L)}} \quad \textbf{(3.30)}$$

But this chain rule changed concerning activation in the previous layer because one activation can feed more than one activations in the second layer, so the chain becomes

$$\frac{\partial C_0}{\partial a_k{}^{(L-1)}} = \sum_{j=0}^{n_L-1} \left(\frac{\partial z_j{}^{(L)}}{\partial a_k{}^{(L-1)}} \frac{\partial a_j{}^{(L)}}{\partial z_j{}^{(L)}} \frac{\partial C_0}{\partial a_j{}^{(L)}}\right) \quad \textbf{(3.31)}$$

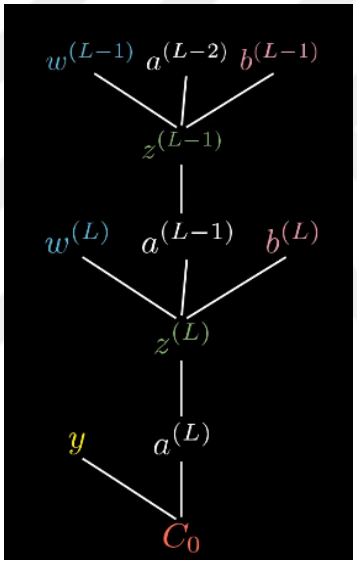


Figure 3.7: Chain rule[40].

4. DEEP LEARNING

The neural network gives a fantastic performance, but it is not understandable how it could do that. It initialized randomly, trained by many samples, then could recognize images that it has not seen before! There is no idea about weights and biases values since these values discovered automatically.

The main goal of the neural network is to discover the mind algorithm. Thus why inventors who were working on neural networks developed artificial intelligence to know how the mind works; unfortunately, that adds another challenge, which is how does neural network work also.

The human face detection task could be addressed without using a learning algorithm or neural network directly. The problem could be divided into sub-problems like: does the image have an eye in the top left? Or in the top right? Mouth down the image? Noise in the middle? So that if the answer for most of these subproblems is yes, then this image is a human face image probably.

Since neural networks can address these subproblems, then it is better to assign a neural network to them. The subproblems are not as simple as they supposed to be. They have to be decomposed into more straightforward and simpler subproblems. For example, for an eye on the top left, is there an iris, eyebrow.. etc. until they reach the pixel scale. Doing these for many series of layers means “Deep neural Network”[7]

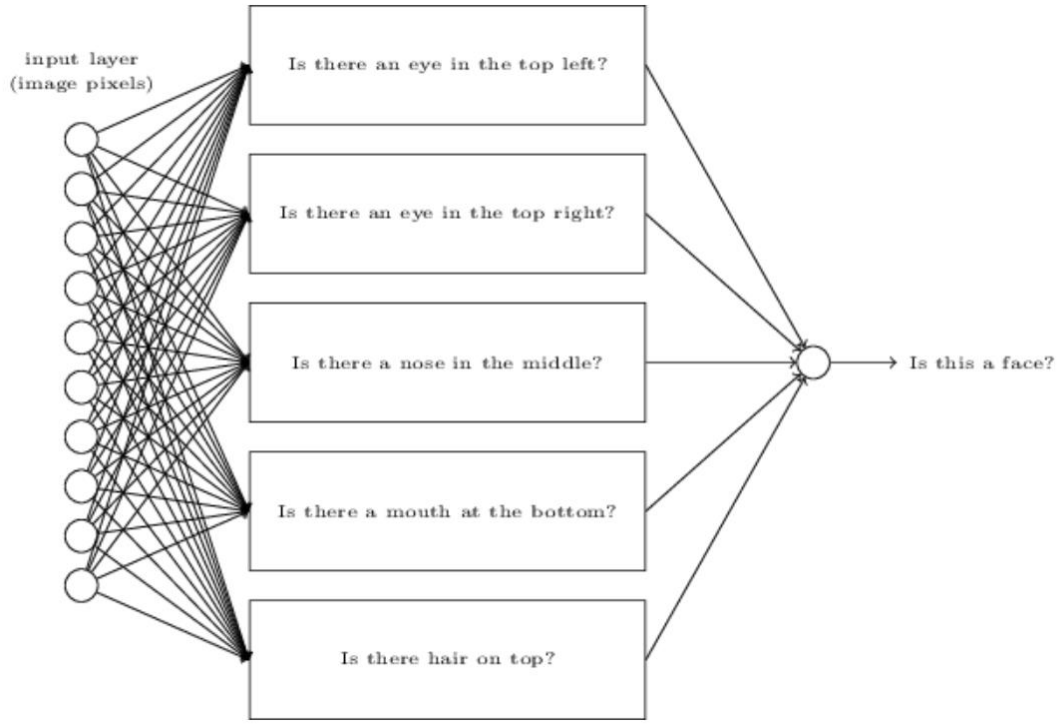


Figure 4.1: Deep learning [42].

4.1 Convolutional Neural Network (CNN)

It is a neural network that works with two-dimensional image data. The convolution process that takes place inside is the main reason for its naming. The convolution is a linear operation that implies multiplication of weights with input. These weights called a filter or kernel as usual.

The filters always smaller than input in size, and the multiplication happened in the dot product. After multiplication, a summation process takes place to produce a single value. This filter goes through the whole image as a sliding window. It can discover the feature anywhere in an image.

The workflow of the convolution process used in CNN could be simplified by illustrate edge detector using 3 x 3 filter with weights chosen specifically to detect vertical edges for a grayscale image from light to dark.

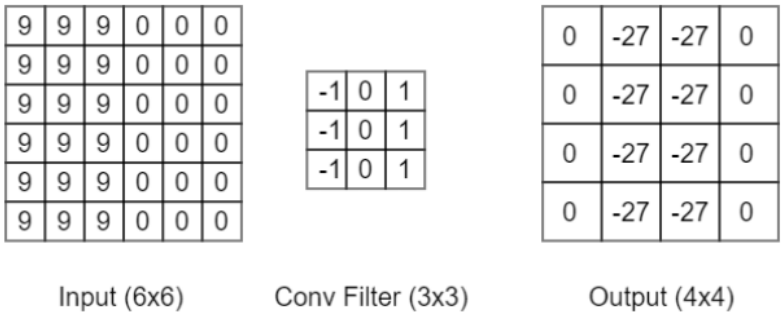


Figure 4.2: Vertical edge detector [41].

4.1.1 Padding process

The convolution process results in a smaller width and height than the original input. Feeding the convolutional layer's output to another one consequently causes size shrinking. Then the neural networks will have a limitation in the number of their layers if there is no treatment for this shrinkage. A popular treatment is to pad the entire image with enough zeroes such that the output shape will have the same width and height as the input, called the SAME padding. Leaving convolution without a treat its shrinking effect also called valid padding.

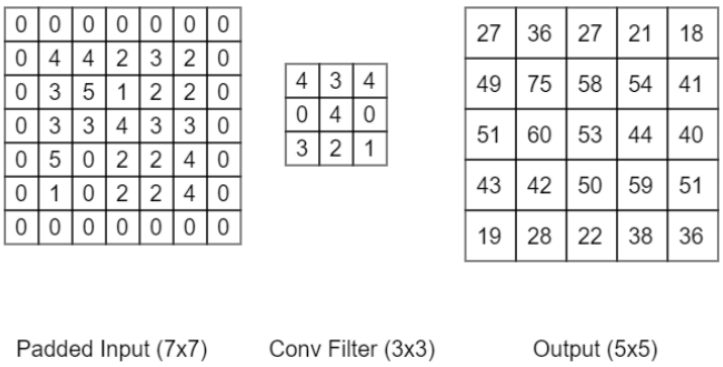


Figure 4.3: Padding effect [41].

4.1.2 Stride size

Stride refers to the number of pixels jumped by the filter in the width and height dimension every time for scanning to compute the dot-product.

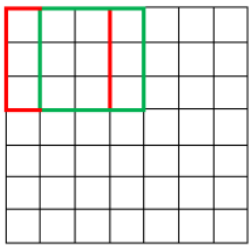


Figure 4.4: Stride size =1[41].

4.1.3 Effect of the input depth

It leads to extending the filter's depth to match the depth dimension of the input. For the following figure, the input depth is two; then the filter has much the same depth. The point here is the increased number of values to be added up for each point in output.

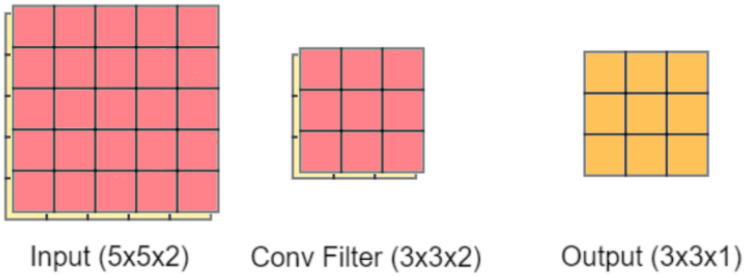


Figure 4.5: Input and filter depth [41].

4.1.4 Output depth controlling

Output depth may be controlled by stacking up multiple convolutional filters. While computing its results, each filter acts independently. All results are stacked together to create the output. As a result, adding or removing convolutional filters leads to control of the output depth.

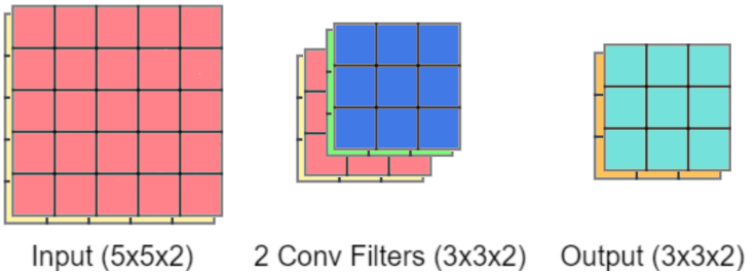


Figure 4.6: Output layers stacking [41].

4.1.5 Max pooling layer

It is a layer that resulted from sliding a window along with the input and simply take the max value at each point. The following figure illustrates this process of an input size 4 x 4, kernel "filter" size of 2 x 2 and stride size 2.This layer introduced to downsample the massive computations resulted from convolutions, especially when processes go deeper and deeper.

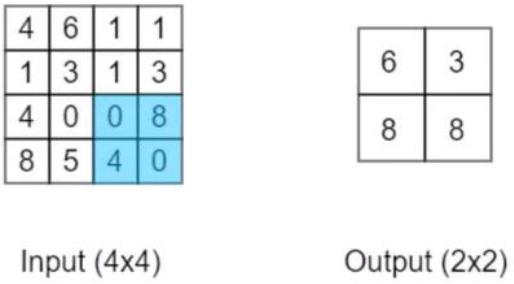


Figure 4.7: Maxpool layer [41].

The Generalized dimensions of output layer for each input with $\boldsymbol{n}$ x $\boldsymbol{n}$ x $\boldsymbol{d_n}$, Filter with $\boldsymbol{f}$ x $\boldsymbol{f}$ x $\boldsymbol{d_f}$, padding $\boldsymbol{p}$ and stride $\boldsymbol{s}$ is

$$Output\ dim = \frac{n\ + 2p - f}{s} + 1\ \times \frac{n\ + 2p - f}{s} + 1\ \times \#\ filters \qquad \textbf{(4.1)}$$

Where $\boldsymbol{d_n}$ and $\boldsymbol{d_f}$ are the input and filter depths.

Generally, in the convolution neural network (ConvNet), there are three types of layers which are convolution layers, pooling layers, and fully connected layers. It is common to consider the convolution layer with the pooling layer as a single layer.

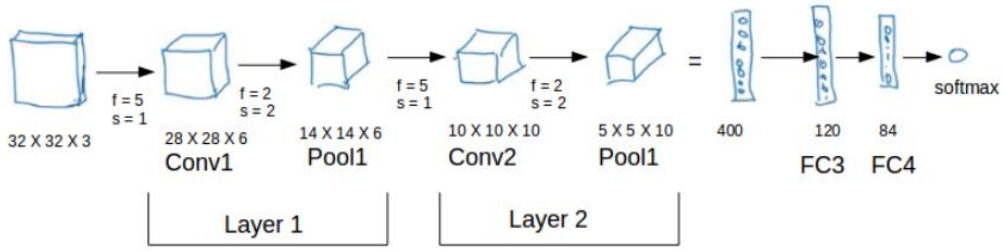


Figure 4.8: Convnet structure with dimentions [41].

There are two significant advantages for convolutional layers over the fully connected layers, which are the parameter sharing and sparsity of connections.for example number of parameters of fully connected layers are huge; dimensions' multiplication. For ConvNet, it deals with filter dimensions and depth.

$$\#\ FC\ parameters\ = (n \times n \times n_d)_{in} \times (n \times n \times n_d)_{out} \quad \textbf{(4.2)}$$

$$\#\ ConvNet\ parameters\ = (f \times f + 1) \times f_d \quad \textbf{(4.3)}$$

4.2 Classic Network Examples

LeNet-5: Its inputs are grayscale images. Once they passed through a combination of convolution and pooling layers, the outputs will be passed through fully connected layers and classified into corresponding classes[20].

Table 4.1: Lenet summary table.

	LeNet-5
Parameters	60k
Layers flow	Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
Activation functions	Sigmoid/tanh and ReLU

AlexNet: Its inputs are images. It had the same structure as LeNet-5 but with more convolution and pooling layers[21].

Table 4.2: Alexnet summary table.

AlexNet	
Parameters	60 million
Activation functions	ReLU

VGG-16: This network uses a more straightforward network structure. The convolution layers have the SAME padding with ***3 x 3*** filters. The max pool layers have ***2 x 2*** filter and stride size = ***2;*** this makes a half size reducing each time. The total parameters are ***138*** milion[22].

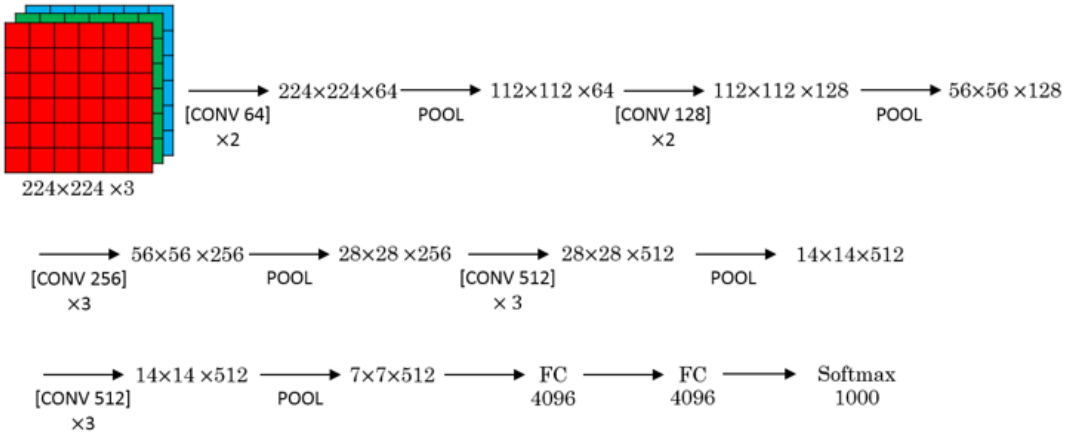


Figure 4.9: VGG-16 structure.

ResNet: It solves vanishing and exploding gradients issues while training intense neural networks. It uses residual blocks which skip connections by two layers as following [23]

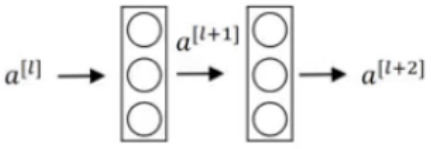


Figure 4.10: Residual blocks.

$$z^{[l+1]} = w^{[l+1]}a^{l} + b^{[l+1]} \quad \textbf{(4.4)}$$

$$a^{[l+1]} = \delta(z^{[l+1]}) \quad \textbf{(4.5)}$$

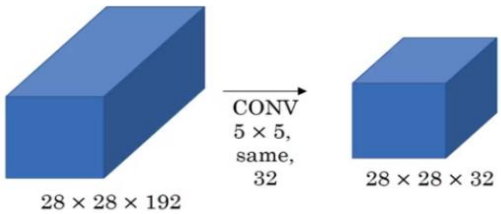
$$z^{[l+2]} = w^{[l+2]}a^{[l+1]} + b^{[l+2]} \quad \textbf{(4.6)}$$

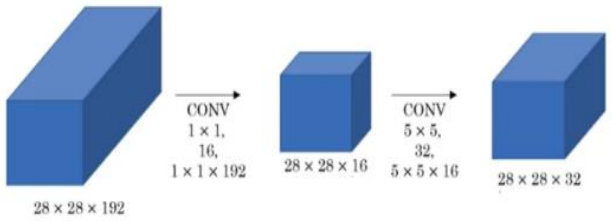
$$a^{[l+2]} = \delta(z^{[l+2]}) \quad \textbf{(4.7)}$$

$$a^{[l+2]} = \delta\left(z^{[l+2]} + a^{l}\right) \quad residual \quad \textbf{(4.8)}$$

Figure 4.11: Resnnet structure.

Inception: many computations "dot multiplications" when using a large size filter is big. Introducing a ***1 x 1*** filter at the beginning is helpful to reduce input depth then reducing the computations cost. As illustrated below

# computations -dot multiplications	
	28 x 28 x 32 x 5 x 5 x 192 = 120 million



28 x 28 x 16 x 1 x 1 x 192 + 28 x 28 x 32 x 5 x 5 x 16= 12.4 million

Figure 4.12: Number of dot multiplications for different filters.

Instead of choosing the filter size to use, or whether to use the convolutional layer or pooling layer, inception uses all of them and stacks all the outputs[24].

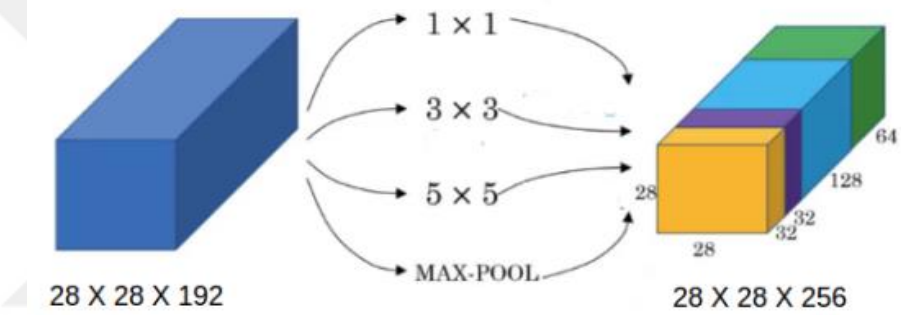


Figure 4.13: Inception network structure.

4.3 Object Detection Algorithm

Region-based Convolutional Neural Networks(R-CNN): It uses an object proposal algorithm, called Selective Search, which reduces the number of bounding boxes that are fed for the classifier; approximately ***2000*** region proposals. The selective search uses local cues like texture, intensity, and color to generate all the possible locations of an object. These boxes could be introduced to the CNN based classifier. Keeping in mind that, fully connected part of CNN takes a fixed-sized input so, all generated boxes should have a fixed size (***224***×***224*** for VGG) and feed to the CNN part[2].

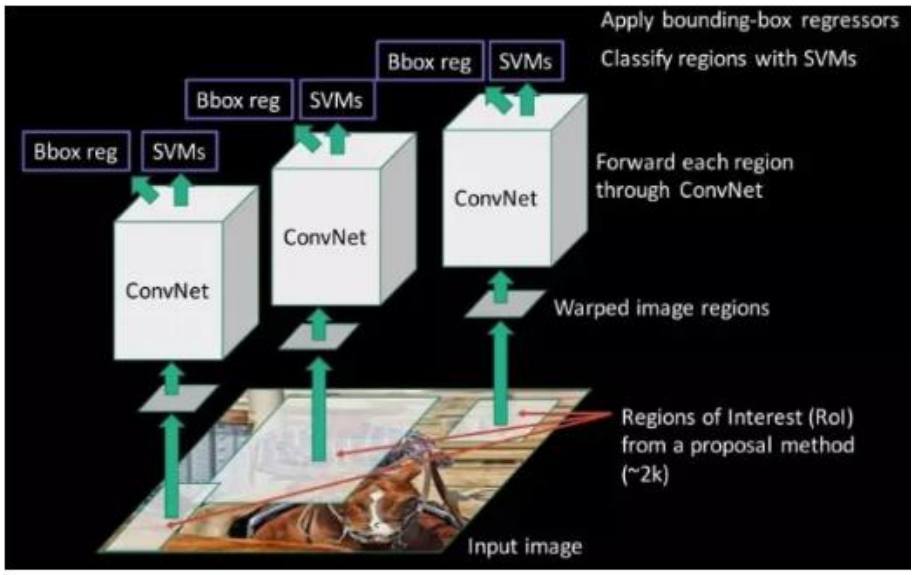


Figure 4.14: RCNN object detection algorithm [43].

Fast RCNN: It inputs images into ConvNet directly, which in turn generates RoI. As mentioned before, fully connected layers needed fixed-size inputs. The pooling layer is applied on RoI (RoI Pooling layer) to make this RoI ready to pass through a fully connected network then softmax layer, which determines classes and regression layer, which determines boxes[3].

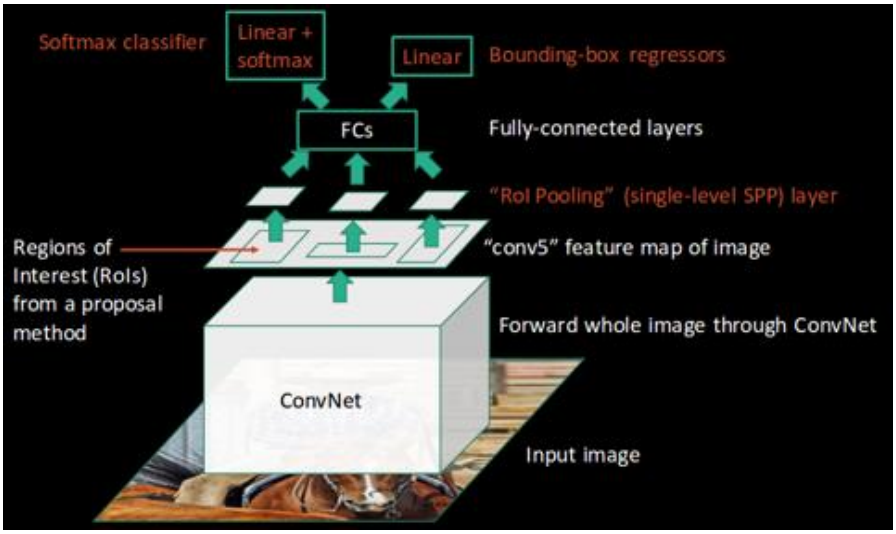


Figure 4.15: Fast RCNN object detection algorithm [43].

Faster RCNN: It replaces selective search with a tiny convolutional network called Region Proposal Network (RPN)to generate regions of Interest(RoI). It introduces the idea of anchor boxes with different scales (***128x 128, 256×256*** and ***512×512***.) and different aspect ratio (***1:1, 2:1*** and ***1:2***) to handle the variations in aspect ratio and scale of objects[4].

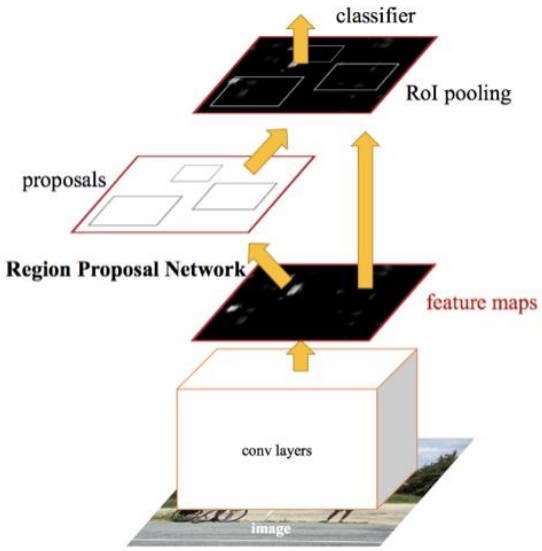


Figure 4.16: Faster RCNN object detection algorithm [43].

Mask RCNN: Extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI) and replaces RolPool by RoIAlign. RoIAlign does not adjust the input proposal to fit the feature map correctly. It merely takes the object proposal and divides it into equal four bins: Top left, Top right. Bottom left and Bottom right using the bilinear interpolation, then It applies the average or maximum function to get the value[5].

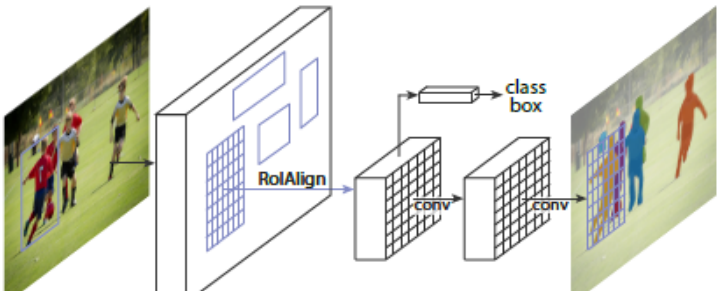


Figure 4.17: Mask RCNN object detection algorithm [5].

5. DETECTION PRACTICAL WORK AND EXPERIMENTS

Experiments of airplanes detection using drones images are explained in this chapter at the beginning describing the dataset collection and related works, including enhancing processes. They are then introducing training models configurations. Finally concluded with visible and evaluation results.

5.1 Dataset

A dataset could be defined as a group or collection of useful data that relates to a specific field. Or basically, it is a group of correlated sets of information that is consists of individual elements and could be dealing with them as a single unit in the computer world. The simple example is tabular data, which its information related to one or several database tables. In these tables, every single column corresponds to a single variable which maybe indicates any kind of properties such as height, width, or thickness of an object. Each information of this called datum, and we can generalize this concept to make data sets consist of a collection of files, images, and documents.

In general, drones should avoid flying around or near airport areas due to air traffic. Aircraft have difficulties avoiding a drone while it is flying; thus, why drone operators have been assigned responsibility for any safety hazard their drone creates in the airport environments[25]. Fortunately, there is an open-access dataset for airplanes dataset for airport view with high spatial resolution 3.14cm/px. This dataset has been taken for Le Bourget airport in Paris. It collected by two eBee Classic drones flying simultaneously[26].

Table 5.1: Airplane dataset collected by drones [26].

Technical data	
Ground resolution	3.14 cm (1.23 in)/px
Coverage	0.66 sq. km (0.25 sq. mi)
Flight height	120 m (393.7 ft)
Number of images	557

Dataset has been categorized into two categorizations; high-resolution images and low-resolution images. The high-resolution images contain more information to be extracted, but it causes slow training. The low-resolution category is divided into two groups; the first one is the general one, which contains all images; the other one is containing just airplane objects. There is a last external dataset of satellite images used here for evaluation reliability of algorithms used in this thesis. Two hundred images have been taken randomly from airplane categorization of the NWPU-RESISC45 dataset, which is a collection of public data available for Remote Sensing Image Scene Classification (RESISC)[39].

Table 5.2: Sorting dataset according to interesting objects.

Dataset Categorizations	Resolution (px)	# Images
General drone dataset-high resolution	922 x 864	557
General drone dataset-low resolution	369 x 259	557
Airplane Objects drone dataset-low resolution	369 x 259	94
Satellite Dataset	256 x 256	200

There is another step done in this thesis; to enhance the performance of the deep learning model. It is an augmentation step for the dataset that is already existed. Usually, deep learning frameworks have their own built-in data augmentation steps, but some times they are not sufficient or lacking the required functionality. Figure 5.1 shows images contain different kinds of airplane objects. These objects have different sizes, shapes, and colors, which gives an additional challenge to this

research. Besides, it is clearly illustrated that the existence of an airport's environment with different buildings, vehicles, and backgrounds will give a key feature for training steps to identify airplanes. It will detect them with high efficiency in the real-world environment.

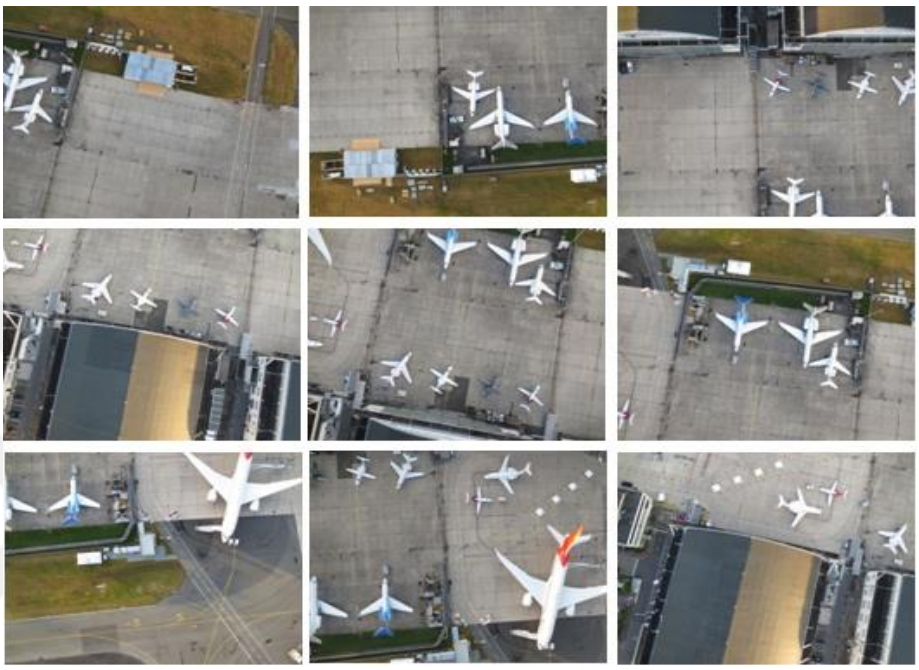


Figure 5.1: Dataset samples.

Each image in the original dataset has one augmented image inside high-resolution and general low-resolution categorizations. For the airplane-object low-resolution dataset, there are six augmented images. Different augmentation methods used for augmentation steps such as rotational, cropping. Hue means changing in color. Elastic trans means a water-like effect. The coarse drop means to set some image areas to zero and Gaussian noise, but some of these effects made some images not proper to be considered as good images for training, so a final manual sorting had been done to choose the good ones. Figure 5.2 shows below the effects of each image augmentation process.

Figure 5.2: Image augmentation.

Table 5.3: Images number per dataset.

Dataset Categorizations	Resolution (px)	# Images
General drone dataset-high resolution	922 x 864	1114
General drone dataset-low resolution	369 x 259	1114
Airplane Objects drone dataset-low resolution	369 x 259	689
Satellite Dataset	256 x 256	200

5.2 Dataset Annotations

The annotation process aims to mark objects and to outline them on images. It offers various keywords for classification purposes, thus could be read by the computer. It is an essential task that helps computer vision models working in a real-world scenario. The image annotation could be implemented using different annotation tools such as 2D bounding boxes, semantic segmentation, or cubic annotation. The image annotation is mainly done manually using any kind of available software. In this work, I am using 'labelImg' software to generate bounding boxes around each plane object.

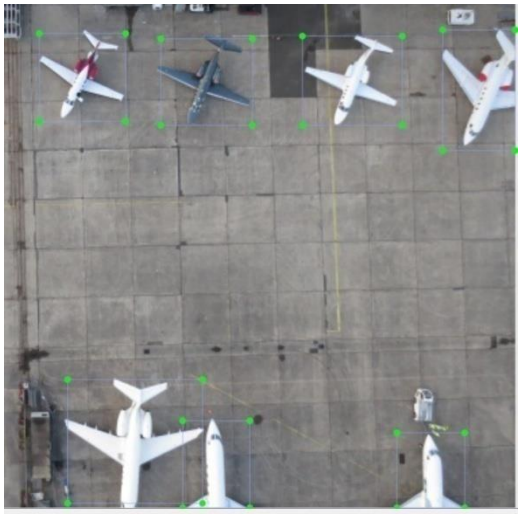


Figure 5.3: Bounding boxes, the annotation.

Figures 5.4-5.6 below is showing the bounding box annotations over both train and test datasets. For low resolution- airplane object dataset, the training dataset has a total of 1401 examples with areas that vary between small, medium, and large. The validation dataset has 211 examples with different areas small, medium, and large. For both high resolution and global datasets, the training dataset has a total of 623 examples, and the validation dataset has 215 examples with different areas.

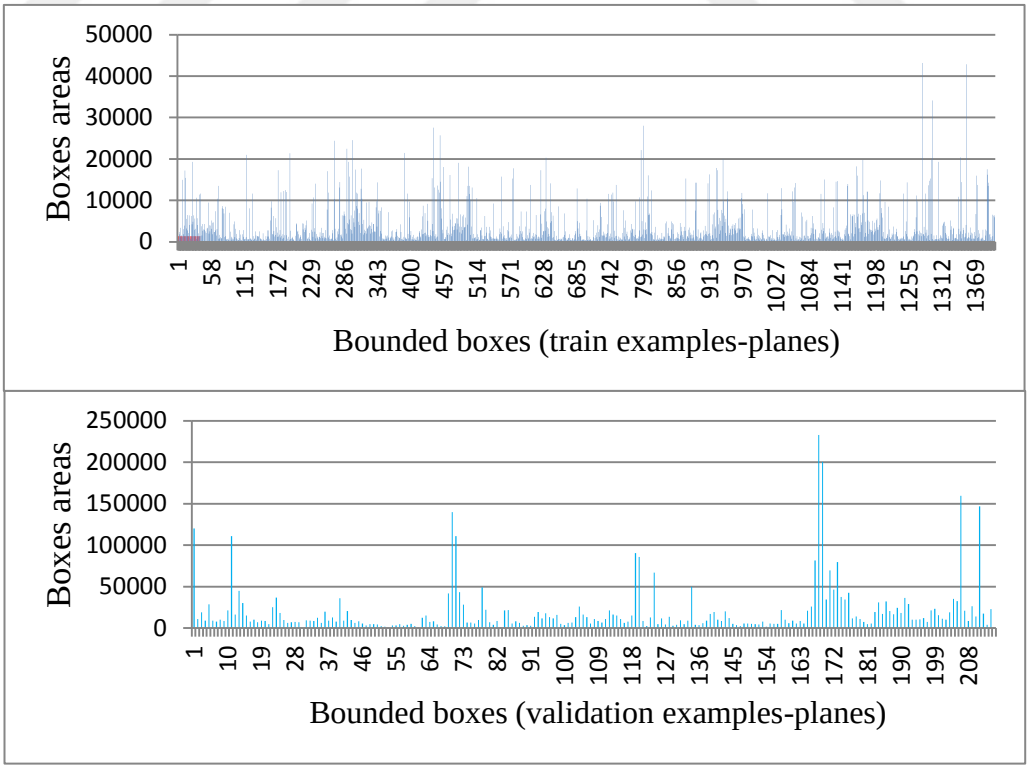


Figure 5.4: Bounded boxes statistics, low-resolution airplane object.

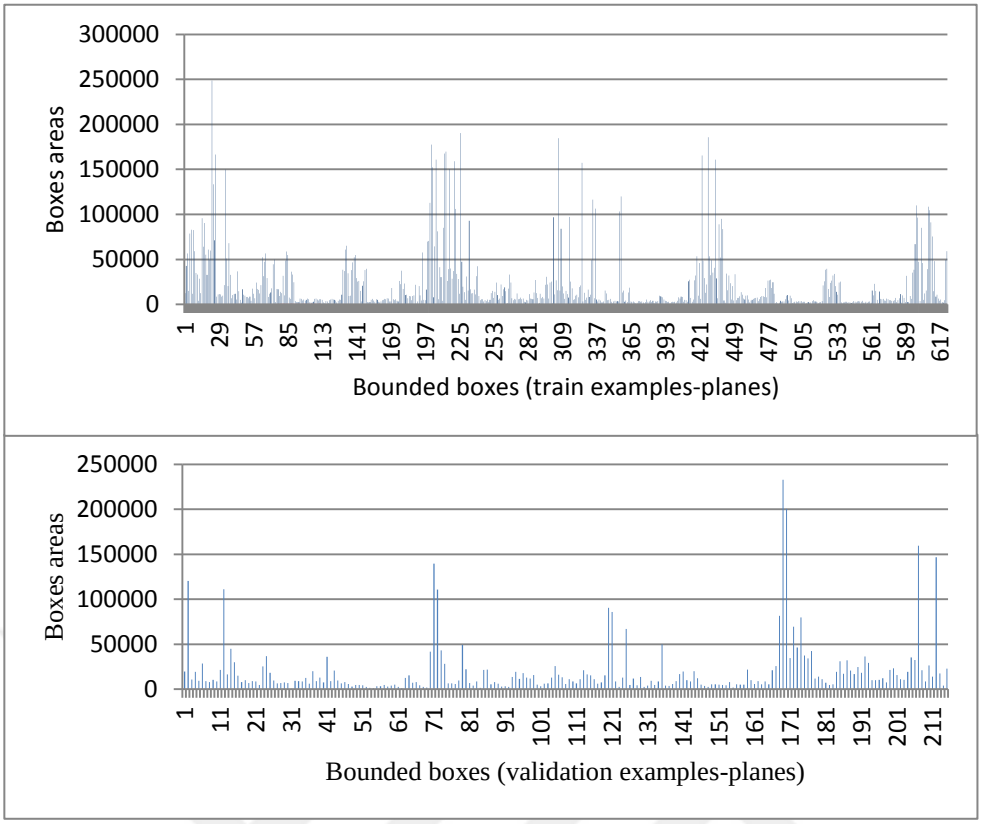


Figure 5.5: Bounded boxes statistics, high-resolution.

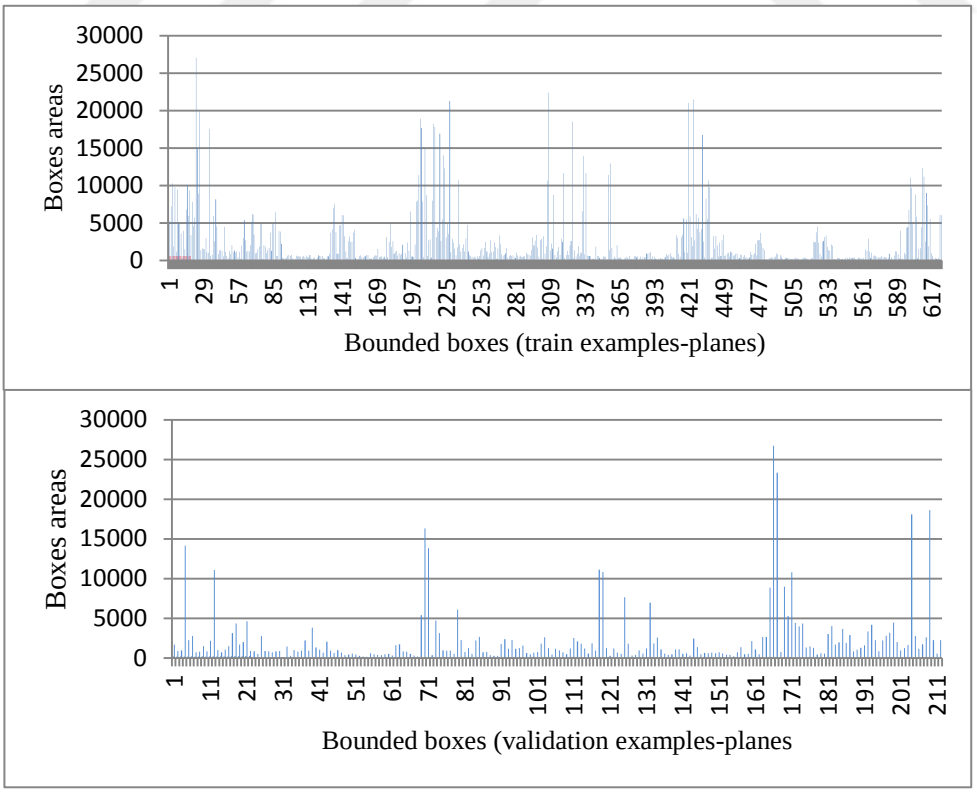


Figure 5.6: Bounded boxes statistics, general-dataset.

For other kinds of annotations, I used the Pixel Annotation Tool, which is a semantic segmentation software that annotates every single pixel within an image. This kind of annotation is mandatory for mask object detection.

Figure 5.7: Mask annotations.

5.3 Training

The whole experimental work, related to this thesis, has been done by a computing machine, a laptop, which has the following specifications**.**

Table 5.4: Training machine.

Technical Data	
Processor	Intel(R) Core(TM) i7-7500 U CPU
Speed	2.70 GHz – 2.9 GHz
RAM	8.00 GB
Graphics Data	
Chip Type	GeForce 940MX
Memory	6030 MB

Together with the machine's specification, there is some software used to allow the machine to performs its best efficiency work to do training processes. For example,

CUDA, which is a platform created by Nvidia to allows software developers and software engineers to use a CUDA-enabled graphics processing unit for general purpose processing. cdDNN, which is a GPU-accelerated library for deep neural networks training purposes.Jupyter which is open-source software and services for interactive computing across multi-programming languages. Google Colab is an online free cloud service that offers running a deep learning or machine learning models. Anaconda is an open-source for Python and R programming languages. Its goal is to simplifying package management, and deployment.python is high-level for general-purpose programming language. Atom: is an open-source code editor for macOS, Linux, and Microsoft Windows operating systems. TensorFlow is a well-known research platform for deep learning and can modify and implement different types of deep learning models.

5.3.1 Training Models

The ultimate goal of this thesis is not just airplane detection but also estimate its surface area; thus, models here specified, for Instance, segmentation purpose. According to recent publications related to Instance segmentation purpose, the Mask R-CNN model gave perfect results compared to other results such as MNC [28], FCIS[29]+OHEM and FCIS+++ + OHEM. Also, It has a reliable performance for overlapping objects, and since it has approximately the same speed as faster RCNN, which used as a detection model by other's work related to airplane detection (0.2s), models used here are masks RCNN models[5].

Table 5.5: Mask R-CNN instance segmentation results[5].

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS+OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++,OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

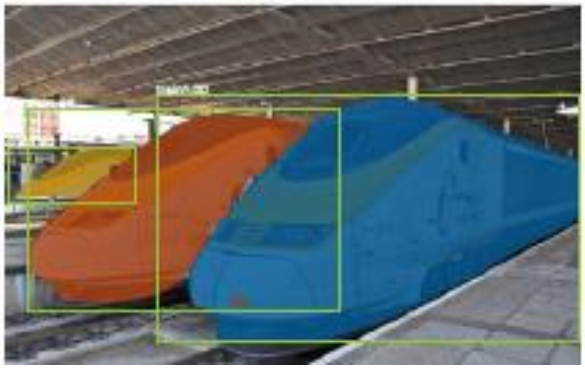


Figure 5.8: Overlapped objects.

mask_rcnn_inception_v2_coco configurations had been used from TensorFlow models. It is the base configuration for the training process which used in this research. The configurations of hyperparameters have no optimal values, and their values change as datasets change [7]. Thus different changes have been made on base configurations. The changes have been categorized into seven additional models, as shown in table 5.5.

Table 5.6: Models.

Model 2	
Dataset	Low resolution with airplane object
Image resize	(230 x 350) px
Model 3	
Dataset	Low resolution with airplane object
first_stage_features_stride	8
height_stride	8
width_stride	8
Model 4	
Dataset	Low resolution with airplane object
Learning rate	Manual_Step_Learning_Rate = 0.002 Schedule= 0.0002 , 0.00002
Model 5	
Dataset	Low resolution with airplane object

first_stage_max_proposals	100
Model 6	
Dataset	Low resolution with airplane object
Learning rate	Manual_Step_Learning_Rate = 0.00002 Schedule= 0.000002 , 0.0000002
Model 7	
Dataset	High resolution
Model 8	
Dataset	Low resolution

5.3.2 CPU vs. GPU

GPU considered now as the heart of deep learning . in terms of hardware, it is a single-chip processor that is used for extensive graphical and mathematical computations. And this makes CPU free to do other functions assigned to it. GPU has a large number of simple cores that allow parallel computing, but for CPU, it has complicated cores run processes sequentially. For machines that have GPU, the host code will be run on CPU, whereas the CUDA code will be run on GPU. In addition to that, the GPUs are bandwidth optimized, but CPUs are memory access time optimized.

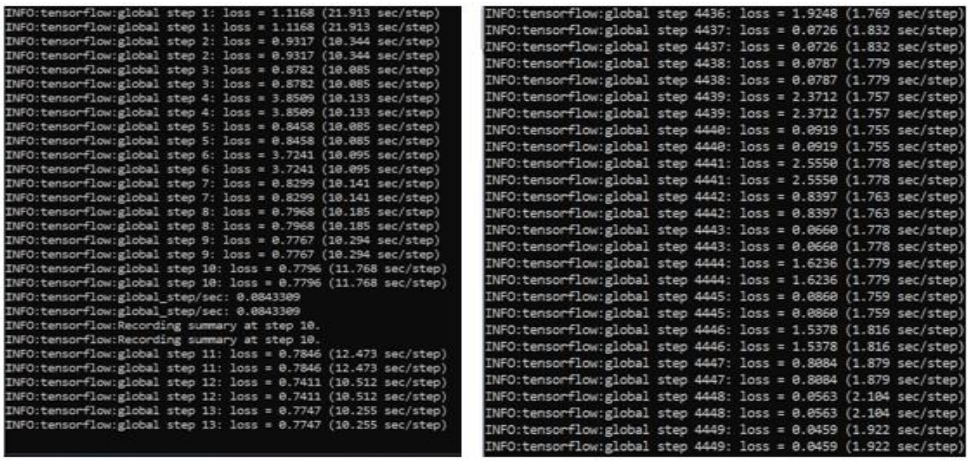


Figure 5.9: GPU vs. CPU training.

5.3.4 Hyperparameters selections

The configurations of hyperparameters have no optimal values, and their values change as datasets change. Different modifications of the hyperparameters had been done with the scope of hardware capabilities of the computational environment. These different iterations had been done until they reached satisfying accuracies based on coco metrics.

a) Learning Rate

It is so essential to select an optimal value for the learning rate within the model configurations. This hyperparameter considered the most important one that needs to be configured in the model. The learning rate interacts with other aspects of optimization operation. Smaller values assigned to the learning rate need more training steps. They give higher efficiency of extracting the information; Thus, in the beginning, the initial learning rate was 0.0002 for model 1, then it has been decreased by ten times in model 6; so the initial learning rate became 0.00002. Additional decreasing has been made for additional models, but model 6 was the best among them, so they have been removed. To see the effect of increasing the value of learning rate, which in general needs fewer training steps, and increasing by ten times has been done for model 4, so the initial learning rate became 0.002. Also, increments of value by 2,5, 15, and 20 times have been made, but they were not as good as model 4 in their evolutions' metrics.

b) Type of optimizer

Optimizers are algorithms used to adjust weights and activation values in the neural network to decrease the losses. Gradient descent (GD) is the most popular one used in deep learning fields, and we can find implementations of GD in most state-of-the-art libraries such as lasagne, Caffe, and Keras. Stochastic gradient descent with momentum has been used in this work, which is a moving average that speeds up the process of finding the minimum valley. It is better than stochastic gradient descent due to its better estimation for the direction of stepping, and Its Beta is a hyperparameter which its value usually set as 0.9.

c) Feature extractor

Convolution neural networks work as feature extractors, as it has been clarified in chapter 4. There are many feature extractors which could be used such as ResNet-101, ResNet-50, ResNetXt-101, ResNet-100 and inception networks. The processing of ResNet networks is too heavy due to a large number of computations compared with the inception network. Due to the scope of hardware capabilities, the inception network has been used within this work.

d) Stride size

Stride refers to the number of pixels jumped by the filter. Smaller stride size means more information to encode, therefore better change for better representation. Larger stride size means higher step sides and, consequently, fewer computations. Thus the stride size has been reduced to 8 pixels in model 3 compared with 16 pixels in model 1. Unfortunately, this gave instant improvements but then turned out too bad results in evolutions; thus, no additional experiment had done to stride size.

e) Batch size

It refers to the number of examples forwarded to the neural network in one iteration. Due to the scope of hardware capabilities, I used just a batch size of 1.

f) Maximum number of proposals

Proposals are regions where objects may be found, and the next stage tries to detect objects in these regions. So increasing the number of proposals may lead to increase accuracy but needs more computational processing. The maximum number of proposals had been increased at the beginning to 320 and 350. There were not any clear or slight improvements. Trying to reduce the number of maximum proposals gave me a perfect result, and the optimal value of the try and error experiments was 100 for model 5.

5.3.5 Training results

This thesis uses TensorBoard to measure different kinds of loss's arguments such as classification_loss, localization_loss mask_loss, objectness_loss, and Total Loss. Which gives a clear representation of training.

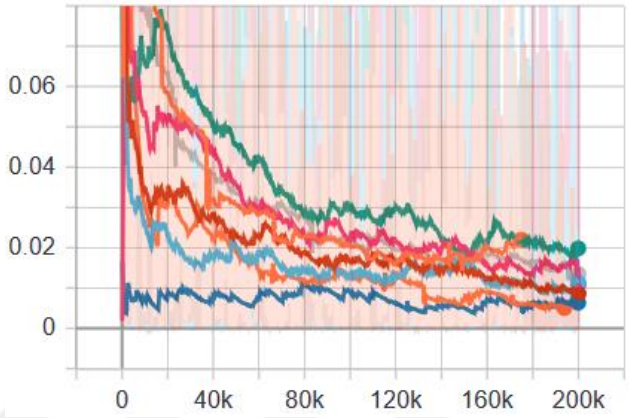
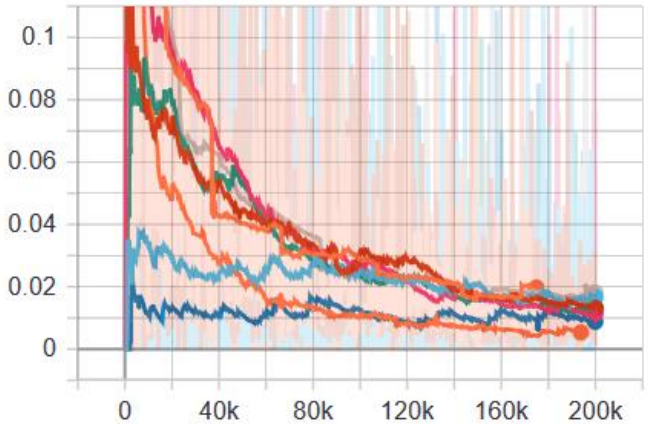
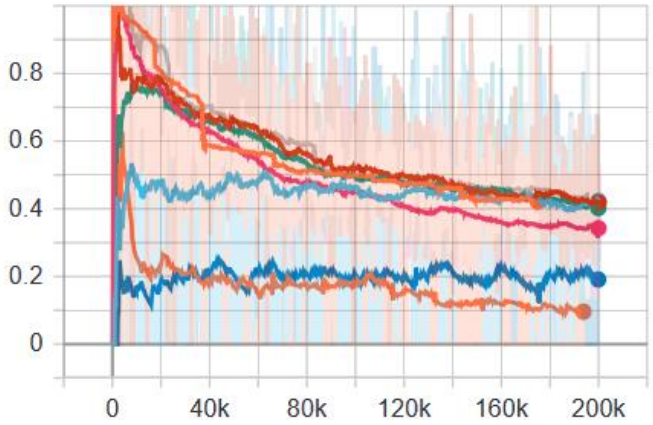
Classification_loss is loss resulted from the classification of a detected object as a plane. Localization_loss due to bounding box regressor for RPN or box classifier. Objectness_loss: loss due to classification whether bounding boxes are planes or background.Mask_loss: loss due to error pixels assignment of one class to another different class.Total_loss: It is the summation of all losses.

Table 5.7: Training results parameters.

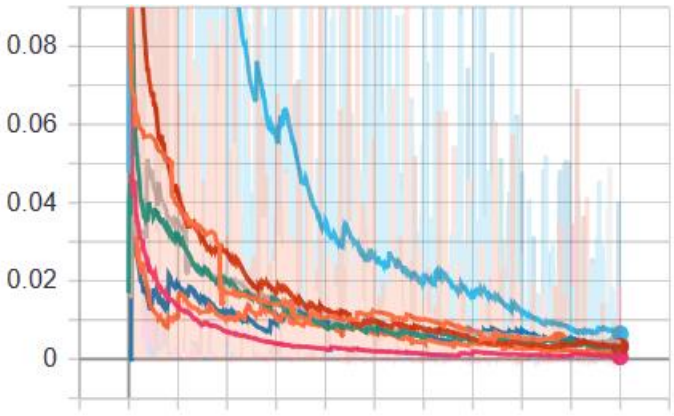
Training Results Parameters		Models							
		1	2	3	4	5	6	7	8
Box classifier Loss	classification_loss	0.022	9.9×10^{-3}	9.77×10^{-3}	0.016	0.019	0.015	**5×10^{-3}**	6.4×10^{-3}
	localization_loss	0.019	0.013	0.017	0.011	0.013	0.018	**5.35×10^{-3}**	8.8×10^{-3}
	mask_loss	0.42	0.42	0.42	0.35	0.4	0.4	**0.096**	0.19
RPN Loss	localization_loss	4.88×10^{-3}	3.21×10^{-3}	6.4×10^{-3}	**6.11×10^{-4}**	2.9×10^{-3}	4.3×10^{-3}	2.2×10^{-3}	3.4×10^{-3}
	objectness_loss	4×10^{-3}	3.4×10^{-3}	6.5×10^{-3}	**5.1×10^{-4}**	2.7×10^{-3}	3×10^{-3}	8.6×10^{-4}	2.7×10^{-3}
Total Loss		0.47	0.45	0.46	0.36	0.44	0.45	**0.1**	0.2

Table 5.7 shows that model 7 has the best performance during the training process. Its box classifier losses and total loss are the smallest comparing with other models. Model 4 has an interesting response to its RPN loss. In total, the best performance obtained by model 7 then four then 8. Table 5.8 below gives details about each loss progress during training.

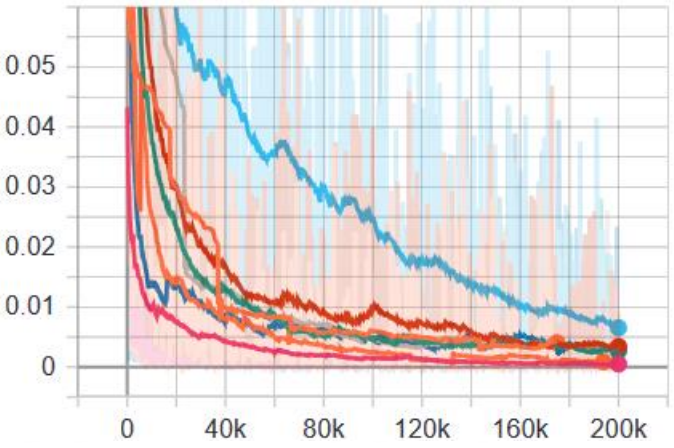
Table 5.8: Training results_ loss graphs.

Training Results	
Box Classifier Loss_ classification_loss	
Box Classifier Loss_ Localization _loss	
Box Classifier Loss_ Mask _loss	

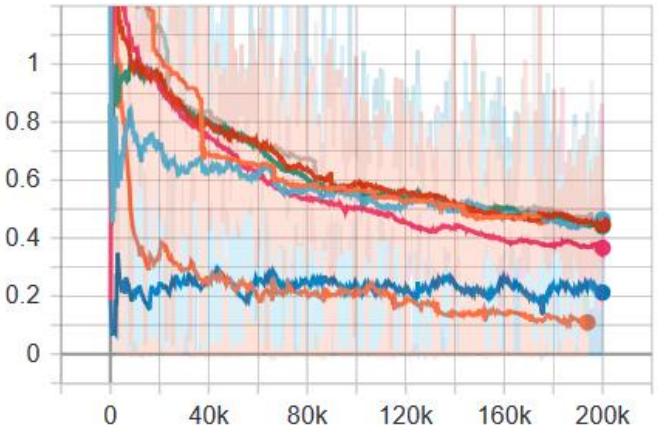
RPN Loss_ Localization _loss



RPN Loss_ objectness _loss



Total Loss



5.4 Evaluation

This thesis implements an evaluation process to measure the performance of plane detection models. Before going into it, there are some essential points needed to be highlighted.

There are multi metrics defined for object detections performances, which have been adopted by popular competitions such as PASCAL VOC, COCO, and Open image challenges[30][31][32]. All of them use mean average precision as a fundamental metric; however, there are some variations in definitions. Besides, COCO uses average recall as a new metric.

Confidence score and Intersection over Union (IoU) are fundamentals concepts in the evaluation process. The confidence score is the probability that the classifier's probability of finding an object inside an anchor box. These two terms used to determine whether detection is a true positive TP, a false positive FP, a true negative TN, or a false negative FN.

TP occurs if three conditions have been satisfied; confidence score higher than a threshold, a predicted class matches a class of ground truth, and IoU greater than a threshold. However, if one of the last two conditions has not met, then FP occurs. If a confidence score that has to detect a ground-truth is lower than the threshold, then it causes a false negative detection FN. Whereas if a confidence score that has not to detect anything is lower than the threshold, then it causes a true negative detection TN.

For analytical purposes, precision and recall terms are introduced. Where precision is the number of true positives divided by the sum of true positives and false positives, recall is the number of true positives divided by the sum of true positives and false negatives (number of ground-truths)

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{\#\ ground\ truths} \tag{5.2}$$

Making multi-variation in a threshold value of confidence score produces multiple values for both recall and precision. Mapping these values with x-y axis forms a precision-recall curve, which indicates the association between the two metrics.

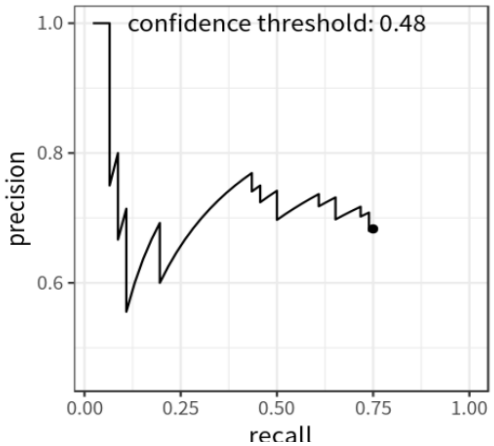


Figure 5.10: Precision-recall curve.

There is another curve that has a particular advantage to evaluate the effectiveness of detection proposals. It formed by making variations in the IoU threshold, which lead to different related values of recall.

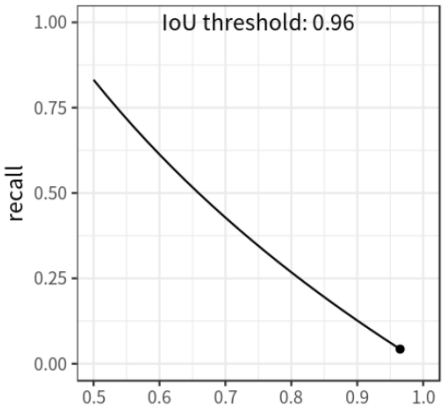


Figure 5.11: Recall-IoU curve.

Since there are different detectors used in object detection, then introducing other numerical metrics is needed, such as average precision (AP), Mean average precision (mAP), Average recall(AR), and Mean average recall (mAR). AR is the precision averaged over all recall levels. It is preferred to reduce wiggles in the curve using interpolated precision $\boldsymbol{p_{interp}}$ at a specific recall level, $\boldsymbol{r}$ then calculate the area under the resulted curve.mAP is the average of AP over all classes C. AR is the recall averaged over all IoU in the interval [0.5, 1.0]. mAR is the average of AR over all classes C

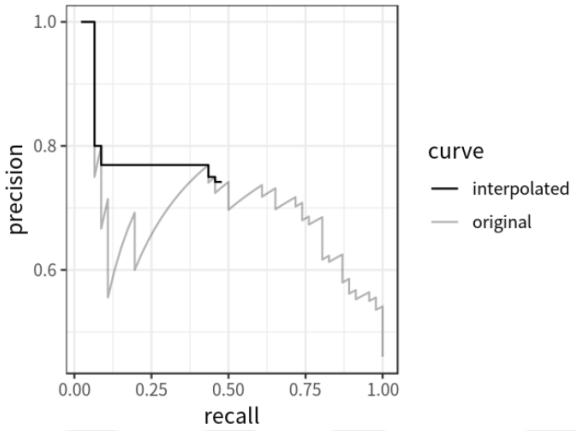


Figure 5.12: Average precision

$$p_{interp} = \max_{r' \geq r} p(r') \tag{5.3}$$

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i)\ p_{interp}(r_{i+1}) \tag{5.4}$$

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{5.5}$$

$$AR = 2 \int_{0.5}^{1} recall(IoU)\ dIoU \tag{5.6}$$

$$mAR = \frac{\sum_{i=1}^{C} AR_i}{C} \quad \textbf{(5.7)}$$

In this thesis, coco metrics have been used for evaluation purposes. COCO has multi mAP metrics defined by multi IoU- thresholds.

Table 5.9: COCO metrics' description 1.

Metrics	**IoU**	**Description**
mAP	0.50:0.5:0.95	10 IoU thresholds (i.e., 0.50, 0.55, 0.60, …, 0.95)
mAP	0.50	Pascal VOC metric
mAP	0.75	strict metric
Total # metrics		**12**
Metrics	**IoU**	**Description**
mAP Small	0.50:0.5:0.95	objects with area < 32^2
mAP Medium	0.50:0.5:0.95	32^2< objects with area< 96^2
mAP Large	0.50:0.5:0.95	objects with area > 96^2
Total # metrics		**30**
Metrics	**max**	**Description**
mAR	1	Max 1 object / image
mAR	10	Max 10 objects / image
mAR	100	Max 100 objects / image
Total # metrics		**3**
Metrics	**area**	**Description**
mAR	small	obj ects with area < 32^2
mAR	Medium	32^2< objects with area< 96^2
mAR	Large	objects with area > 96^2
Total # metrics		**3**

Table 5.10: COCO metrics' description 2.

Metric NO	**Average**	**IoU**	**Area**
1	**Average Precision**	0.50:0.95	all maxDets=100

2	**Average Precision**	0.50	all maxDets=100
3	**Average Precision**	0.75	all maxDets=100
4	**Average Precision**	0.50:0.95	small maxDets=100
5	**Average Precision**	0.50:0.95	medium maxDets=100
6	**Average Precision**	0.50:0.95	large maxDets=100
7	**Average Recall**	0.50:0.95	all maxDets= 1
8	**Average Recall**	0.50:0.95	all maxDets= 10
9	**Average Recall**	0.50:0.95	all maxDets=100
10	**Average Recall**	0.50:0.95	small maxDets=100
11	**Average Recall**	0.50:0.95	medium maxDets=100
12	**Average Recall**	0.50:0.95	large maxDets=100

Table 5.11: 12 performance metrics of COCO using the training set.

Model	Metrics											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0.899	0.99	0.98	0.848	0.917	0.931	0.369	0.927	0.927	0.882	0.947	0.952
3	0.898	0.99	0.98	0.819	0.918	0.978	0.399	0.918	0.918	0.849	0.943	0.985
4	0.269	0.577	0.205	0.053	0.311	0.688	0.197	0.348	0.389	0.157	0.448	0.755
5	0.892	0.99	0.99	0.875	0.903	0.903	0.389	0.924	0.924	0.908	0.931	0.925
6	0.92	0.99	0.99	0.863	0.943	0.961	0.403	0.940	0.941	0.888	0.963	0.977
7	0.921	0.99	0.983	0.875	0.937	0.969	0.402	0.942	0.943	0.901	0.958	0.982
8	0.902	0.99	0.99	0.582	0.897	0.914	0.434	0.926	0.928	0.580	0.924	0.939
9	0.816	0.965	0.928	0.737	0.896	0.935	0.412	0.840	0.842	0.772	0.922	0.957

Table 5.12: 12 performance metrics of COCO using the validation set.

Model	Metrics											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0.568	0.945	0.614	0.416	0.617	0.791	0.284	0.614	0.619	0.502	0.664	0.821
3	0.542	0.928	0.570	0.367	0.597	0.808	0.278	0.588	0.595	0.459	0.647	0.829
4	0.240	0.533	0.174	0.054	0.287	0.710	0.187	0.304	0.349	0.147	0.418	0.742
5	0.554	0.936	0.605	0.389	0.611	0.766	0.284	0.598	0.603	0.467	0.664	0.796
6	0.570	0.927	0.652	0.413	0.628	0.801	0.287	0.615	0.62	0.484	0.677	0.829

7	0.573	0.938	0.645	0.426	0.627	0.781	0.289	0.617	0.625	0.504	0.672	0.829
8	0.525	0.955	0.556	0.00	0.440	0.610	0.216	0.586	0.593	0.00	0.51	0.672
9	0.364	0.767	0.335	0.175	0.51	0.756	0.174	0.419	0.435	0.274	0.573	0.791

Table 5.13: 12 performance metrics of COCO using satellite set.

Model	Metrics											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0.448	0.911	0.384	0.226	0.477	0.637	0.230	0.505	0.520	0.386	0.540	0.697
3	0.405	0.845	0.317	0.184	0.434	0.611	0.221	0.466	0.484	0.329	0.508	0.659
4	0.244	0.531	0.174	0.061	0.29	0.465	0.157	0.334	0.375	0.234	0.394	0.579
5	0.397	0.813	0.300	0.188	0.427	0.575	0.209	0.455	0.482	0.335	0.506	0.656
6	0.456	0.902	0.398	0.209	0.495	0.637	0.241	0.520	0.524	0.343	0.555	0.709
7	0.472	0.932	0.430	0.260	0.499	0.687	0.240	0.530	0.538	0.387	0.560	0.738
8	0.393	0.852	0.250	0.245	0.436	0.456	0.211	0.480	0.506	0.374	0.540	0.532
9	0.314	0.693	0.244	0.106	0.360	0.491	0.190	0.380	0.409	0.224	0.441	0.612

Model 6 shows the best performance among other models. As expected, due to its lower learning rate. Model 5 shows perfect results due to a decrease in the max proposal regions to 100. In general, table 5 and table 6 give promise values in terms of COCO metrics with comparing to other researchers' results, which implement satellite images.

Table 5.13 is a benchmark that shows the detection accuracy for all eight models on satellite images; 200 images have been taken randomly from airplane categorization of the NWPU-RESISC45 dataset [39]. Although Satellite images are different from drone images, the accuracy values seemed to be close to others' results [37], which indicates the reliability of “Airplane detection using deep learning algorithms with drone images.”

Figures (5.13-5.20) give some visual implementations for these results, including TP, FP, TN, and FN.

Comparing the evaluation results for model 1 vs. model 8, There is no improvement of accuracies associated with an increase in the number of images in the dataset. Given that the additional images do not contain airplane objects, but they had taken from the same environment, i.e., the airport.

Comparing the evaluation results for model 7 vs. model 8, we can see the improvements of accuracies with model 7 associated with using high-resolution images, except small scale areas metrics they missed, and some accuracies did not improve.

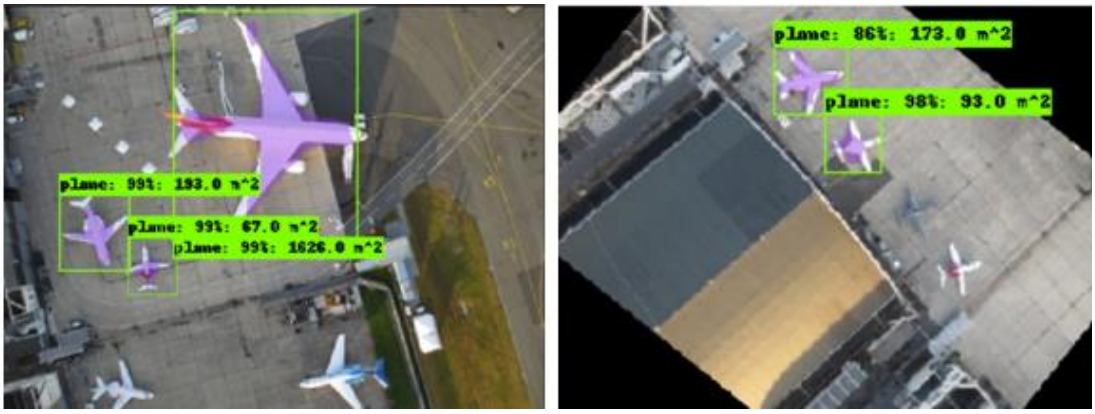


Figure 5.13: Detection samples - model 4.

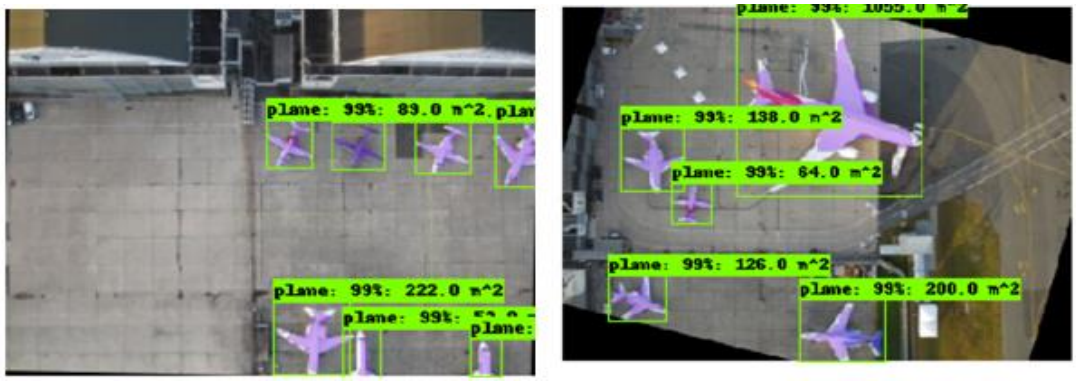


Figure 5.14: Detection samples - model 5.

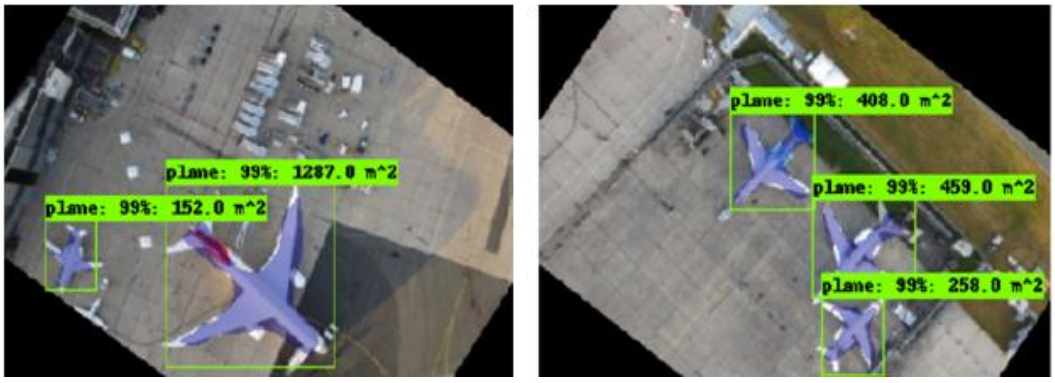


Figure 5.15: Detection samples - model 6.

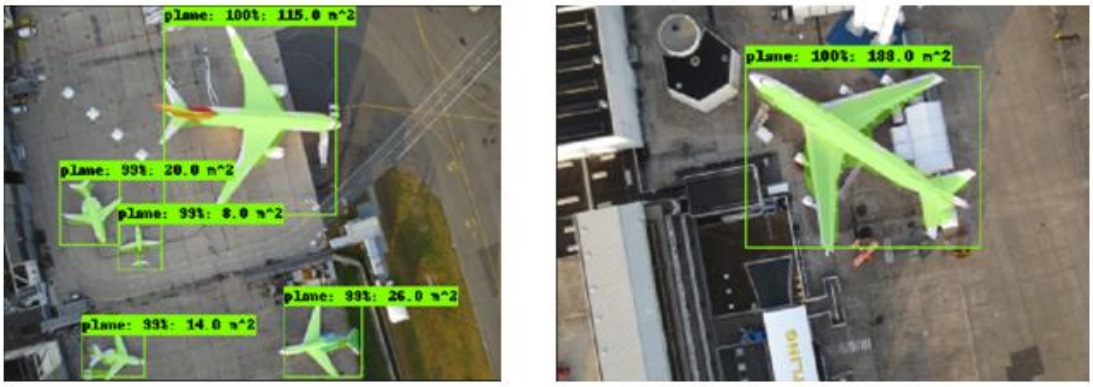


Figure 5.16: Detection samples - model 1.

Figure 5.17: Detection samples - model 2.

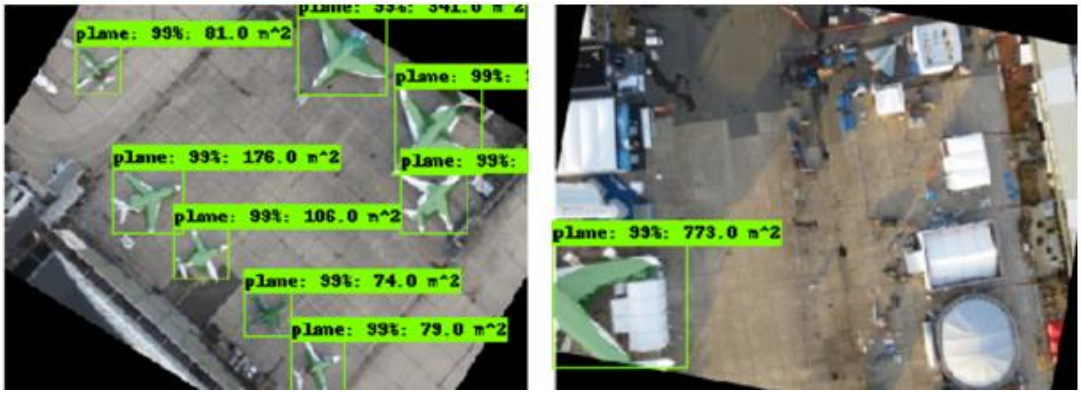


Figure 5.18: Detection samples - model 3.

Figure 5.19: Detection samples - model 7.

Figure 5.20: Detection samples - satellite images.

5.5 Plane Surface Area Calculation

The eBee perfect drone had taken the dataset images. This type of drones has one option to mount its camera, which makes its direction looking straight down or pointing to the nadir direction or the point directly below the camera [26]. The following figure introduces a visual representation of the camera sensor and field of view (FOV). Using camera sensor width (W_s), image width (W_I) focal length(FL), and drone altitude (h), the ground sample distance GSD or ground resolution can be calculated[33][34].

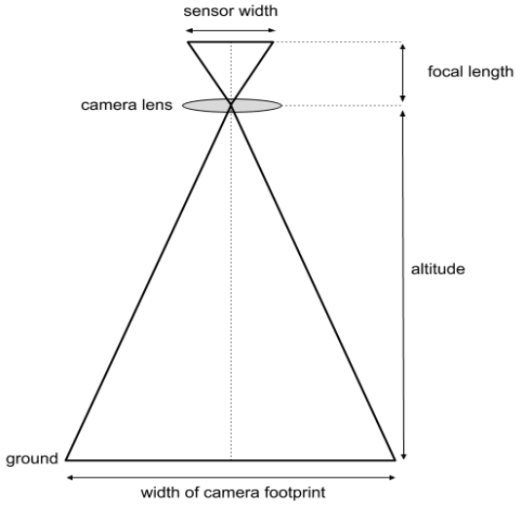


Figure 5.21: GSD calculation parameters.

Table 5.14: Camera spesifications.

SensorFly S.O.D.A camera	
Sensor	1 inch (12.75mm x 8.5 mm)
RGB lens	F/2.8-11, 10.6 mm (35 mm equivalent: 29 mm)
RGB resolution*	5,472 x 3,648 px
Exposure compensation	±2.0 (1/3 increments)
RGB shutter	Global Shutter 1/30 – 1/2000s
White balance	Auto, sunny, cloudy, shady

*Downloaded dataset has the size of 4608 x 3456 px

$$\text{GSD} = \frac{W_s \times h \times 100}{FL \times W_I} = \frac{12.75\,mm \times 120\,m \times 100}{10.6\,mm \times 4608\,px} = 3.13\,cm/px \qquad \textbf{(5.8)}$$

Considering this opportunity of GSD, This thesis approximately calculates a surface area (A_{appr}) for detected planes by multiply the total number of pixels (*# px*) for each airplane by GSD value. The total number of pixels for each airplane determined by detected masks since masks have tolerances with edges coverage; these calculations are approximated.

$$\text{A}_{\text{appr}} \cong px \times GSD \qquad \textbf{(5.9)}$$

Due to images resizing, there is a need to rescale the approximate areas to match the original dimensions. For the first ***seven*** models, the input images were with size ***304 x 231*** px. Thus the reducing scale is ***1:15*** for each width and hight. As a result, the multiplication factor $\boldsymbol{15^2}$ be applied to each approximate area.

Most airplane manufacturers do not offer precise numbers about their planes' surface area. Instead, they offer the plane's dimensions. Airports may have calculated surface areas for each plane within the airports' database. Airports may do that kind of calculations using data extracted from dimensions of airplanes, or may they have officers who have sufficient sense and experience to compare measured approximated surface area with real ones. The benefits of approximated area data will be used and not be lost.

Figure 5.22 illustrates an example of surface area dimensions. Companies usually give wings surface area, but the remaining area is left without calculations. Figures (5.23- 5.26) give some visual implementations for surface area detection.

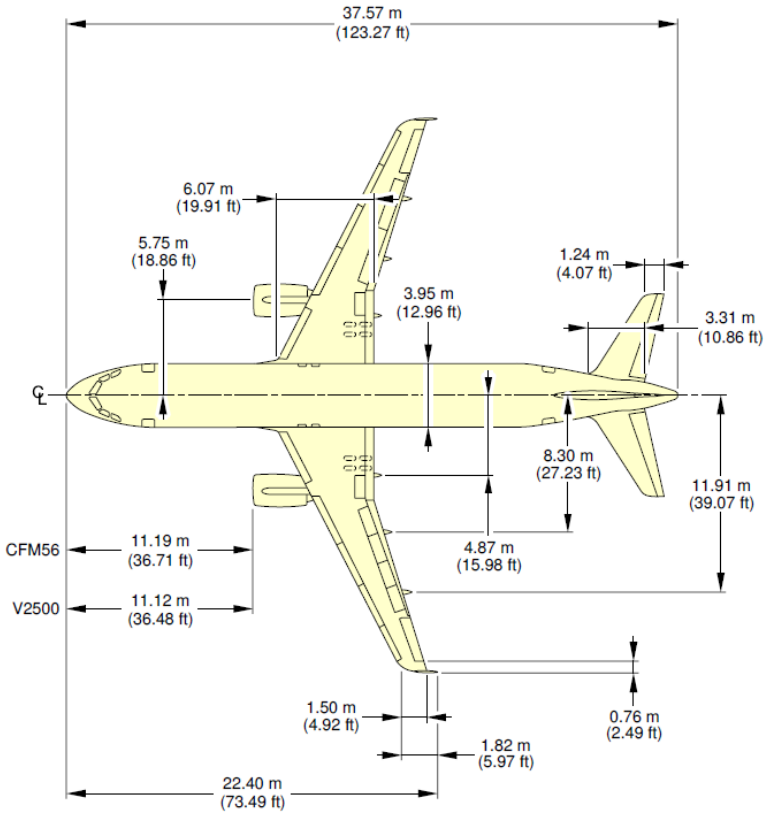


Figure 5.22: Airplane dimensions example-airbus a320 [35].

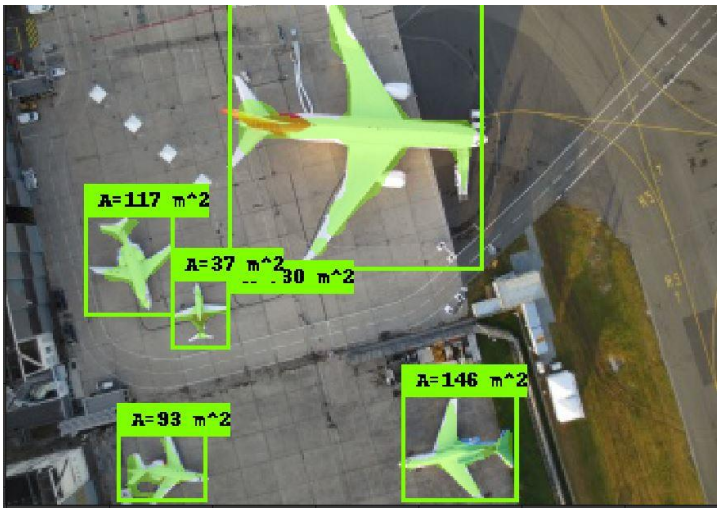


Figure 5.23: Area detection - model 1.

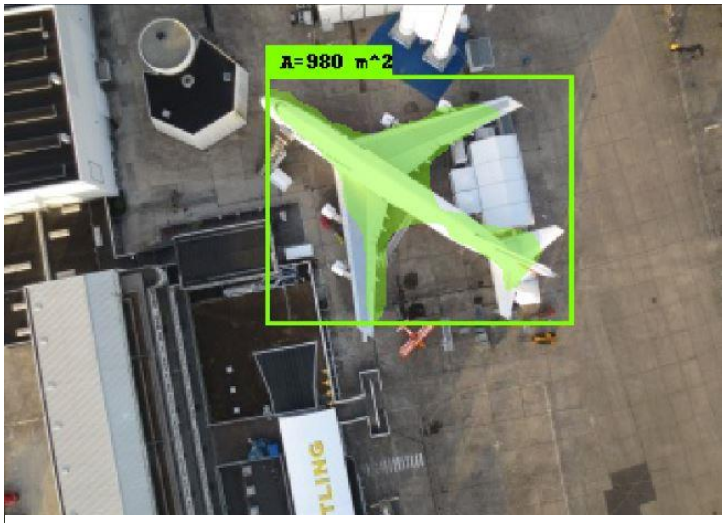


Figure 5.24: Area detection - model 4.

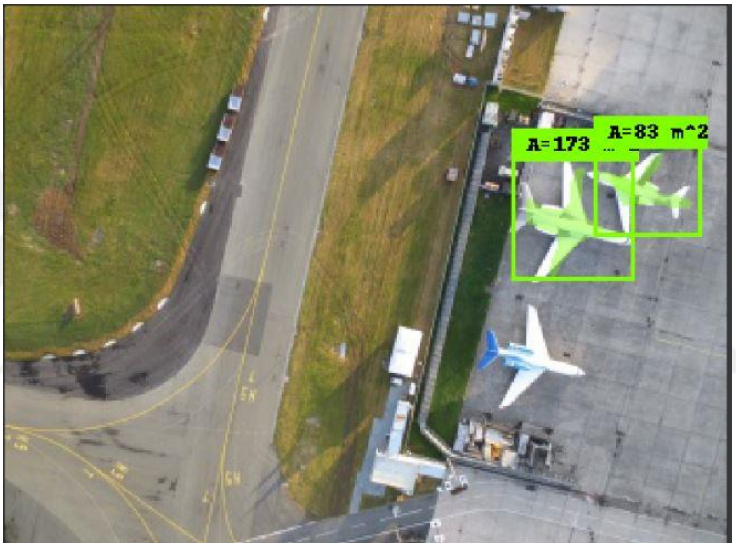


Figure 5.25: Area detection with one object lost – model 4.

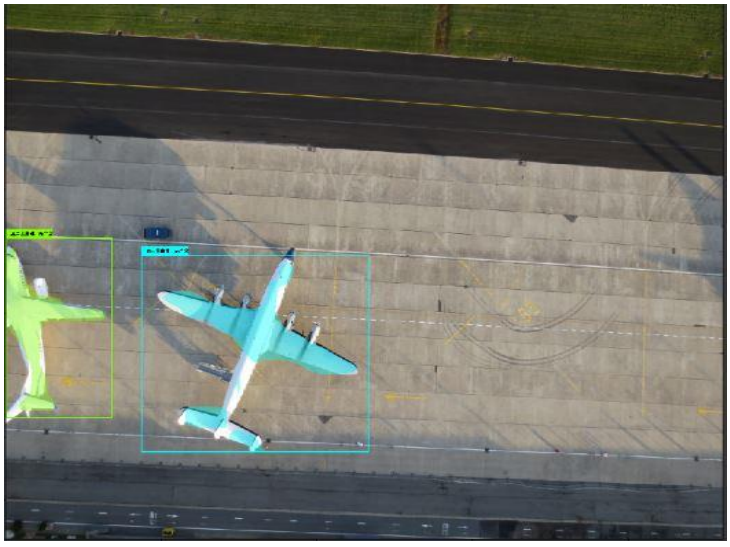


Figure 5.26: Area detection - model 8.

If the airport has a poor database, then this thesis offers an additional feature to detect the type of airplane using its length since most manufacturers offer planes length. This approach inherited from the previous one by finding the shortest pairwise distances between pixels within the masked area, then finding the furthest distance. The furthest distance could be computed between pixels, which are located as vertices in a convex hull, thus using the convex hull algorithm reduces computations for a fewer number of pixels.

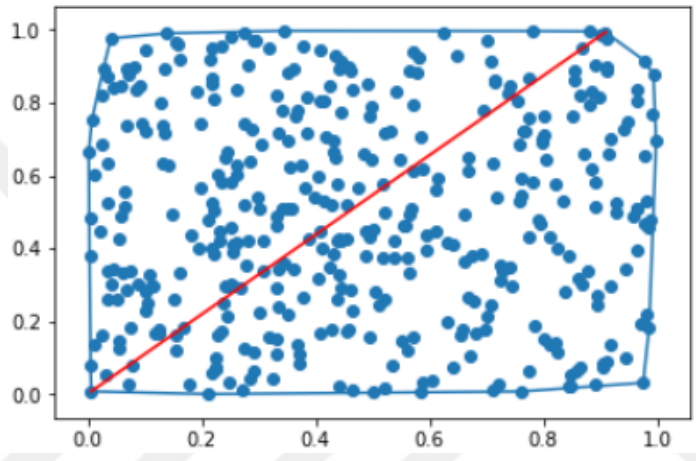


Figure 5.27: Convex hull algorithm.

Fortunately, there is a list of aircraft offered by the International Paris Air Show. This show happens in the same airport in which this thesis uses its dataset [36]. Thus the evaluation of detection performance could be performed according to airplanes' lengths through averaging detected length over the original validation set.

$$\text{Detected length avg} = \frac{\sum_{j=1}^{j=n} x_j}{n} \tag{5.10}$$

Where j is the number of validation images, x is the validation image, and n is the total number of validation set used in the evaluation.

Table 5.15: Length detection evaluation.

Length Detection Accuracy %								
plane	Model							
	1	2	3	4	5	6	7	8
LockheedMartin-LM100J	98	99	98	98	99	99	98	99
GULFSTREAM-G-280	91	99	93	98	98	99	91	80
GULFSTREAM-G-550	78	78	73	75	78	78	71	78
GULFSTREAM-G-650	86	95	92	91	91	90	86	98
Cessna-Citation CJ4	91	97	-	96	91	91	85	77
Cessna-Citation M2	77	85	-	85	92	73	96	69
Boeing 787-8	97	99	99	99	95	99	60	96
Airbus A-380	99	98	99	100	99	100	88	95
Airbus A-320	53	53	-	56	56	56	43	53
Avrage	86	**89**	-	**89**	**89**	87	80	83

According to statistics provided in table 5.15, experiments showed that Best estimation for lengths was for models 2, 4, and 5 with average accuracy 0f 89%. Model 6 has approximately the same average, 87%, and it has a perfect detection performance, as discussed in the previous section. Model 3 failed in detection with some airplane types, so its results have been ignored. The following figures (5.28-5.33) give visual implementations for these results.

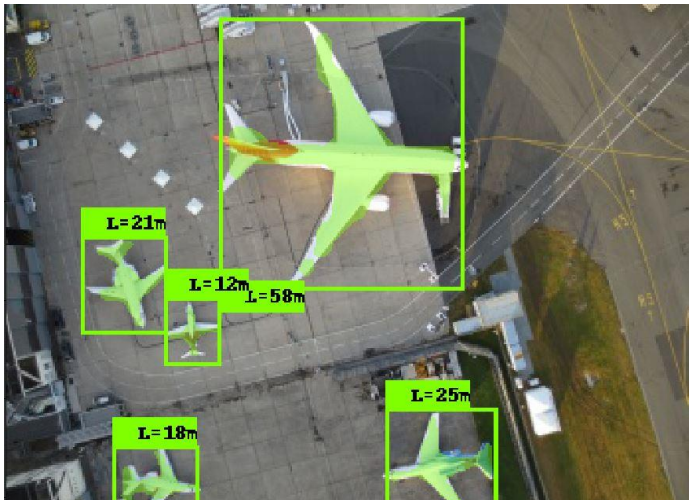


Figure 5.28: Length detection - model 1.

Figure 5.29: Length detection - model 1.

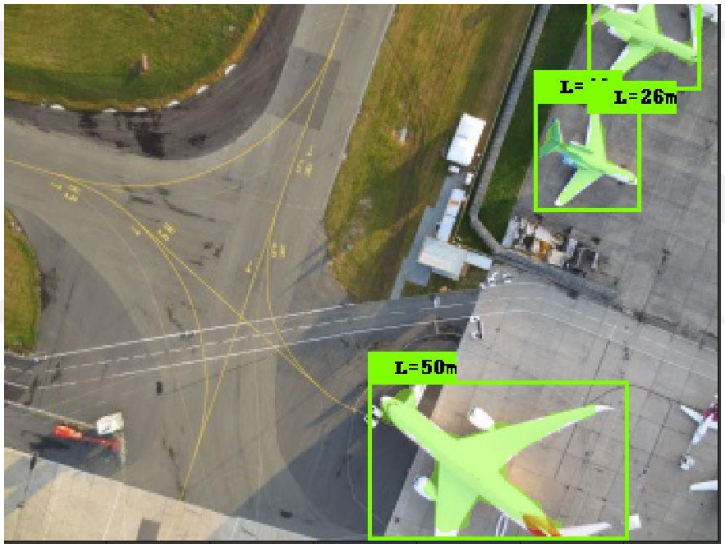


Figure 5.30: Length detection - model 2.

Figure 5.31: Length detection - model 2.

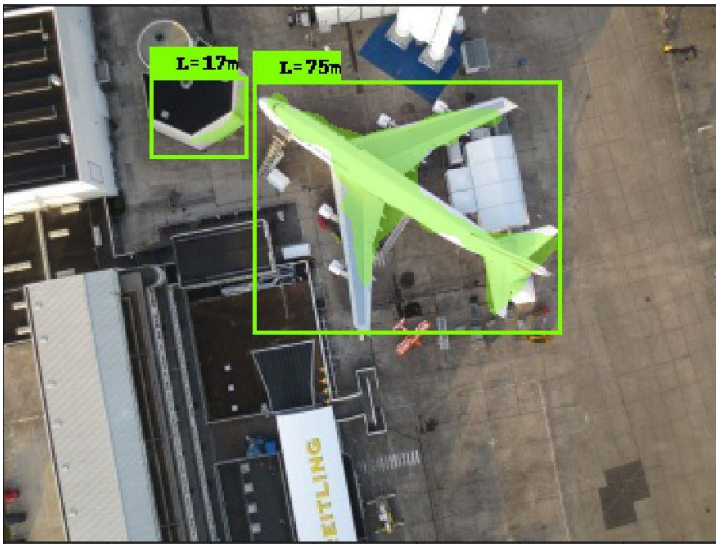


Figure 5.32: FP Length detection- model 3.

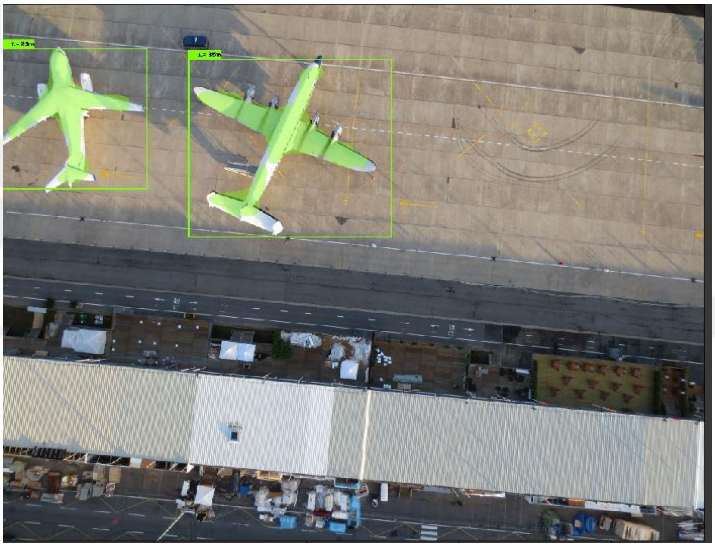


Figure 5.33: Length detection - model 8.

Detected length values have been assigned to the closest actual lengths to determine the airplane's type. Shortcut names used to make proper output annotations. Finally, the confusion matrix provided to add a quantitive evaluation.

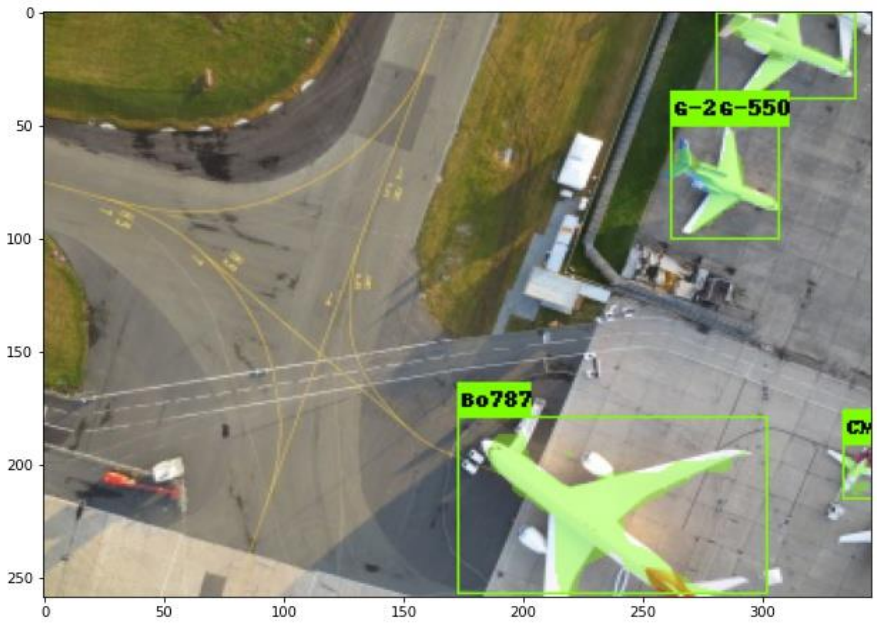


Figure 5.34: True assigned airplane's type for Bo787and Cm2; but false assignment for G-550 instead of G-650

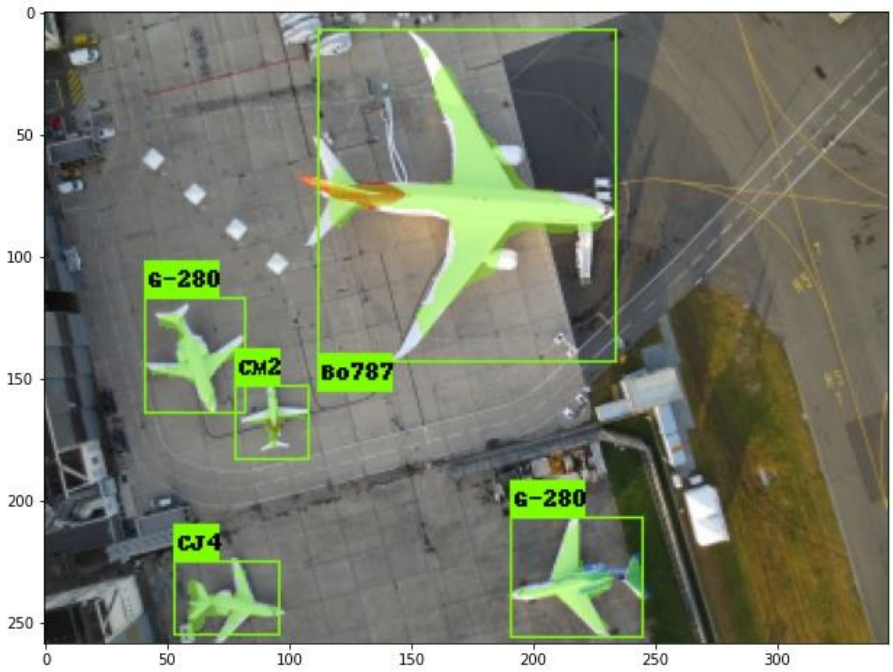


Figure 5.35: True assigned airplane's type for Bo787, G-550, Cm2, G-280 and CJ4

Table 5.16: Airplanes full names, shortcut names, and actual lengths.

Plane Full Name	Plane Shortcut	Actual length
Lockheed-Martin-LM100J	LM100J	35
GULFSTREAM-G-280	G-280	20
GULFSTREAM-G-550	G-550	29
GULFSTREAM-G-650	G-650	30
Cessna-Citation CJ4	CJ4	16
Cessna-Citation M2	CM2	13
Boeing 787-8	Bo787	57
Airbus A-380	A-380	73
Airbus A-320	A-320	38

Table 5.17: Airplane type identification-confusion matrix.

	LM100J	G-280	G-550	G-650	CJ4	CM2	Bo787	A-380	A-320
LM100J	9			2					1
G-280		20	4		2				2
G-550		1	10	3					1
G-650	2		2	8					1
CJ4		3			11	3			
CM2						31			
Bo787							12	2	
A-380								6	
A-320	2						1	2	4

Through the confusion matrix, we can see some errors in airplanes' type identifications. These errors are associated with airplanes that have close lengths, for example, G-550 and G-650, with a difference of 1 meter. Another reason is that some images contain cropped airplanes which affect the lengths detection. So proper choosing of place and time while acquiring the images is so important, which could be solved by scheduling the drone flights at specific times and proper places. In the future, additional detection for wings surface area will be provided to enhance the airplane type identification.

6. CONCLUSION

Inside this thesis, an easy and low-cost approach had been provided to support airplane traffic control in airports. The suggested approach is using a deep learning model with aerial images collected by drones to detect airplanes as a first step. The next step is using data provided to identify the airplane type according to its surface area then to its length.

This approach adds a universal contribution to any country in the world can use it. It needs just drones to feed the system with aerial images rather than a satellite approach that needs advanced and high-cost technology.

The flow of this thesis provides a clear explanation, taking that this work is a year recap of working in the research field, and it may be used as a reference for other research and publications. The thesis starts by introducing some terminologies and clarify some headlines in the introduction section, then goes through deep learning background and theory, simplifies different concepts with a broad, straightforward explanation for different mathematical formulas, and finally ending by practical implementation and results.

Evaluation gives promise values in terms of COCO metrics with comparing to other researchers' results, which their results based on satellite images. Although Satellite images are different from drone images, the accuracy values seemed to be close to others' results as a benchmark, which indicates the reliability of this approach.

REFERENCES

[1] **Zhao, Z.Q, Zheng, P, Xu, S.T and Wu, X**, (2019). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), pp.3212-3232.

[2] **Girshick, R, Donahue, J, Darrell, T and Malik, J**,(2014). 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).

[3] **Girshick, R.,**(2015). Fast r-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[4] **Ren, S, He, K, Girshick, R and Sun, J**.,(2015). Faster r-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[5] **He, K, Gkioxari, G, Dollár, P and Girshick, R**., (2017). Mask r-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[6] **Traory, B**,(2020) "automatic airplane detection using deep learning techniques and very high-resolution satellite images." Istanbul technical university, Istanbul, Turkey

[7] **Nielsen, M.A**., (2015). Neural networks and deep learning (Vol. 2018). San Francisco, CA, USA:: Determination press.

[8] **Zhang, L, Zhang, L and Du, B,** (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. IEEE Geoscience and Remote Sensing Magazine, 4(2), pp.22-40.

[9] **Chen, X, Xiang, S, Liu, C.L and Pan, C.H**.,(2014). Aircraft detection by deep convolutional neural networks. IPSJ Transactions on Computer Vision and Applications, 7, pp.10-17.

[10] **Malladi, C.**, (2017). Detection of Objects in Satellite images using Supervised and Unsupervised Learning Methods.

[11] **Romero, A, Gatta, C and Camps-Valls, G**.,(2015). Unsupervised deep feature extraction for remote sensing image classification. IEEE Transactions on Geoscience and Remote Sensing, 54(3), pp.1349-1362.

[12] **Tang, J, Deng, C, Huang, G.B. and Zhao, B**, (2014). Compressed-domain ship detection on a space-borne optical image using deep neural network and extreme learning machine. IEEE Transactions on Geoscience and Remote Sensing, 53(3), pp.1174-1185.

[13] **Chen, X, Xiang, S, Liu, C.L and Pan, C.H**., (2014). Vehicle detection in satellite images by hybrid deep convolutional neural networks. IEEE Geoscience and remote sensing letters, 11(10), pp.1797-1801.

[14] **Cheng, G, Han, J, Guo, L, Qian, X, Zhou, P, Yao, X, and Hu, X**,(2013). Object detection in remote sensing imagery using a discriminatively trained mixture model. ISPRS Journal of Photogrammetry and Remote Sensing, 85, pp.32-43.

[15] **Bi, F, Zhu, B, Gao, L and Bian, M,** (2012). A visual search inspired computational model for ship detection in optical satellite images. IEEE Geoscience and Remote Sensing Letters, 9(4), pp.749-753.

[16] **Hu, G, Yang, Z, Han, J, Huang, L, Gong, J and Xiong, N**, (2018). Aircraft detection in remote sensing images based on saliency and convolution neural network. EURASIP Journal on Wireless Communications and Networking, 2018(1), pp.1-16.

[17] **Tang, T, Zhou, S, Deng, Z, Zou, H, and Lei, L**.,(2017). Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining. Sensors, 17(2), p.336.

[18] **Zhang, S, Wu, R, Xu, K, Wang, J and Sun, W.**, (2019). R-CNN-based ship detection from high-resolution remote sensing imagery. Remote Sensing, 11(6), p.631.

[19] **LB, W**, (2017). A high-resolution optical satellite image dataset for ship recognition and some new baselines.

[20] **LeCun, Y, Bottou, L, Bengio, Y and Haffner, P**., (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[21] **Krizhevsky, A, Sutskever, I, and Hinton, G. E**.,(2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[22] **Simonyan, K and Zisserman, A,** (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[23] **He, K, Zhang, X, Ren, S and Sun, J.,** (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[24] **Szegedy, C, Liu, W, Jia, Y, Sermanet, P, Reed, S, Anguelov, D, Erhan, D, Vanhoucke, V and Rabinovich, A**, (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[25]**Url-1**<https://www.faa.gov/uas/recreational_fliers/where_can_i_fly/airspace_restrictions /flying_near_airports/.> 15/01/2020

[26] **Url-2 <** https://www.sensefly.com/dataset/airport-drone-dataset/> 02/01/2020

[27] **Url-3<** https://www.tensorflow.org/tensorboard/get_started> 02/12/2019

[28] **Dai, J, He, K and Sun, J**, (2016) Instance-aware semantic segmentation via multi-task network cascades. In CVPR.

[29] **Li, Y, Qi, H, Dai, J, Ji, X, and Wei, Y** , (2017) Fully convolutional instance-aware semantic segmentation. In CVPR

[30]**Url-4<**http://host.robots.ox.ac.uk/pascal/VOC/voc2012/htmldoc/devkit_doc.html> 12/12/2019

[31]**Url-5<** http://cocodataset.org/#detection-eval> 02/11/2019

[32]**Url-6<** https://storage.googleapis.com/openimages/web/object_detection_metric.html > 02/11/2019

[33]**Url-7<** https://www.propelleraero.com/gsd-calculator > 07/02/2020

[34]**Url-8<**https://learn.droneblocks.io/courses/droneblocks-math-with-drones/lectures/2908697> 07/12/2019

[35]**Url-9<**://www.airbus.com/aircraft/passenger-aircraft/a320-family.html> 17/01/2020

[36]**Url-10<** https://www.siae.fr/en/the-show/list-of-aircraft.htm > 02/02/2020

[37]**Soydaş**, **M** (2019),'' Aircraft Detection From Large Scale Remote Sensing Images With Deep Learning Techniques,'' MSc Thesis, Istanbul Technical University,

[38] **Khan, M, Yousaf, J, Javed, A, Nadeem, N and Khurshid, K**., (2017). Automatic target detection in satellite images using deep learning. J. Space Technol, 7(1), pp.44-49.

[39] **Cheng, G, Han, J, Lu, X**. (2017), "Remote Sensing Image Scene Classification: Benchmark and State of the Art," Volume: 105, DOI: 10.1109/JPROC.2017.2675998.

[40]**Url-11**< https://www.youtube.com/watch?v=aircAruvnKk>06/07/2020

[41]**Url-12**<https://www.coursera.org/lecture/neural-networks-deep-learning/deep-l-layer-neural-network-7dP6E>06/07/2020

[42] **Url-13** < http://neuralnetworksanddeeplearning.com/chap3.html>06/07/2020

[43]**Url-14**<https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/>

CURRICULUM VITAE

Personal Information

Name	WALEED AL-SHAIBANI
Birth Date	15/10/1991
Nationality	Yemeni
Mobile	+90 5536843228 +967 771800274
Email	tawfiqwaleed@gmail.com
Address	SultanSalim Cad,SultanSalim Mah,No18/ Istanbul , Turkey. 40 St, Kelaba/ Taiz City, Yemen

Education

2010-2015	B.Sc Electronics Engineering
2018-Now	M.Cs Satellite Communication And Remote Sensing

Work Experiences

2016-2018	Lecturer, University of Science and Technology-Jordan
2015-2017	Maintenance Engineer, AlKbuse Group Company- Jordan