ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AUTOMATIC AIRPLANE DETECTION USING DEEP LEARNING TECHNIQUES AND VERY HIGH-RESOLUTION SATELLITE IMAGES

M.Sc. THESIS

Bakary TRAORE

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Prof. Dr. Elif SERTEL

JANUARY 2020

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AUTOMATIC AIRPLANE DETECTION USING DEEP LEARNING TECHNIQUES AND VERY HIGH-RESOLUTION SATELLITE IMAGES

M.Sc. THESIS

Bakary TRAORE
(705181004)

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Prof. Dr. Elif SERTEL

JANUARY 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

DERİN ÖĞRENME TEKNİKLERİ VE ÇOK YÜKSEK ÇÖZÜNÜRLÜKLÜ UYDU GÖRÜNTÜLERİ KULLANILARAK OTOMATİK UÇAK TESPİTİ

YÜKSEK LİSANS TEZİ

Bakary TRAORE
(705181004)

İletişim Sistemleri Anabilim Dalı

Uydu Haberleşmesi ve Uzaktan Algılama Programı

Tez Danışmanı: Prof. Dr. Elif SERTEL

OCAK 2020

Bakary TRAORE, a M.Sc. student of ITU Informatics Institute student ID 705181004, successfully defended the thesis entitled "AUTOMATIC AIRPLANE DETECTION USING DEEP LEARNING TECHNIQUES AND VERY HIGH-RESOLUTION SATELLITE IMAGES", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Elif SERTEL** Istanbul Technical University	.....
Jury Members :	**Dr. Öğr. Üyesi Ugur ALGANCI** Istanbul Technical University	.....
	Prof. Dr. Üyesi Bulent BAYRAM Yıldız Technical University	.....

Date of Submission : 12 December 2020
Date of Defense : 21 January 2020

To my family,

FOREWORD

A master's degree is defined as the second cycle in many countries and universities. It is awarded by universities or colleges upon successfully completion of all courses of study that showing mastery of a specific domain. In my case, the domain is Satellite Communication and Remote Sensing with "AUTOMATIC AIRPLANE DETECTION USING DEEP LEARNING TECHNIQUES AND VERY HIGH-RESOLUTION SATELLITE IMAGES" as my thesis topic. To achieve all this process of mastery, we need to review many preview works, to access to many materials and guided by someone who already acquired the knowledge and is expert in that field of study. It is in this sense that I had the chance to have Prof. Dr. Elif SERTEL as my supervisor. Prof. Dr. Elif SERTEL provided me all necessary materials and even gave me the permission to have access to the laboratory of Istanbul Technical University-Center of Satellite Communication and Remote Sensing (ITU-CSCRS). Prof. Sertel was always available to check my work and to give a new and better direction. So, I would like to express my deep appreciation and thanks to my dear supervisor, Prof. Dr. Elif SERTEL.

In the laboratory, I had a change to meet again my dear Ass. Prof. Dr.Ugur ALGANCI. I thank him to have provided me an excellent computer environment for my practical applications and a wonderful introduction to remote sensing.

A thank from me is also going to Ms. Esma MUTLUER and Mr. Sinan SIVRI to have checked my work during the process.

I also thank deeply ITU-CSCRS for this big opportunity that helped me to achieve my research successfully.

I thank all the people and my dear family who encourage and support me.

JANUARY 2020

Bakary TRAORE
Mechatronics Engineer

TABLE OF CONTENTS

ABBREVIATIONS

AE : Autoencoder
ANN : Artificial Neural Network
API : Application Program Interface
ATR : Automatic Target Recognition
BOVW : Bag-Of-Visual-Words
BP : Backpropagation
CCRS : Canada Centre for Remote Sensing
Ce : Cross-entropy
CGI : Common Gateway Interface
CNN : Convolutional Neural Network
COCO : Common Objects in Context
ConvLSTM : Convolutional Long Short-Term Memory
CPU : Central Processing Unit
CT : Computerized Tomography
CUDA : Computer Unified Device Architeccture
CUDNN : The NVIDIA CUDA Deep Neural Network Library
DBM : Deep Boltzmann Machine
DBN : Deep Belief Network
DL : Deep Learning
DNA : Deoxyribonucleic Acid
DNN : Deep Neural Network
E : Sum-square error
Faster RCNN: Faster Region-based Convolutional Network
GPU : Graphics Processing Unit
HR : High-Resolution
HSI : Hyper-Spectral Image
IR : Infrared
ISPRS : International Society for Photogrammetry and Remote Sensing
ITU-CSCRS : Istanbul Technical University- Center of Satellite Communication and Remote Sensing
L : Total Loss
LL : Log-Likelihood
LR : Low Resolution
MAE : Mean Absolute Error
MLP : Multilayer Perceptron
MR : Mitral Regurgitation
MRE : The Mean Relative Error
MRI : Magnetic Resonance Imaging
MS : Multi-Spectral

MSE : Mean Square Error
NIR : Near Infrared
NN : Neural Network
Qc : Quadratic cost
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
SAE : Sparse Autoencoder
SAR : Synthetic-Aperture Radar
SC : Sparse Coding
SGD : Stochastic Gradient Descent
SIFT : Scale-Invariant Feature Transform
Sig(x) : Sigmoid function applied on x
SR : Sparsity Constraint
SSD : Single Shot Multibox Detector
stddev : Standard Deviation
SURF : Speeded Up Robust Features
SVM : Support Vector Machine
TL : Transfer Learning
UC Merced : University of California Merced
YOLO : You Only Look Once
3-D : 3-Dimensional

LIST OF TABLES

Page

LIST OF FIGURES

Page

AUTOMATIC AIRPLANE DETECTION USING DEEP LEARNING TECHNIQUES AND VERY HIGH-RESOLUTION SATELLITE IMAGES

SUMMARY

Object detection in high resolution remote sensing (RS) images is challenging problem in remote sensing imagery processing, especially for civil and military application, due to the complex object background and complex scenes. Many researches were conducted and are also under process in the field of remote sensing using deep learning (DL) techniques, a state-of-the-art technique, for automatic object detection. Some of them successes as well as others have failed. The successful researches have also faced some problems: High training time; low detection performance; high detection time; poor localization accuracy. Also, in their studies, different models lead to different results on a same given dataset.

The main concern in this thesis is to figure out the reasons of these problems by analyzing different components of some of the available models**.** In this research, we proposed airplane detection approach using different deep learning techniques applied to high-resolution satellite images. Two different well-known DL techniques such as Single Shot Multibox Detector (SSD) and Faster Region-based Convolutional Network (Faster RCNN) were used for our airplane detection. These techniques are the most used deep learning techniques for object detection. We present fifteen models with seven models based on SSD and eight based on Faster R-CNN. They were pretrained using Microsoft Common Objects in Context (MS COCO) dataset. Each model was settled with specific structure and hyper-parameters over time in order to determine the importance of the model architectures and hyper-parameters in object detection to overcome the above-mentioned problems. During the experiments, we encountered overfitting problem with some Faster R-CNN models, we overcame the problem by using regularizers with different weight values. The models were trained with a well-built dataset composing of small, medium, and large-scale airplane and non-airplane objects. Moreover, our dataset is a mix of images of high-resolution satellite images from different sources such as Northwestern Polytechnical University-REmote Sensing Image Scene Classification (WPU-RESISC45) dataset, WHURS19 dataset, Aerial Image Dataset (AID), and Istanbul Technical University-Center of Satellite Communication and Remote Sensing (ITU-CSCRS) dataset, used in several object detection applications. We used images from these datasets because there are composed of images with different resolutions, different scenes with different backgrounds, from different sensors and earth's regions. Also, the datasets were used in many scene classification and object detection projects and they produced state-of-the-art results. All images from these sources are very high-resolution satellite images and mainly created for scene classification and object detection. Our dataset, with this variety of sources, is composed of 1402 images in total with 7 701 airplane objects in 1167 images other images contain non-airplane objects such as Jet Plane, High Building, Residence, Storage Tank, and Pool. We used 80% of the dataset for training with 6 287 plane objects

in 1 119 images and the remaining for testing which includes 1 414 plane objects in 283 images.

We achieved airplane detection with fast training, accurate detection, and detecting targets regardless of their background and scene configuration. At the end, the models were analyzed and compared in terms of their performances on high-resolution satellite images, where we found out that the model architectures and hyper-parameters, for instance feature extractor and learning rate and many others, have a significant role in model training and the accomplishment of a desirable object detection result. It is our hope that this research will be beneficial for the researchers to avoid above mentioned problems in their future studies and applications.

DERİN ÖĞRENME TEKNİKLERİ VE ÇOK YÜKSEK ÇÖZÜNÜRLÜKLÜ UYDU GÖRÜNTÜLERİ KULLANILARAK OTOMATİK UÇAK TESPİTİ

ÖZET

Yüksek çözünürlüklü uzaktan algılama (RS) görüntülerinde nesne algılama, karmaşık nesne arka planı ve karmaşık sahneler nedeniyle, özellikle sivil ve askeri uygulamalar için uzaktan algılama görüntü işlemede zor bir sorundur. Otomatik nesne tespiti için en son teknoloji olan derin öğrenme (DL) teknikleri kullanılarak uzaktan algılama alanında birçok araştırma yürütülmüştür ve bu süreçler devam etmektedir. Bazıları başarıları ve diğerleri başarısız oldu. Başarılı araştırmalar da bazı sorunlarla karşılaşmıştır: Yüksek eğitim süresi; düşük algılama performansı; yüksek algılama süresi; zayıf yerelleştirme doğruluğu. Ayrıca, çalışmalarında, farklı modeller aynı veri kümesinde farklı sonuçlara yol açar.

Bu tezdeki ana endişe, mevcut modellerin bazılarının farklı bileşenlerini analiz ederek bu sorunların nedenlerini bulmaktır. Bu araştırmada, yüksek çözünürlüklü uydu görüntülerine uygulanan farklı derin öğrenme tekniklerini kullanarak uçak tespit yaklaşımını önerdik. Uçak tespitimiz için Tek Çekim Çok Kutuplu Dedektör (SSD) ve Daha Hızlı Bölge Tabanlı Evrişim Ağı (Faster RCNN) gibi iki farklı iyi bilinen DL tekniği kullanılmıştır. Bu teknikler nesne tespiti için en çok kullanılan derin öğrenme teknikleridir. SSD'ye dayalı yedi model ve daha Faster R-CNN'ye dayalı sekiz model içeren on beş model sunduk. Bağlamda Microsoft Ortak Nesneler (MS COCO) veri kümesi kullanılarak önceden eğitilmiştir. Her model, yukarıda belirtilen problemlerin üstesinden gelmek için nesne mimarisindeki model mimarilerinin ve hiper parametrelerin önemini belirlemek için zamanla spesifik yapı ve hiper parametrelerle yerleşmiştir. Deneyler sırasında, bazı Faster R-CNN modellerinde aşırı uyum sorunu ile karşılaştık, farklı ağırlık değerlerine sahip düzenleyiciler kullanarak sorunu aştık. Modeller, küçük, orta ve büyük ölçekli uçak ve uçak dışı nesnelerden oluşan iyi oluşturulmuş bir veri kümesi ile eğitildi. Ayrıca, veri setimiz WPU-RESISC45 veri seti, WHURS19 veri seti, AID ve ITÜ-UHUZAM gibi farklı kaynaklardan gelen yüksek çözünürlüklü uydu görüntülerinin bir karışımıdır. Bu veri kümelerindeki görüntüleri kullandık çünkü farklı çözünürlüklerde görüntüler, farklı arka planlara sahip farklı sahneler, farklı sensörlerden ve dünya bölgelerinden oluşuyor. Ayrıca, veri kümeleri birçok sahne sınıflandırma ve nesne algılama projesinde kullanılmış ve son teknoloji sonuçlar üretmiştir. Bu kaynaklardan gelen tüm görüntüler çok yüksek çözünürlüklü uydu görüntüleridir ve temel olarak sahne sınıflandırması ve nesne algılama için oluşturulur. Bu çeşitli kaynaklara sahip veri setimiz 1167 görüntüde 7 701 uçak nesnesiyle toplam 1402 görüntüden oluşuyor, diğer görüntüler Jet Düzlemi, Yüksek Bina, Konut, Depolama Tankı ve Havuz gibi uçak dışı nesneler içeriyor. Veri

setinin% 80'ini 1 119 görüntüde 6 287 düzlem nesnesiyle, 283 görüntüde 1 414 düzlem nesnesini içeren test için geri kalanı kullandık.

Arka plan ve sahne yapılandırmasına bakılmaksızın hızlı eğitim, doğru algılama ve hedefleri tespit etme ile uçak algılama gerçekleştirdik. Sonunda modeller, yüksek çözünürlüklü uydu görüntüleri üzerindeki performansları açısından analiz edildi ve karşılaştırıldı, burada model mimarilerinin ve hiper parametrelerin, örneğin özellik çıkarıcı ve öğrenme oranı ve diğerlerinin önemli bir rolü olduğunu keşfettik. model eğitiminde ve istenen bir nesne algılama sonucunun gerçekleştirilmesinde. Bu araştırmanın, araştırmacıların gelecekteki çalışmalarında ve uygulamalarında yukarıda belirtilen sorunlardan kaçınmaları için yararlı olacağını umuyoruz.

1. INTRODUCTION

Object detection in high resolution remote sensing images is challenging problem in remote sensing imagery processing due to the complex object background and complex scenes, and also due to image acquisition condition. The remote sensing images are acquired generally from high altitudes with atmospheric interferences, different acquisition geometry, illumination, and topography. Object detection is fundamental in remote sensing and in many applications such as military applications, intrusion detection, border security, advanced driver-assistance systems, etc. It is an important task for understanding high-resolution images. The main purpose of object detection is to determine whether a given remote sensing image contains a specific target of interest and determine the position information of the predicted target. Many man-made objects that were difficult to be detected are now detectable with the increased number of detection algorithms. Since the 1980s, geospatial object detection in remote sensing imagery is being widely studied [1], mainly using shallow features that were processed by skilled people who are experienced in the field and also often required domain-expertise. Their framework was built for specific tasks and at specific conditions that is if the conditions change even slightly, the framework which works well in a given task may fail in another task. These disadvantages led researchers in the field looking for a more robust and effective technique.

Object detection is a computer vision technology and applied to image to detect instance objects of a certain class such as face, building, human, airplane, or car. It is composed of classification and localization of objects in general. Many researches were conducted for automatic target detection using different techniques. In [2], they used the random convolutional network (RCNet), the spatial pyramid matching kernel (SPMK) method, the scale-invariant feature transform (SIFT) added to sparse coding (SSC) approach, and unsupervised feature-learning method called the saliency-guided sparse AE (SSAE) method to recognize 8 categories of object such as airplane, river, runway, residential, ocean, meadow, industrial and bare soil. In their work, dense low-

level feature descriptors were extracted for the local spatial patterns. The unlabeled feature measurements were used to learn a set of basic functions. And the low-level feature descriptors were explored to generate new sparse representation for the feature descriptors. Among those four techniques they used, the RCNet had the best performance with 94.53% and 98.78% accuracies on University of California Merced (UCM) and Google Earth of Sydney datasets, respectively. To learn the feature of each class, the RCNet was firstly performed on UCM dataset which includes 21 classes such as airplane, agricultural, beach, baseball diamond, chaparral, buildings, forest, dense residential, freeway, golf course, intersection, harbor, mobile-home park, medium residential, overpass, runway, parking lot, river, storage tanks, tennis court, and sparse residential, with 100 images for each class and 80% of the samples from each class assigned to training and the remaining samples for testing, and 500 cycles for training using stochastic gradient descent of 64 as batch size, 0.9 for momentum, 0.0005 as weight decay and 0.01 for the learning rate value, implemented in MATLAB 2014a environment using a CUDA kernel. And in the second part, they used images acquired from Google Earth of Sydney, Australia, with spatial resolution of 1.0 m and 7849x9073 pixels, with 25 samples from each class for training set and the remaining for testing. In total in Sydney dataset, there were 200 samples for training and 898 samples for testing for eight classes, including residential, meadow, airplane, industrial, ocean, bare soil, rivers, and runway. The same stochastic gradient descent was used but with a batch size of 32, 0.9 as momentum, 0.0005 as weight decay, and a learning rate value of 0.01 with 800 epochs for the whole training in the same implementation environment. In addition to these experiments from [2], many different deep learning techniques like supervised and unsupervised learning techniques (e.g. Restricted Boltzmann Machines (RBMs), Convolutional Neural Networks **(**CNNs) and Multilayer Perceptron (MLP)) were used by researchers for the object detection such as airplane [3,4,5,6,7,8]. They generate state-of-the-art results. Furthermore, the researchers, in the domain of deep learning, also developed Single Shot Multi-box Detector (SSD) model [9] and Faster Region-based Convolutional Neural Network **(**Faster R-CNN) model [10] which are the most used models in object detection nowadays [9,10,11,12]. SSD model and Faster R-CNN model are pre-trained models on natural images that can be also used to process very high-resolution satellite images [12]. In [12], airplane detection was proposed using different models such as

SSD model, Faster R-CNN, and You Look Only Once (YOLO)-v3 model [13] to DOTA dataset. Their image sizes were in the range of 800 x 800 to 4000 x 4000, within 1631 selected images from DOTA dataset which contain mostly commercial airplane objects and the number of them is 5209 and 90% of the images for training. The images were divided into two groups as the size of 1024 x 1024 patches to train Faster R-CNN and 608 x 608 for training SSD and YOLO-v3 detectors. They trained their networks on Nvidia Geforce GTX 1080 graphic card. Different training configurations were done by them according to the network and their hardware specifications. In the same study, transfer learning technique was applied by pre-trained parameters learned from Common Objects in Context (COCO) dataset [14]. And for the YOLO-v3 architecture training, Adam optimizer was proposed with learning rate 5 x 10−5 and decay factor 0.1 for each 3 epochs. The detection took about 97s with YOLO v3**,** 37s with SSD**,** 102s with the Faster R-CNN model. By comparing their models with their F1 scores, Faster R-CNN model gave the best result. SSD could not converge the training data well with low iterations. However, it performs better when localizing objects. Furthermore, [15] applied R-CNN [16], Fast R-CNN [17], Faster R-CNN, and YOLO [18] on NWPU VHR-10 dataset that contains totally 800 very-high-resolution (VHR) remote sensing images with 10 classes such as airplane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, and vehicle, for rapid target detection. They also built and trained their network on their own airport dataset that contains 1893 remote sensing images and airplane dataset containing 250 remote sensing images. The experiments on NWPU VHR-10 dataset also on their airport and airplane datasets gain from Google Earth demonstrate that YOLO model has a strong applicability for remote sensing image, especially in speed of prediction, however their YOLO has poor positioning accuracy and bad training approximation. Also using the same dataset as in [15], NWPU VHR-10 dataset, [19] proposed geospatial object detection using Faster R-CNN, SSD, Bag-of-Words (BoWs), and many other techniques with the main concern detection SSD model. The application was process under 64-bit Ubuntu 16.04 computer with CPU Intel(R) Core(TM) i7-6770K @4.00 GHz 8 and GeForce GTX1080 GPU with 8 GB memory CUDA8.0 cuDNN5.0, with end-to-end network trained by using the mini-batch stochastic gradient descent algorithm, where the momentum was fixed to 0.9 and the weight decay was set to 0.0005. The learning rate was initialized to 0.0005, with a step strategy

of g = 0.2 and the step size N = 25,000. The total iteration number of the proposed method was set 75000. As result, SSD produces bounding boxes with low IoU and confidence scores by multi-scale prediction. For some of their detection results, there is a small number of overlapped bounding boxes which had a large intersection with others.

In this thesis, we used different deep learning techniques with different hyperparameter settings to detect airplanes with very high accurate detection in very high-resolution satellite imagery. And also, we examined the results of each technique since it is promulgated that these techniques of deep learning with different hyperparameters lead to different performances in remote sensing object .

1.1 Purpose of Thesis

The main goal of this thesis is to detect airplanes using different deep learning techniques, with high speed of training and with fast and very accurate airplane detection from a variety of very high-resolution satellite images, also, compare those techniques in terms of their performances on very high-resolution satellite image airplane detection to solve some of occurred problems in some of prior object detection researches by analyzing different components of these deep learning techniques.

To reach this goal, we present a literature review in section 2, where we made a summary in a table of different literatures on object detection. Then in section 3, we introduce some deep learning techniques and our detection methodology. Next, we present the experiments of airplane detection in section 4 followed by experimental results in section 5. Finally, conclusions and future works are presented in section 6.

2. LITERATURE REVIEW

Many researchers have published many articles in remote sensing, especially in the domain of object detection. Here we review supervised and unsupervised methods, target recognition and object detection studies. Supervised and unsupervised methods are two main methods used as deep learning techniques for object recognition and detection.

2.1 Supervised Methods

In deep learning, we can consider these two supervised methods regarding target recognition, CNNs and Multilayer Perceptron (MLP). CNNs are models that transform the input image or an image patch into layers of feature maps. These feature maps represent high-level discriminative features of the original input data. For the MLP model, the input image or patch is reshaped into a vector. The final feature representation is obtained after transformation of each fully connected layer. These final features are sent to the classification layer to label the input image [20]. They transform the input image into vector, to make it ready for a one-class object detection by generating a two-dimensional vector space with that input image. This vector gives the predicted label for the input candidate to be the target or not, or gives the probability of the input being the target or not. The networks face a training process which takes as input a sample containing the target and a sample which does not contain the target.

Regarding the testing, the extracted proposals from a new image are processed by the models and attributed to it a probability. Then, the proposals which contain the target are selected by a given empirical threshold or other criteria. For images with a large number of objects and a large variance in the backgrounds, it is better for a good detection accuracy, a training with a CNN that extracts multiscale feature representations [2].

S. Xiang et al. [6] proposed a Hybrid Deep Neural Network (HDNN) where a division of the maps of the final convolutional and the max-pooling layer of the network into multiple blocks of variable receptive field sizes or max-pooling field sizes are done by HDNN to extract variable-scale features for detecting the objects in remote sensing images. Notice that the input to HDNN with convolutional layers is a gray image. The HDNN is a network where the feature maps of the last layer are divided into several T-blocks as *B*1, *B*2, and up to *BT*, with filter sizes of *s*1 x *s*1, *s*2 x *s*2, up to *sT*x*sT*, respectively. The *i*th block covers *i*th feature maps of the final convolutional layer. The activation propagation between the last two convolutions is defined such as

$$Bt = v\,(CL1 * ft + bt), \qquad \textbf{(2.1)}$$

with *Bt* as the *t*th block of the last feature maps, *ft is* the filters of the corresponding block, and *v* is the activation function.

The output of the HDNN is a two-node layer that indicates the probability of the input image patch containing or not the target. Once, we learn the multiscale feature representations for the final convolutional layer, an MLP network is used to classify the features. Some results of vehicle detection can be found in [6], to show that with a number of vehicles in the scene, the modified CNN network can successfully recognize the location of most of the targets with precision, also to indicate that the HDNN can learn fairly discriminating feature representations to recognize objects [2].

In [21], they use a CNN augmentation method, called pixel pair, for the training data to train a CNN appropriately with limited labeled data. For 3-Dimensional (3-D) CNN, convolution is made spatial-spectrally, and for 2-Dimensional (2-D) CNN, they perform convolution only spatially.

2.2 Unsupervised Methods

Unsupervised methods can learn feature representations with limited labels and they have less dependency for large labeled data. Unsupervised methods are typical considered with these methods: Restricted Boltzmann Machines (RBMs), sparse coding, AutoEncoders (AEs), k-means clustering, and the Gaussian mixture model. Some of these methods have been successfully applied to recognize remote sensing

scenes and targets [2]. From the input, the unsupervised methods learn features without having information about the correlated labels or without aids, or any supervision [2]. They are utilized to learn features from the object images [2]. As an example, in [7], the Deep Belief Network (DBN) built from stacking RBMs was used to recognize aircraft with different shapes and rotation angles from RS scene images and gives a good result.

In [22], they employ an unsupervised convolutional network to learn the spectral-spatial features with sparse learning to estimate the network parameters (weights). From P. Ghamisi et al. [23,24], we have a presentation of a fully residual conv-deconv network used for unsupervised spectral-spatial feature learning of HSIs.

Unsupervised methods is good for RS images since the methods can learn the features with unlabeled images and they can learn also the feature representations from the images or patches with no prior labels. RS labeled images are limited and supervised DL methods rely on a large number of labeled training samples.

A. Cheriyadat (2014) [5] used a sparse coding method to extract the dense low-level features and encoded in terms of the basic functions to generate a new sparse representation. Their methods can be stacked to form deep unsupervised models. Sparse coding is an unsupervised method for learning sets of over-complete basis vectors to represent an input vector as a linear combination of these vectors. Sparse codes of multiple features were used by G. Sheng et al. [25], which include color histogram descriptors, SIFT descriptors and local ternary pattern histogram Fourier descriptors to a two-stage SVM classifier.

We have another unsupervised method called Bag-Of-Visual-Words (BOVW) for feature learning for land-use scene recognition [26]. BOVW models consist of feature extraction and feature encoding. The algorithms extract local feature such as Scale-Invariant Feature Transform (SIFT) from the image and apply a quantization to the extracted features to a histogram by matching these features with a pre-constructed codebook learned through k-means algorithm. [27] utilizes sparse autoencoder to learn feature on multiscale saliency in an unsupervised manner. Both [5,27] also use the single-layer sparse coding. Contrary to them, Dai and Yang [28] made a two layer-

sparse coding scheme for satellite image classification by learning the saliency attention model.

In [26], they use a new unsupervised feature learning algorithm with multipath sparse coding together with multilayer architecture on Very High Resolution (VHR) remote sensing imagery. Their works were used for scene recognition in VHR of RS images within complex urban remote sensing sceneries. From them, unsupervised feature learning is performed on the visual data (RGB) and the non-visual data (Near Infrared (NIR) data). Unsupervised methods provide more information for distinguishing different remote sensing sceneries [26]. In their algorithm, simple building blocks of multipath sparse coding were done by dictionary learning and sparse features aggregation. They learn the dictionary D and compute a sparse code at each pixel using batch orthogonal matching pursuit. Then, make sparse code aggregation with a max-pooling. An example is taking an image patch P and divide it spatially into smaller cells. The max-pooled sparse codes represent the features of each spatial cell C. These max-pooled sparse codes are simply the component-wise maxima over all sparse codes in a cell [26]. From [26], the sparse codes from small image patches (input data) are aggregated into path level features in the first layer. The sparse codes from the first layer output are aggregated across the entire image to obtain image level features.

2.3 Target Recognition

Target recognition consists of extracting features from the object. It has three stages: detection, discrimination, and classification. In some papers, they used many CNNs techniques for target recognition, discussed below. For remote sensing, DL is mainly composed of supervised methods and unsupervised methods, where we can obtain low-level characteristics with a high frequency, such as outline, contours, edges of the objects, regardless of the color, size, rotation angle or shape of the targets [2]. From Figure 2.1 below, we have a general framework for target recognition.

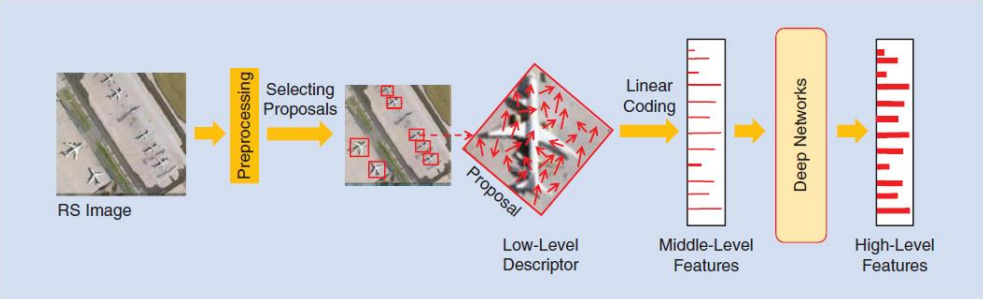

Figure 2.1: A framework used for target recognition using deep learning [2].

From this Figure 2.1, We can see that deep learning is used to learn the high-level features and then send them to classifiers for the classification. This classification can be done by supervised deep networks.

L. Zhang et al. [2] used CNNs for Synthetic-Aperture Radar (SAR) Automatic Target Recognition (ATR) with ATR dataset from Moving And Stationary Target Acquisition And Recognition (MSTAR) dataset [29] to recognize airplane objects and many other objects. Their problem was the lack of training sample compare to optical images. F. Xu et al. [30] employed a CNN without fully connected layers with MSTAR data and they achieved better accuracy about 99.1%. Many researchers [31,32] used CNNs to SAR ATR and used the MSTAR dataset for testing.

S. A. Wagner [33] presents a CNN to first get feature vectors and use SVM [34] for classification. J. Yang et al. [35] used DBN for SAR ATR, by extracting features and then fed them to a classifier. For object recognition, Diao et al. [8] use Deep Belief Network (DBN). At the top of the DBN, they use a supervised layer for fine-tuning, and they train their network with backpropagation (BP) that gives a sparse penalty constraint.

2.4 Object Detection

In remote sensing imagery, the object detection consists to identify the objects and find their locations with a bounding box depending on the detection algorithm. The idea is to get the target locations and attribute to these targets a specific class. For one of the object detection applications, they detected vehicles and non-vehicles from satellite images of IKONOS. They used a Neural Network (NN) with guided vector approach that learns the vehicle type and then integrates spectral and special features and makes

the classification using pixels, explained in [36]. In addition to that, [37] applied a detection model using DBNs and supervised learning. The framework takes labels, the weak ones, to identify an object in the entire image.

Object tracking is also a part of object detection which consists to estimate the location of the object in the image over time [38]. For tracking multiple objects from 2-Dimensional laser data, W. Diao et al. [8] applied Recurrent Neural Network (RNN) with an end-to-end technique to map raw sensor data to a hidden sensor space. [39] presented R-CNN-Based for ship detection. They have used four different models such as DPM, ZF-net based-Faster-RCNN, SSD300, and YOLOv2, where Faster-R-CNN based model achieved the best recall = 96.46 and precision=95.79, using 8 RS images for training and a validation data set including 2 images based on WGS-84 datum and UTM projection.

The below paragraphs are also resuming some object detection researches.

[40] presented a new deep learning technique called You Only Look Twice (YOLT) inspired by the 30-layer YOLO network to reduce model coarseness and to accurately detect dense objects such as buildings or cars. Their network architecture uses 22 layers and a down-sampling method by a factor of 16. They collected their training data from Planet satellites, DigitalGlobe satellites, and aerial Platforms with five categories such as airplanes, building footprints, boats, cars, and airports, including 13,303 labeled training data of a resolution of 15 cm ground sample distance (GSD) for car; 221,336 labeled buildings with a resolution of 30 cm GSD DigitalGlobe imagery and labeled building footprints over four cities: Las Vegas, Paris, Shanghai, and Khartoum; Airplane category has a total number of objects of 230; Boat data is a total of 556 boats; and Airport data included 37 Planet images for training purposes with a down-sampling by a factor of four. For their training, a stochastic gradient descent was used and with 5 boxes per grid, an initial learning rate of 0.001, a weight decay of 0.0005, and a momentum of 0.9. The training takes 2 to 3 days on a single NVIDIA Titan X GPU. For the network evaluation, 19 807 test cars, 73 778 test footprints, four airport test images for a total of 74 airplanes, and four boat images are labeled for 771 test boats were used. After evaluation of their network, it was noted poor results due to confusion between small and large features, such as runways and highways. The maximum F1 score obtained was 0.9.

In [41], they proposed a model called Rotation Dense Feature Pyramid Networks (R-DFPN) for ships detection in different scenes including ocean and port. Their framework consists of two parts such as a Dense Feature Pyramid Network (DFPN) for feature fusion and a Rotation Region Detection Network (RDN) for prediction. They used RPN to get rotational proposals for high-quality region proposals. At the end, the location regression and class prediction of proposals are processed using the Fast-RCNN. Their dataset was collected publicly from Google Earth with 8000 large scene images covering 400 square meters with RGB after geometric correction. The ratio of training set to test set was 1/4. All experiments were conducted on the deep learning framework, TensorFlow. A pretraining model ResNet-101 was used to initialize the network. They trained with a total of 80 k iterations, with a learning rate of 0.001 for the first 30 k iterations, 0.0001 for the next 30 k iterations, and 0.00001 for the remaining 20 k iterations. Weight decay and momentum were 0.0001 and 0.9, respectively. The chosen optimizer was Momentum Optimizer, and the IoU threshold of the positive sample was 0.5. After a series of experiments on remote sensing datasets, their method achieved 88.2% for Recall, 91.0% for Precision, and 89.6% for F1 score. Also, their experiments show that R-DFPN has state-of-the-art performance in ship detection in complex scenes, especially in the task of detecting densely arranged ships. However, more false alarms resulted.

[42] presented a Siamese-GAN method that learns invariant feature representations for both labeled and unlabeled images coming from two different domains. Their data set composed of trees, houses, bare soil, grass, roads, cars, water, solar panels, train tracks, from four cities such as Toronto, Trento, Vaihingen, and Potsdam with 224x224 pixels. The images collected from Vaihingen city in Germany were acquired using Leica ALS50 system at a spatial resolution of 9 cm. Each image is represented by three channels such near infrared (NIR), red (R), and green (G) channels. Those from city of Toronto in Canada were obtained by the Microsoft Vexcel's UltraCam-D camera and the Optech's airborne laser scanner (ALTM-ORION M) with a ground resolution of 15 cm and RGB spectral channels. And, the images of city of Potsdam were acquired using an airborne sensor consisting of RGB images with a ground resolution of 5 cm. Finally, the Trento dataset consists of UAV images acquired over the city of Trento in Italy. The images were captured using a Canon EOS 550D camera with an 18

megapixels CMOS APS-C sensor with a ground resolution of approximately 2 cm and RGB spectral channels. Their method was implemented in a Keras environment. For training the related subnetworks, the mini-batch size was fixed to 100 samples, the learning rate of the Adam optimization method to 0.0001, the exponential decay rates for the moment estimates and epsilon were 0.9 and 0.999 and 1e-8, respectively. In their first set of experiments, the regularization parameter of the reconstruction loss was fixed to 1. The experiments were performed on a MacBook Pro laptop (processor Intel Core i7 with a speed of 2.9 GHz, and 8 GB of memory). They have reached a total accuracy of 90.34%.

[43] proposed an aircraft type recognition framework based on conditional generative adversarial networks (GANs). Firstly, the method precisely detects aircraft's eight key-points, consisting of the tail, nose, right wingtip (RW), left wingtip (LW), and the four joints of the wings and fuselage. Secondly, a conditional GAN with a region of interest (ROI)-weighted loss function is trained on unlabeled aircraft images and their corresponding masks. Thirdly, an ROI feature extraction method is carefully designed to extract multi-scale features from the GAN in the regions of aircrafts. After that, a linear SVM classifier is adopted to classify each sample using their features. To train the model, dataset was collected publicly from Google Earth containing 562 large-scale optical remote sensing images at different airports around the world. And from these 562 images, they obtained 40,000 image crops resized to 256x256. They annotated each aircraft's eight key-points manually. The training dataset contained 30,000 crops and the validation dataset contained 10,000 crops. To evaluate the performance of their method, they created again another testing dataset. That testing dataset contained eight types of aircrafts, and each type had 1000 crops. For the aircraft key-point detection model, the network was trained from scratch using Adam with a mini-batch of 16, a learning rate starting from 0.001, and was divided by 10 when the loss plateaued. The network was trained for up to 30 epochs in total. For the GAN model, Adam was also used from scratch with a mini batch of 16, a learning rate of 0.0002 and the network was trained for up to 50 epochs on the training dataset. Both the key-point detection network and the GAN were built on PyTorch, and training and testing were on a NVIDIA K80 GPU with 24 GB of memory. For the SVM classifier, Principle Component Analysis (PCA) was first used to process the features extracted

from GAN. Then, a linear SVM with L2 regularization was adopted for classification. Both the PCA and SVM were trained and tested on an Intel Xeon CPU@2.40 GHz. Their method outperformed the end-to-end models by more than 5%.

[44] proposed YOLT2, an enhanced method, to detect object of multi-scale such as Boats in open water, Boats in harbor, Airplanes, Airports. Training data is collected from small chips of large images (DigitalGlobe + Planet) and applied augmentation method via random scaling of exposure, rotations, random scaling of saturation. And trained on one NVIDIA GTX Titan X GPU which took 3 to 4 days. They obtained an F1 score of 0.81.

In [45], they used 15 cm resolution images including 33000 unique labeled cars from 6 regions Toronto-Canada, Selwyn-New Zealand, Potsdam-Germany, Vaihingen-Germany, Columbus-Ohio, Utah-United States for car detection with deep learning, a YOLT model. They took the largest region (Utah) for testing, with 20000 labeled cars and 23 images. With their model, they reached An F1 score of 0.94. However, the F1 score was subject to the image resolution, for instance at images with sizes greater than 3.5 pixels, predicted number of cars was within 4% of ground truth.

Here in [46], a new method was proposed to detect circles belonging to oil tanks from satellite images. They firstly used a Faster R-CNN to extract the region of oil tank objects. Then, they applied an improved co-segmentation method with utilizing saliency co-fusion to segment bright oil tanks in regions. Finally, a circle detection method was used to detect circles considered as oil tanks. The dataset was downloaded from Google-Earth. For faster R-CNN training, 100 colored images with unified resolution of 1.3 m were used, with sizes varying from 810×1440 to 1080×1920 pixels. And for co-segmentation data, 30 different region of oil depots were used. The implementation and processing were performed in MATLAB. The results of their algorithm show F1 Score of 0.956.

[47] proposed Edge-Boxes and Convolutional Neural Network (CNN) for classifying target as aircraft or non-aircraft objects in a scene. Edge-Boxes method was used for the edge information to filter the set of target proposals. Then, a well-trained CNN was used to extract features of the proposed objects and classify them as aircraft or non-aircraft objects. The used dataset contained 500 aircraft patches, 5000 non-aircraft

patches and 26 test images obtained from Google Earth. For the CNN, the images were resized to 32×32. Their proposed system achieved an average precision and recall of 77.9% and 91.3% respectively. But for aircraft objects that coincide or overlap with other objects lower recall rates were obtained. Also, lower precision rates with high recall rates were observed in images containing a lot of objects like aircraft objects.

In [48], They developed a transferred deep model built on AlexNet CNN model to extract RSI features. A transferred deep model was used for better extracting domain-specific features of remote sensing images. Then, they integrated negative bootstrapping scheme into iterative detector training process to make the detector converge more stably and faster by selecting the most discriminative training samples. They evaluated the method on three different RSI datasets, which come from Google Earth containing airplanes, ISPRS (provided by the German Association of Photogrammetry and Remote Sensing (Cramer 2010)) containing vehicle, and Landsat-7 ETM+ containing airport. Google Earth dataset was separated into two parts, where 70 RSIs for training and 50 RSIs for testing. For ISPRS dataset, they randomly selected 60 RSIs for training and the remaining 40 RSIs for testing. For Landsat dataset, the training set included 123 RSIs and the testing set contained 57 RSIs. There were 50 additional negative RSIs in Google Earth dataset, 37 negative RSIs in Landsat dataset, and 24 negative RSIs in ISPRS dataset. The platform for feature extraction was MATLAB platform and run the Caffe toolkit on a PC with Windows 7 in CPU mode and Intel Core2 2.93 GHz CPU and 4 GB memory. For each image patch, the time consumed was about 1.1s. for their model result in terms of Average Precision (AP), they achieved 0.7626 on Google Earth/airplane, 0.4647 for ISPRS/vehicle, and 0.3365 for Landsat/airport.

In [49], They presented aircraft detection method which is a modified model of Fast-RCNN. Selective search method of Fast-RCNN was changed to a more efficient saliency method in their proposed system. They sampled 1200 images containing the aircraft from google earth of size between 300×300 pixels and 700 × 700 pixels. Their algorithm produced its best mean Average Precision (mAP) = 0.99 with 40000 iterations. However, their saliency algorithm has a poor effect when the boundaries are fuzzy between the background and the foreground, which leads to failure of subsequent object detection.

[50] proposed the detection method based on Faster R-CNN which they modified to accomplish vehicle detection. Firstly, in their proposed method in order to improve the recall, a hyper region proposal network (HRPN) employed to extract vehicle-like targets with a combination of hierarchical feature maps. Then, a cascade of boosted classifiers was used instead of the classifier after RPN to verify the candidate regions. They evaluated the method on the Munich vehicle dataset and the collected vehicle dataset from different sources. The Munich Vehicle data set was collected over the city of Munich in Germany, containing 20 aerial images captured by a DLR 3K camera system. And, the collected vehicle data set contained a total of 17 UAV images of resolution of 2 cm and 85 very high-spatial-resolution pansharpened color infrared (CIR) images acquired from a North-Western Polytechnical University (NWPU) VHR-10 data set. Experiments were implemented based on the deep learning framework Caffe and run on a PC with Intel core i7-4790 CPU, a NVIDIA GTX-960 GPU (NVIDIA, Santa Clara, CA, USA), (2 GB video memory), and 8 GB of memory. The operating system was Ubuntu 14.04 (Canonical, London, UK). The method got an AP of 83.9%. with this AP value, It can be observed that the proposed method can detect most of the vehicles successfully from the collected vehicle dataset but it performed poorly for large satellite images, and also their method cannot handle highly occluded vehicles, and the vehicles in thick shadows.

[51] presented an object detection pipeline to better recognize and localize target. In their model conditional probabilities are used to provide useful information regarding the location of the boundaries of the object inside the search region and allow the accurate inference of the object bounding box under a simple probabilistic framework. Their method is based on a convolutional neural network architecture that is properly adapted for this task, called LocNet. VOC2007+2012 dataset [52] was used for the training and testing. To recognize (detection) the targeted object, they employed Fast-RCNN or the Reduced-MR-CNN recognition models. They accomplished their best mAP with Reduced-MR-CNN, at IoU=0.5 they obtained mAP=0.784 and at IoU=0.7, the mAP=0.654.

[53] developed a model with 2D Gabor filter for stationary aircraft detection in satellite images obtained from Google Earth. In their system, 2D Gabor filter was used to determine features that emphasize the geometric structure of an aircraft. Then, the

aircraft detection was done using Support Vector Machines (SVM). Their images for their application were obtained from the well-known airports in Europe and the United States, with a total of 120 aircraft images and a total number of 450 non-aircraft images. All images were resized to 40x40 pixels for less time processing and memory requirements. As result, the detection rate was 0.91 and false alarm rate was 0.075.

[54] proposed a new detection framework based on spatial sparse coding bag-of-words (BOW) (SSCBOW) model. In the model, a sliding window was used to select a processing unit. Then, a new spatial mapping strategy was employed to encode the geometric information. Moreover, sparse coding was introduced for a much lower reconstruction error instead of K-means. Finally, linear support vector machine was for target detection. The model training set composed of 150 image patches containing an aircraft in the centre and 200 image patches of airfield background from Google Earth. The method was evaluated on 50 images collected publicly also from Google Earth. They obtained recall of 0.85 to 0.9 and precision 0.75 to 0.8. They had an increase in value when they had increased the number of training images.

[55] presented a multi-scale Faster R-CNN method based on deformable convolution for object detection with single/low graphics processing unit (GPU) systems. In the proposed system, Weight Standardization (WS) was used instead of batch normalization (BN) for a small batch size such as 1 image per GPU on single GPU systems. The proposed method used ResNet50 and deformable convolution for extracting high-resolution features. NWPU-VHR 10 dataset was used for the experiments, to which they applied augmentation method with blurring, rotating horizontally, rotating vertically, random image brightness, and gamma conversion operations. Experiments were conducted using the MM Detection toolkit on a desktop PC with Intel® Core ™ i5 2.4 GHz CPU, 6 GB RAM (Intel®, Santa Clara, CA, USA), single Geforce GTX 1080 graphics card (NVIDIA, Santa Clara, CA, USA) and Ubuntu 16.04 LTS (Canonical, London, United Kingdom). Their model achieved a 92.3 mAP, but with low detection accuracy for the bridge category.

[56] developed an object detection framework using a discriminatively trained mixture model. In the model, multi-scale histogram of oriented gradients (HOG) feature pyramids were constructed. Then, a mixture of multi-scale deformable part-based models was trained by training a latent Support Vector Machine (SVM). Multi-scale

HOG feature pyramid was firstly constructed for object detection stage. Then, computing and thresholding the response of the mixture model were used for the final object detection. Their proposed scheme was evaluated using two different types of RSI databases from Landsat-7 Enhanced Thematic Mapper Plus (ETM+) sensor and a high spatial resolution airplane imagery dataset from Google Earth. The dataset from Landsat-7 satellite composed of 65 shortwave-infrared (SWIR) images of 30-m-spatial-resolution and 31 panchromatic images of 15-m-spatial-resolution from China, including 60 SWIR images and 26 panchromatic images containing airport targets and the rest 10 negative images. 25 images containing airports were used for the training data and the remaining 71 images for testing, with 56 labeled airports as positive samples and randomly selected 200 image patches as negative samples for the airport model training, 125 airports from test images were labeled and used as ground truth. The second database consisted of 71 images in which 61 images with airplane targets. They selected randomly 18 images with 160 labeled airplanes as positive samples for training and the rest 53 images for testing. Moreover, 366 airplanes from 53 test images were labeled as ground truth. The framework was implemented on a 24-core Lenovo Server with 2.8 GHz Intel Xeon CPU, 64 GBRAM and LINUX operation system using MATLAB R2010b. The running time for an image with the size of 1000x800 was about 6 s. And with their framework they reached an AP of 0.9092 for airport and an AP of 0.8997 for airplane category. However, the developed method had problem detecting similar structure and shape such as straight roads and coastlines for airport, passenger terminals for airplane, in image with low contrast between the background and the target.

[57] proposed a novel practical framework where a computational saliency prediction model was built. And, the output of this model was used to predict a small set of target candidate areas. Afterwards, discriminative sparse coding was used to simultaneously localize multiple targets from multiple classes. Finally, the trained multi-class object detector was only applied to the target candidate areas for classification into various categories of target. They collected 400 high resolution RSIs of size 1000x800, from Google Earth for evaluations, which contained airplanes, storage tanks, ships, and baseball diamonds, with spatial resolution ranging from 0.5 m per pixel to 2 m per pixel. They used 150 images for testing saliency models, 43 images were used for

training the multi-class target detector, and 207 images for testing the performance of the target detection. From the 43 training images, they labeled 21 objects from each class and then applied data augmentation method to obtain 378 ships, 756 airplanes, 756 baseball diamonds, and 21 storage tanks for the whole training. The testing set contained 756 airplane targets, 1876 storage tank targets, 295 ship targets, and 106 baseball diamond target objects. Their proposed algorithm achieved an AP of 0.7680.

A summary of some researches regarding object detection is presented in Table 2.1 below.

Table 2.1: A summary of some researches regarding object detection.

Dataset	Method	Target	Performance	Reference
UC Merced And Sydney City	Rcnet, SPMK,SSC,SSAE	Multiple	Accuracy = 98.78%	[2]
City Of San Francisco Google Earth (63 Images)	HDNN	Vehicle	Recall Rate=91.6%	[6]
Google Earth (51 Images)	DBN	Aircraft	Recall Rate=54.9%	[7]
DOTA And Pleiades 1A&1B	SSD, Faster R-CNN, YOLO-V3	Airplane	F1 = 0.94	[12]
NWPU VHR-10 & Google Earth (2143 Images)	R-CNN, Fast R-CNN, Faster R-CNN, YOLO	Multiple	Recall=92.66%, TF=16, FP=29,TP=206,All=218	[15]
NWPU VHR-10	SSD	Multiple	AP=0.958	[19]
Google Earth (120 Images) ISPRS (100 Images) Landsat-7 (180 Images)	WSL Based Object Detector	Airplane, Vehicle, Airport	AP=0.4676	[37]
WGS-84 Datum And UTM Projection (10 Images)	DPM, ZF-Net Based-Faster-RCNN, SSD300, And Yolov2.	Ship	From Faster-RCNN: Recall = 96.46 Precision = 95.79	[39]
Planet Satellites, Digitalglobe Satellites, And Aerial Platforms	YOLT	Multiple	F1=0.9	[40]
Google Earth-8000 Large Scene Images	R-DFPN- Fast-RCNN-Resnet-101	Ship	Recall=88.2%, Precision=91.0%, F1= 89.6%	[41]
Toronto, Trento, Vaihingen, And Potsdam Cities Images From Airborne Sensor And UAV	Siamese-GAN	Multiple	Accuracy Of 90.34%.	[42]
Google Earth-562 Large-Scale-(48000 Objects)	GANs	Aircraft	+5% To End-To-End Models	[43]
Digitalglobe + Planet	YOLT2	Multiple	F1 = 0.81	[44]
53000 Labeled Cars	YOLT	Cars	F1=0.94	[45]
Google-Earth (130 Colored Images)	Faster R-CNN	Oil Tank	F1=0.956	[46]
Google Earth (500Aircraft, 5000 Non Aircraft, 26 Test Images)	EdgeBoxes-CNN	Aircraft	Average Precision=77.9% And Recall =91.3%	[47]
Google Earth (170), ISPRS (137), Landsat-7 ETM+(204)	Alexnet-CNN	Airplane, Vehicle, Airport	AP=0.7626	[48]
Google Earth (1200 Images)	Fast-RCNN	Aircraft	mAP= 0.99	[49]
Munich Vehicle Dataset(20 Aerial Images), NWPU VHR-10(85 Images),UAV(17 Images)	Faster R-CNN	Vehicle	AP=83.9%.	[50]
VOC2007+2012	Locnet, ,Fast-RCNN, Reduced-MR-CNN	Multiple	mAP=0.784	[51]
Google Earth(120 Aircraft Images-450 Non-Aircraft)	2D Gabor Filter- SVM	Aircraft	Detection Rate =0.91 And False Alarm Rate = 0.075	[53]
Google Earth (150 Image-200 Image Patches Of Airfield-50 Images)	SSCBOW	Aircraft	Recall = 0.9 And Precision = 0.8	[54]
NWPU-VHR 10 Dataset	Faster R-CNN- Resnet50	Multiple	mAP = 92.3%	[55]
Landsat-7- ETM+ (96 Images), Google Earth (71 Images)	HOG- SVM	Airport And Airplane	AP Of 0.9092 For Airport And An AP Of 0.8997 For Airplane	[56]
Google Earth (400 Images)	Discriminative Sparse Coding	Multiple	AP=0.7680	[57]

From Table 2.1, we can find the resume of our literature review on object detection. All of these papers were discussed within the introductory and the literature review sections. Multiple is for detection of multiple classes, at least 4 targets. FP for False Positive and TP is the True Positive. AP is the Average Precision. mAP stands for mean Average Precision. Many studies were done in object detection, especially for airplane target detection, using several different deep learning techniques, where they achieved high performances.

3. DEEP LEARNING AND USED DETECTION METHODOLOGY

We discuss here some deep learning algorithms and the methodology that we used in our airplane detection. In general, deep learning has three main components which are input data, the core deep networks and the output data.

3.1 Deep Learning Algorithms Overview

Various deep learning techniques currently exist in the literature. The need will decide which technique to use. In this sub-section, we give an overview of some DL algorithms such as Convolutional Neural Networks (CNNs), ResNet, Deep Neural Networks (DNNs), and Deep transfer learning model mechanisms, respectively.

3.1.1 Convolutional neural networks (CNNs)

Convolutional neural network (CNN) is based on image and composed of trainable multilayer architecture. It is composed of convolutional layer, nonlinearity layer, and pooling layer. Each of these layers has a specific task. One of feature of CNN is that it resists to noise, rotation, scaling, and different light conditions. CNN can be used with convolutional layers to perform image filtering, denoising, enhancement, and detection. CNN is the leading model in deep learning with its excellent aspects found in image processing and classification. Regarding some models [58] of CNNs, we can enumerate GoogLeNet, Visual Geometry Group (VGG) networks, Fully Convolutional Network.

VGG-VD-16: composed of 13 convolutional layers followed by 3 fully connected layers.

GoogLeNet has more than 50 convolutional layers distributed inside the inception modules with one fully connected layer at the end. In [59], in an experiment, a state-of-the-art output was got from the GoogLeNet architecture to detect mediastinal lymph nodes. In [60], they used the CNN model to detect target for the robot navigation, by

using small images as input to the system with GoogLeNet. They used GoogLeNet for its deepest layer and expected a good performance from it.

Training CNN (Figure 3.1) from scratch or full training is a challenge because it needs large training labeled data, large computational and memory resources. It can get to overfitting and convergence problems. In CNN regularization, it is important that the network is augmented to avoid overfitting. CNN is used for supervised learning.

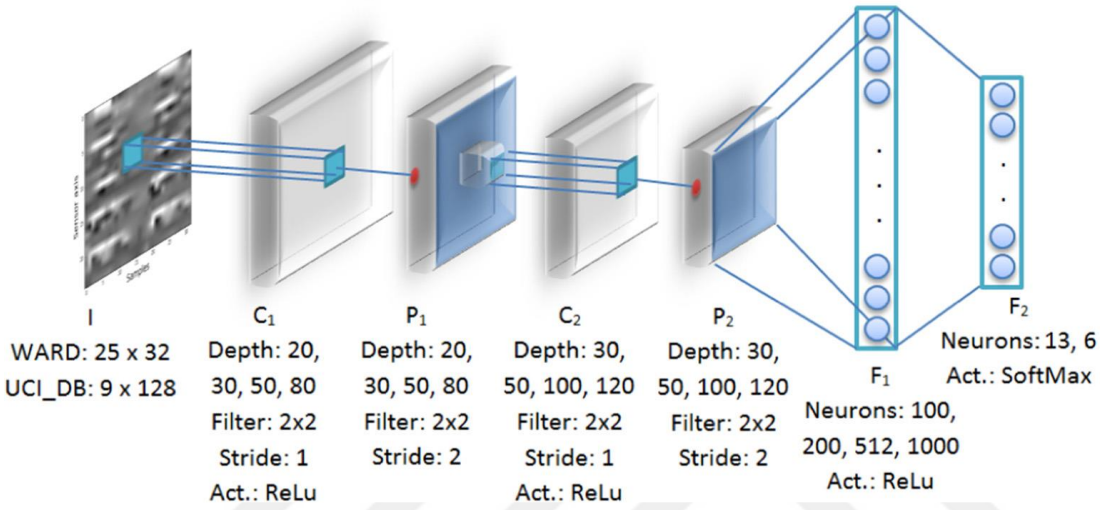

Figure 3.1: An example of Convolutional Neural Network (CNN) architecture [61].

Figure 3.1 shows the depth of the layers, the neuron number and the type of activation function, filter size, stride length. The model is with two convolutional layers (C_1 and C_2), two pooling layers (P_1 and P_2) and two fully connected layers (F_1 and F_2) in this Figure 3.1. WARD and UCI_DB are the two input datasets into the network with image size of 25x32 and 9x128, respectively. In the network, from the first convolution layer (C_1) to the first fully connected layer (F_1), ReLu activation function was used and SoftMax activaton function for the last fully connected layer (F_2). The activation functions are explained in nonlinearity layer subsection. The depth numbers show how deep is each convolution and pooling layer. The filters refer to the kernels. The stride represents the number of pixels shifts over the input matrix. That is when the stride is 1 the filters are moved to 1 pixel at a time, and for 2 the filters are moved to 2 pixels, and so on. The number of neurons in F_2 is the number of object categories. Each neuron in this layer represents an object class, also called output neuron. The above mentioned elements (depth of the layers, the neuron number and

the type of activation function, filter size, and stride length) from Figure 3.1 are the spatial feature learning hyperparameters of CNN.

In CNN, the number of volumes can determine the depth of the layer. Taking i, j^{th} hidden neuron on the l^{th} layer's, v^{th} volume, the output of CNN is defined as

$$a_{i,j}^{l,v} = \sum_{pv}^{V} \sigma(\sum_{n}^{N} \sum_{m}^{M} (w_{n,m}^{l,v} a_{i+n,j+m}^{l-1,pv} + b^{l,v})), \quad \textbf{(3.1)}$$

with filter size $N \times M$, V the number of volumes on the preceding layer, b the bias, w the weights and $\sigma(x)$ the activation function taking x as input. From researcher experiments, CNN with Rectified Linear Unit (ReLU) gives better results as activation function. Tanh and sigmoid are also used.

$$\text{ReLU (x)} = \max (0, w^T x + b), \quad \textbf{(3.2)}$$

with w the weight, 0 for all zero vector, b bias vector. Another activation function is the P-norm (Y) as defined below.

$$\text{Y} = \|x\|_p = (\sum_i |x_i|^p)^{1/p}. \quad \textbf{(3.3)}$$

At the end of the network, the volume of the last convolution or pooling layer produces the input of multi-layer with a feed forward network which gives the result.

To determine the quality and the performance of the system a loss function is calculated. First, we calculate the loss depending on the loss the gradient is calculated. And all the parameters are calculated based on the gradient [62]. There are many types of loss function such as:

- The quadratic cost

$$\text{Qc} = \sum_j^M (y_j - a_j^L)^2 + \frac{\gamma}{2N} \sum_w w^2 \quad \textbf{(3.4)}$$

- Cross-entropy

$$\text{Ce} = \sum_j^M (y_j ln a_j^L + (1 - y_j) \ln (1 - a_j^L)) + \frac{\gamma}{2N} \sum_w w^2; \quad \textbf{(3.5)}$$

- Log-likelihood

$$\text{LL} = ln a_y^L + \frac{\gamma}{2N} \sum_w w^2; \quad \textbf{(3.6)}$$

$w^l \in N\ (0, \frac{1}{\sqrt{k}})$, $\alpha = a_0 e^{-\varphi \varepsilon}$, N is the number of sample and M number of output neurons. y_j the actual output; a_j is the activation of the output neuron; γ is regularization strength; w is the weights; φ is the learning rate decay factor; a_0 is the initial learning rate; h the number of hidden neurons. The sum of squared weight matrices term in all three loss functions is the L2 regularization term. The system reshapes the input to 2-D space $K \times K$ for convolution and pooling where K is the size of the input matrix. We take the convolution layers, each of them, with 3-D convolutions with $W \times W \times B$. The kernel size of the convolution is $(W \times W)$ and B the number of kernels in each convolution. We get the neuron value as

$$x_j^I = g\ (b_j^I + \sum_{i=1}^M x_i^{I-1} * k_{ij}^I), \quad \textbf{(3.7)}$$

at the j-th feature map in the I-th layer. With M as the input feature maps number, b_j^I taken as the bias, k_{ij}^I as the weight which is connected to j-th feature map, $g\ (\cdot)$ as the function for activation, and $(*)$ the operator for convolution.

The convolution gives many features, so the pooling is used to reduce the number of parameters for the training. After two convolutional and pooling layers, we get the deep spectral features from pixel spectral.

The probability

$$\text{P}\ (y = c) = \frac{e^{\theta_c^T F}}{\sum_{j=1}^C e^{\theta_j^T F}}, \quad \textbf{(3.8)}$$

shows that the input is an element of the class *c*, with $\theta_1, \theta_2, \theta_3, \ldots, \theta_c$ takes as model parameters. *y*=1,2, 3, *c*, ..., *C* as the ground truth with *c* the target, and *F* the feature vector [63].

All the number of the convolutional filters, pooling sizes, filter sizes should be learned from the output of the preceding layer. Basic feature learns from lower layers and so on. The size of the filters or feature maps (multiple maps of neurons) is equal to the dimension of the input image.

The set of weights is called filter bank. Filtering operation done with a feature map is called discrete convolution. The training hyperparameters of CNN are the learning rate, momentum, learning rate decay, early stopping patience, maximum number of epoch, weight decay, and dropout rate. The learning rate determines how to adjust the weights of the network regarding the loss gradient. The learning rate decay is used to control the learning rate at a defined time or step. The momentum is used to speed up the training in general. it is also used to accumulate the gradient from the past steps to decide the next direction. Dropout rate is a regularization technique. Specifically, It is used to impide overfitting.

3.1.1.1 Convolutional layer

The convolution, also called pattern matching, scans the image with a given pattern and calculates the strength of the match for all positions. Convolution, as shown in Figure 3.2 below, serves to convolve element-wise products of a kernel with an image. In the same figure below, its input has three channels as a matrix of size 3x3, and its output has 2 channels as a matrix of size 2x2. The channel presents a view of the same image. For example, we can use three channels (Red (R), Green (G) and Blue (B)) to describe an RGB image. Each of them with the same size.

A kernel (Figure 3.3) is a matrix with a total number equalling to the multiplication of the input and the output channel numbers. We have all kernel parameters in this matrix.

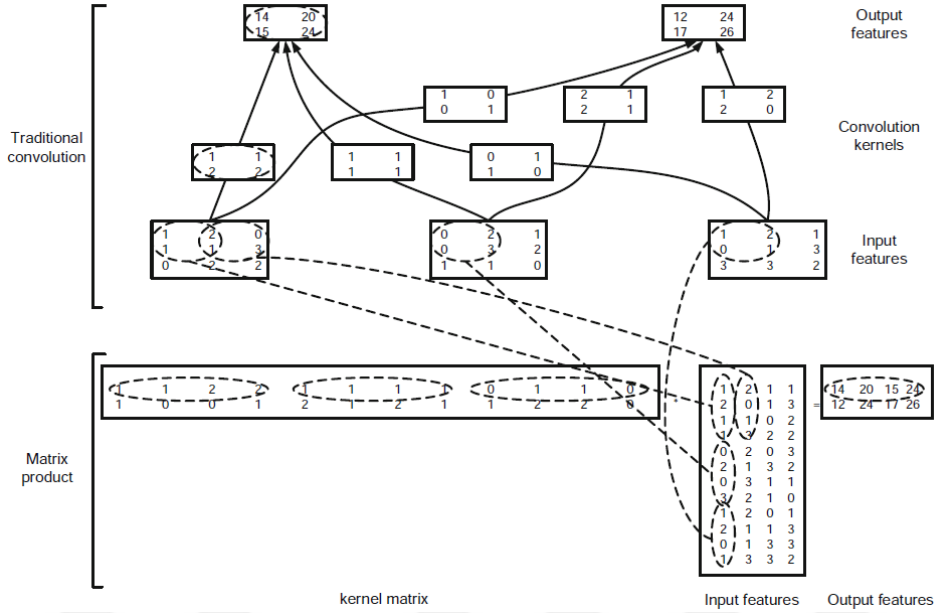

Figure 3.2: An example of convolution [64].

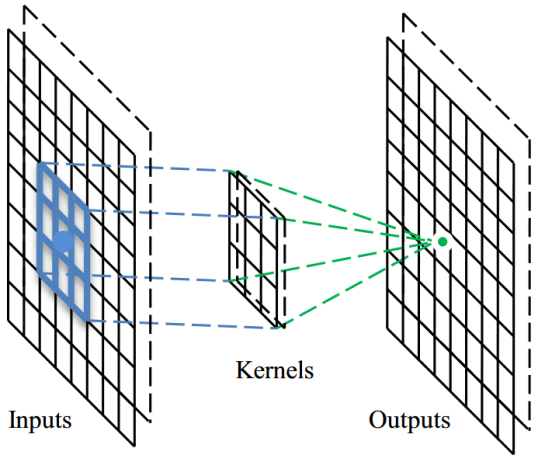

Figure 3.3: Kernel maps [62].

From Figure 3.2, we have, on top, operations of the original convolution, and down, the same representation as top but with product of large matrix.

In convolution step, we have shared of weight in the same feature map and which reduce the parameter number, correlation learning among neighboring pixels with local connectivity [62].

Convolution calculates the output feature map as follows:

$$z^s = \sum_{i=1}^{q} w_i^s * x^i + b_s, \quad \textbf{(3.9)}$$

With $(*)$ as the convolution operator; *w* as the filter bank: trainable filters; *b* as trainable bias; *x* as input.

3.1.1.2 Nonlinearity layer

It consists to apply a nonlinearity function to each component of the feature map got from the convolutional layer as defined below:

$$a^s = f(z^s), \quad \textbf{(3.10)}$$

$f(\cdot)$ can be ReLU or another activation function (nonlinearity function). z^s is the feature map and a^s the output of the activation function. The typical activation function in CNN is ReLU.

Activation function is used to decides, whether a neuron should be activated or not by calculating weighted sum and adding bias with it. The activation function is served to introduce non-linearity into the output of a neuron. In simple words, it is used to limit the output signal to a finite value. So activation functions are for restricting the output value to a determined finite value. There are many types of activation function, some of them enumerated below:

-Sigmoid function:

$$\text{Sig}(x) = \frac{1}{1+e^{-x}} \quad \textbf{(3.11)}$$

Sig (*x*) is the most used activation function with an output range [0, 1] which gives sparse and asymmetric activation values. The values can be close to 0 but not 0.

-Rectified Linear Unit (ReLU):

$$\text{ReLU}(x) = \max(0, x) \quad \textbf{(3.12)}$$

It is a simple gradient and it enforces the activation of sparse. Its output values can be 0. ReLU function gives 0 as output for a negative input. If the imput value is positive the function is returning the same input as its output.

-Hyperbolic tangent function:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \textbf{(3.13)}$$

It has an output range of [-1, +1] which gives a symmetric activation value.

3.1.1.3 Pooling layer

Pooling (or sub-sampling) layer follows convolutional layer. It reduces the dimensionality [65]. Pooling merges semantically similar feature into one [2,66]. Pooling (or aggregation) determines the presence of a pattern in a region. Pooling is used to keep the most relevant features and eliminate the redundant ones from the feature maps obtained from the convolutional layer [67].

We have many types of pooling explained as follows:

-Max-pooling (Figure 3.4): it applies the maximum pooling operation to the input in a window matrix regarding each channel. The most known pooling is 2x2 max-pooling. It helps to select the maximum activation in a region of 2x2 size on the convolutional layer. It consists to apply a max operator to the activation in a reduced spatial boundary of the feature map.

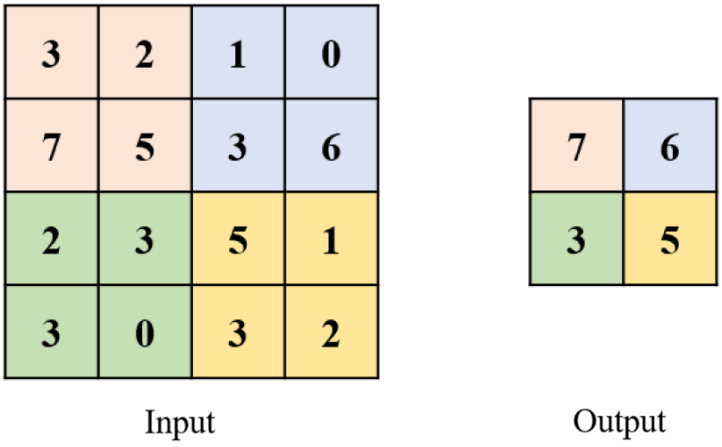

Figure 3.4: An example of max pooling [38].

In Figure 3.4, they use the max pooling layer with 2x2 pooling mask, where it took the first output as the max of {3,2,7,5} which is 7. The same for others. This method is used to reduce the data.

The max pooling is sensitive to overfit the training data, which is an obstacle for a well generalization of test samples [68]. To solve this problem, [3,58,59] used a stochastic pooling. Wu Zhihuan et al. [15], proposed a spatial pyramid pooling to remove some limitation in the CNN model by changing the last pooling to spatial pyramid pooling layer in airplane and airport detection. The deformation is controlled more efficiently by def-pooling introduced by Ouyang et al., where they made a constrained pooling of formation to enrich the deep model.

- Min pooling: it selects the minimum pixel value of the batch.

-Average pooling: it applies the average pooling operation to the input in a window matrix regarding each channel.

3.1.1.4 CNN in some applications

CNN was also used for person re-identification and gave good result [65]. For multi-classification tasks, for feature extraction and distance metric learning, they calculated the loss in two ways using two loss functions. They used some CNN frameworks such as ImageNet and ResNet-50. They used a triplet loss function.

$$L_{trp}=[\ \text{thre}\ +\ \text{d}\ (F_w(x_a), F_w(x_p))\ -\ \text{d}\ (F_w(x_a), F_w(x_n))\ ]_+ \qquad \textbf{(3.14)}$$

L_{trp}: First lost function,

where $[\ x\ ]_+$=max (x,0); thre is threshold greater than 0; d (x, y) is the relative distance between x input and output y.

d (x, y) = $\|x - y\|_2$ is metric function as Euclidean distance. $x_a, x_p,\ x_n$ are triplet units.

CNN is composed of feature extractor and multilayer perceptron (MLP) for assigning class labels or calculating the probabilities. In CNN classification application, the probability is defined for an element M to be an element of the class j as

$$P_j = \frac{e^{\frac{(v_j^T M)}{\vartheta}}}{\sum_{l=1}^{L} e^{\frac{(v_l^T M)}{\vartheta}} + \sum_{k=1}^{Q} e^{\frac{(\mu_k^T M)}{\vartheta}}}, \quad \textbf{(3.15)}$$

with v_j^T transposition of *j*-th column of a lookup table (LUT); μ_k^T is the transposition of *k*-th identity of the circular queue. With Softer probability distribution parameter ϑ . *L* to be the number of columns for LUT and *Q* the size of the queue.

Also, the probability that the object belongs to *j*-th of the circular queue is

$$R_j = \frac{e^{\frac{(\mu_j^T M)}{\vartheta}}}{\sum_{l=1}^{L} e^{\frac{(v_l^T M)}{\vartheta}} + \sum_{k=1}^{Q} e^{\frac{(\mu_k^T M)}{\vartheta}}}. \quad \textbf{(3.16)}$$

The second loss function is

$$L_{alm} = E_M [\log p_n]. \quad \textbf{(3.17)}$$

Then, we sum all of loss functions and get

$$L = L_{trp} + \rho L_{alm}, \quad \textbf{(3.18)}$$

ρ is a trade-off parameter to balance the two types of loss function. In [65], it was set to 2. In the same application, they used ADAM optimizer and calculated learning rate for different parameters, with GPU, and 971 images of size 256x128. They used epoch as 150, weigh decay 0.0005, batch size with a value of 180, and ϑ was 0.03. The momentum was 0.5 and their training took about 10-12h. They formulated the learning rate as

$$\text{Lr} = \beta 0.1^{epoch//40}, \quad \textbf{(3.19)}$$

where β the initial learning rate set to 0.0002.

The stochastic gradient descent (SGD) is mostly used in CNNs, however finding a good learning rate and converging to local minimum is a problem.

Sometime in the application of CNN, a new term is added to the function to avoid overfitting in network training and testing. This term is called regularization term.

$$Cost\ function = Loss + Regularization\ term. \quad \textbf{(3.20)}$$

Regularization technique makes slight modifications to the learning algorithm for better generalization of the model. It has the main function to penalize the coefficients (weight matrices). Once the regularization term is added, the values of weight matrices decrease by assuming that a neural network with smaller weight matrices leads to simpler models. Thus, the overfitting is reduced. There are two main types of regularization term, L1 and L2 the most common types of regularization.

In L2, we have:

$$Cost\ function = Loss + \frac{\varphi}{2m} * \sum \|w\|^2 \quad \textbf{(3.21)}$$

φ is the regularization hyperparameter. A vey high value of φ will lead to underfitting. So, its value is optimized for better results. m is the number of samples.

In L1, we have:

$$Cost\ function = Loss + \frac{\varphi}{2m} * \sum \|w\| \quad \textbf{(3.22)}$$

L1 compresses the model.

In [65], they used three convolutional networks with different inputs to extract features at the same time. CNN was used successfully by (LeCun, Bottou, Bengio, and Haffner, 1998), as mentioned in [61], on Modified National Institute of Standards and Technology (MNIST) database for digit classification.

CNN was used by many researchers and in many applications discussed in [38]. They used CNN in biology, specifically in training DNA sequence [66]. [69] shows the work and appreciation of CNNs in medical. They used CNN for feature extraction from R-

CNN and SVM for the classification in [15]. They transferred the region extraction in feature map and used Faster-RCNN model to diminish the repetitive calculation. Faster-RCNN is a model which uses region proposal network (RPN). In the same experience the You Only Look Once (YOLO) model was used to make result comparisons. This YOLO model they used, is a 24 convolutional layers model with two fully connected layers.

3.1.2 ResNet

Very deep neural networks are difficult to train and they were producing higher training errors, thus test error. [70] presented a residual learning framework, called residual network (ResNet) to ease the training of networks that are deeper compared to those used previously. Their framework addressed the degradation problem. In the system, the layers were explicitly reformulated as learning residual functions with reference to the layer inputs. This new framework makes the training possible up to hundreds or even more of layers with lower training errors compared to prior deeper neural networks.

In ResNet, an identity shortcut connection is introduced to skip one or more layers (Figure 3.5).

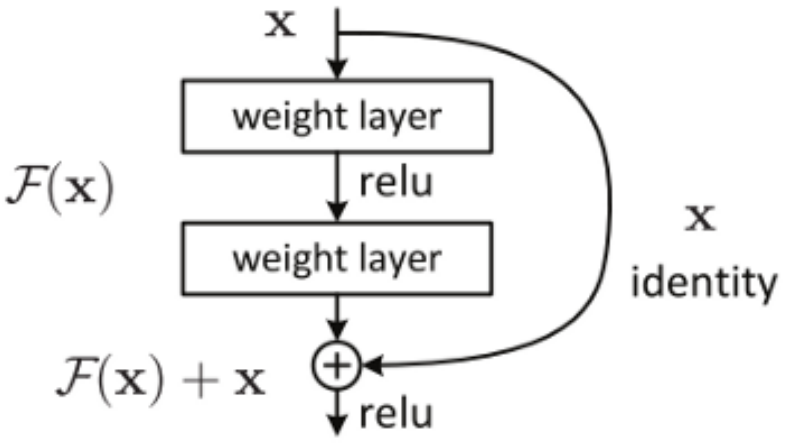

Figure 3.5: A residual bloc [70].

Each few stacked layers fit a residual mapping instead of a desired underlying mapping. They considered H(x) as the desired underlying mapping with x the input to the first layers, and the stacked non linear layers fit another mapping giving

$$F(x) = H(x) - x \quad \textbf{(3.23)}$$

Then, x is added to both side of equ. 3.23, that recasts the original mapping into F(x) + x.

This new formulation of F(x) + x can be done by feedforward neural networks with shortcut connections (Figure 3.5). In ResNet, the shortcut connections are for performing identity mapping, and their outputs are added to the outputs of the stacked layers.

In the Figure 3.5, we have building block of ResNet defined as

$$y = F(x,\{Wi\}) + x, \quad \textbf{(3.24)}$$

With y output vectors of the layers considered. And, The function F(x,{Wi}) is the residual mapping to be learned, W is the weight with i the layer number. For a two layers, F= $W_2\sigma(W_1 x)$ where σ is ReLU activation function. if the x and F have different dimensions a linear projection coefficient (W_s) is used to match their dimensions. That is:

$$y = F(x,\{W_i\}) + W_s x \quad \textbf{(3.25)}$$

For a single layer, F will be similar to a linear layer such as

$$y = W_1 x + x. \quad \textbf{(3.26)}$$

ResNet architecture (Right) is showing in the Figure 3.6 with 34 layers compared to two deeper neural networks, VGG-19 (left) and 34- layer Plain (middle) networks.

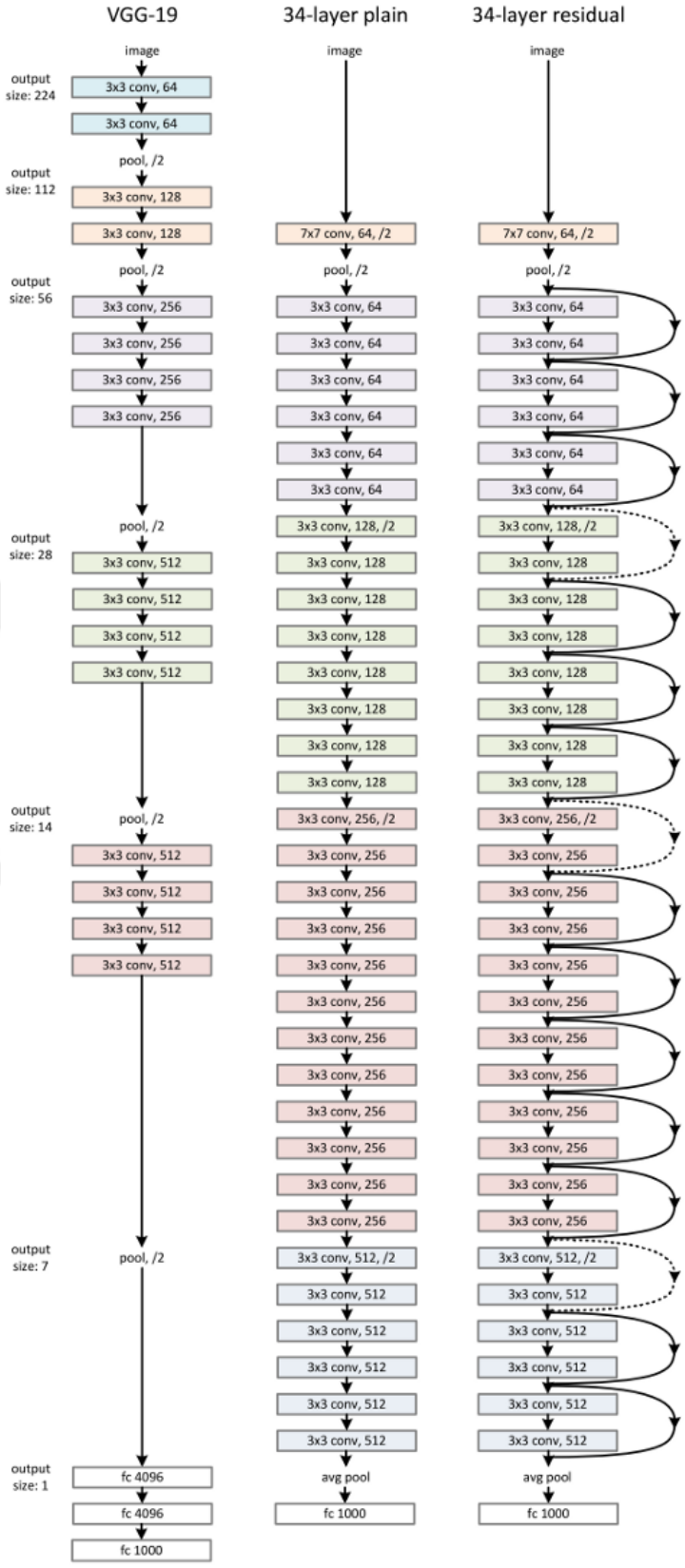

Figure 3.6: VGG-19 architecture (left), Plain architecture (middle), and ResNet architecture (right) [70].

Plain Network (middle) is inspired by VGG nets (left). The convolutional layers have 3×3 filters, with the same number of filters for the size of the same output feature, and layers with equal filter number; and for halved feature map size, the number of filters is doubled to keep the same time complexity per layer, and downsampling is performed by convolutional layers with a stride of 2. This network with a total number of weighted layers of 34 is ending with a global average pooling layer attached to a 1000-way fully-connected layer and a softmax.

Residual Network (ResNet) is also based on the Plain network with the same structure, however shortcut connections are inserted to Plaine network to form ResNet, which makes the difference between Plain and ResNet networks. ResNet was trained and evaluated on ImageNet dataset [71] and it reduced the top-1 error by 3.5%. And it won the 1st place in ILSVRC 2015. Also, they obtained a 28% improvement on the COCO object detection dataset.

3.1.3 Deep neural networks (DNNs)

Deep Neural Network (DNN) is an advanced method in machine learning. The difference between DNN and Neural Network (NN) is that DNN contains more hidden layers mostly more than two hidden layers. Figure 3.7, below, is an example of deep neural network. DNNs are used in many applications like RS applications, speech recognition, etc. In this section, we will go through DNNs architecture.

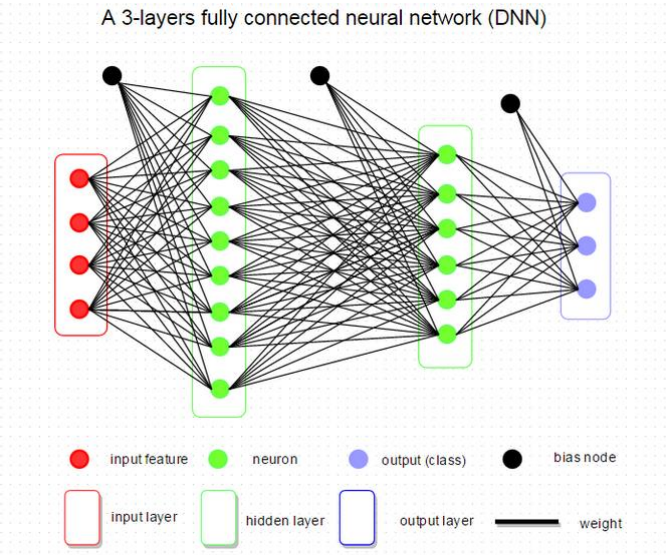

Figure 3.7: A sample Deep Neural Network (DNN) model [72].

From Figure 3.7, we can see a DNN with four layers including the input layer in red, two hidden layers in green and the output layer in blue.

3.1.4.1 Building a DNN

To construct the mathematical model of the deep neural network, we take the input layer as 0 and the output layer as *L* and the hidden layers are between these two values such as $0< l <L$.

The general model is composed of an activation vector, an excitation vector, a weight matrix, and a bias vector defined below:

Let's take *L* layers, for $0 < l < L$, we have:

W^l: Weight matrix at the *l*-th layer;

b^l: Bias vector at the *l*-th layer;

ex^l: excitation vector at the *l*-th layer;

a^l: activation vector at the *l*-th layer;

$$ex^l = W^l a^{l-1} + b^l, \tag{3.27}$$

$$a^l = f(ex^l) = f(W^l a^{l-1} + b^l), \tag{3.28}$$

where $b^l,\ ex^l\ and\ a^l \in \mathbb{R}^{N_l \times 1};\ W^l \in \mathbb{R}^{N_l \times N_{l-1}};\ N_l$: the number of neurons at layer *l* and *f(.)* is an activation function which has input and output domain $\mathbb{R}^{N_l \times 1}$.

For *l*=0, $a^0 = \mathbf{o} \in \mathbb{R}^{N_0 \times 1}$ called the observation or feature vector with $N_0 = D$ as the dimension of the feature. **o** is the input.

The output layer for a regression is:

$$a^L = W^L a^{L-1} + b^L. \tag{3.29}$$

Each neuron in the output layer represents a class. For a system with C classes (C = N_L= number of neurons in the output layer = number of classes), let's take $a_i^L \in a^L$ as

the value for the class *i* with *i* = 1, 2, . . ., C. This value is considered as the probability which measures the chance for the **o** vector to be an element of the class *i*.

The probability values are between 0 and 1 or their sum is equal to 1. To be in this interval [0,1], we normalize the excitation by using SoftMax function.

$$a_i^L = \text{softmax}_i\ (ex^L) = \frac{e^{ex_i^L}}{\sum_{j=1}^{c} e^{e_j^L}} \quad \textbf{(3.30)}$$

is the normalized value of a_i^L where $ex_i^L\ \epsilon\ ex^L$.

To make a forward computation with an **o** vector to get the output value with the parameters *W* and *b*, we just need to calculate the activation vector through all hidden layers from layer 1 to the layer *L*-1 and apply the regression function to the result. And to classify the output, we use the normalize function defined to get a_i^L with Softmax function. And, a backpropagation computation is used to train the model over several ilterations depending on the application.

3.1.5 Deep transfer learning model mechanisms

The Transfer Learning (TL) consists to use a trained model to another training. Transfer learning is used to transfer the knowledge between tasks. With TL, we can train with small dataset and have a good result. Transfer learning has better performance when a huge data is to be processed like in speech recognition, in [73], etc. In the same paper, [73], they used pre-trained CNN ResNet34 architecture. ResNet34 architecture is a pre-trained model with about 1000 categories trained on ImageNet database. There are other pre-training models such as inception and VGG. The DL can use a pre-trained model to perform well. They used TL in [73], to classify brain as normal or abnormal.

-Optimal learning rate finder: It helps to improve the model performance. It helps to make decision on among of the updated model parameters by taking account the gradient. It makes also adjustment depending on the learning rate value, small or high**;**

-Stochastic Gradient Descent With Restarts (SGDR): This decreases the learning rate while training is under process;

-Fine-tuning: It adjusts the weights of the pre-trained model.

3.2 Methodology Used in this Thesis for Airplane Detection

Deep learning algorithm used in this research is convolutional neural networks (CNNs) discussed above with some modifications in the network structures. The models are SSD and Faster R-CNN models. They are used models in object detection applications [9,10,11,12,15,74]. Two feature extractors are used as inception v2 and ResNet50.

Inception v2 is fast due to its factorization methods to reduce the computational complexity, where 5x5 convolution is factorized to two 3x3 convolutions leading to a great performance and to a boost in speed. Also, with inception v2, there is less loss of information since there are wider filter banks[75,76]. Illustrations are given in Figure 3.8 and 3.9, below where there are unexpended inception module and expended inception module (module with wider filter banks), respectively.

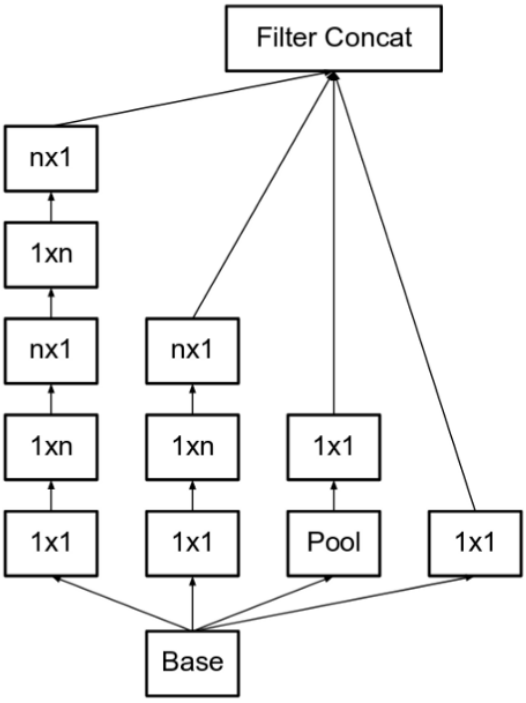

Figure 3.8: Not expended inception module [75,76].

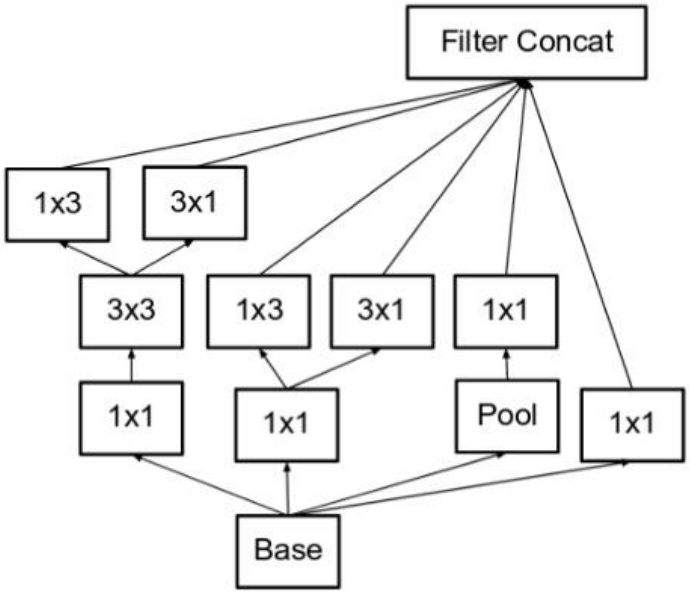

Figure 3.9: Expended inception module [75,76].

Both two inception modules from Figure 3.8 and 3.9 are equal. We can notice that from Figure 3.8, we have a wide network while in Figure 3.8, the network is deeper compared to the module in Figure 3.9.

ResNet-50 is a pretrained convolutional neural network trained over a million images from the ImageNet. It belongs to ResNet family, explained above. ResNet-50 is a deep network with 50 layers and takes its input as an image of 224x224, with the capability to categorize images into 1000 different object classes, such as pencil, animals, mouse, etc [70,77].

3.2.1 Single Shot Multibox Detector (SSD) model

SSD is a single feed-forward convolutional network which predicts classes (object categories) and anchor offsets (predicted bounding boxes) in a single way. The convolutional layers have different sizes of filter where the sizes reduce from the input layer to the output layer, 10x10, 5x5 and 3x3, with average pooling layer. With this model, there are about 7308 detections per class and non-maximum suppression (NMS). Lower and upper feature maps are used for detection. The top-most convolutional feature map is selected at a lower level to add it to a sequence of convolutional layers. SSD uses a convolutional filter to make the prediction and uses different predictors regarding different aspet ratio detection for multiple feature map

from the preview layer, which helps to detect multiple scales. Multiple layers are used for prediction. The conv4 3, conv7 (fc7), conv8 2, conv9 2, conv10 2, and conv11 2, from Figure 3.10 are used to predict confidences and location. The model creates bounding boxes over different aspect ratios and scales for each feature map. At the end, it creates a bounding box on the object category including the score of precision. It adjusts the box to the object shape such that the box can contain the whole object. The boxes are organized in order such that the box with the highest confidence is selected depending on the ratio between the positive and the negative. For more detail about SSD model, refer to [9].

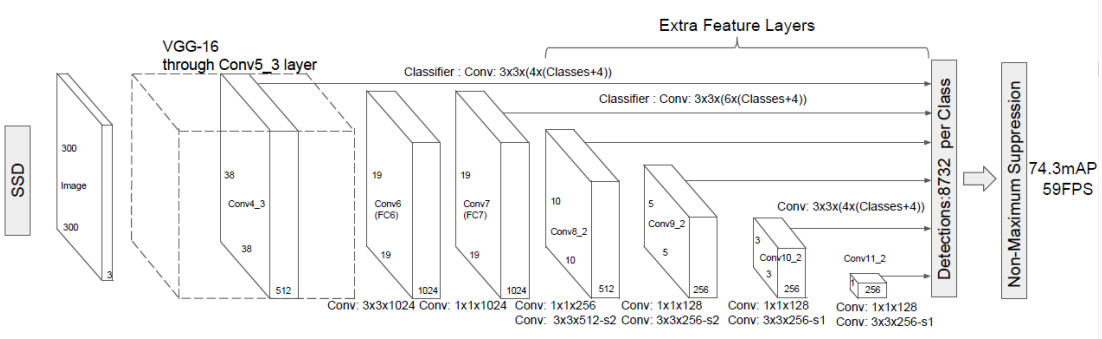

Figure 3.10: The network of Single Shot Multibox Detector (SSD) model [9].

In the Figure 3.10, they used VGG-16 as the base and with input image of size 300x300. Each bounding box has 4 parameters which are the width, the height, and the coordinates of the center (xc,yc), Figure 3.11.

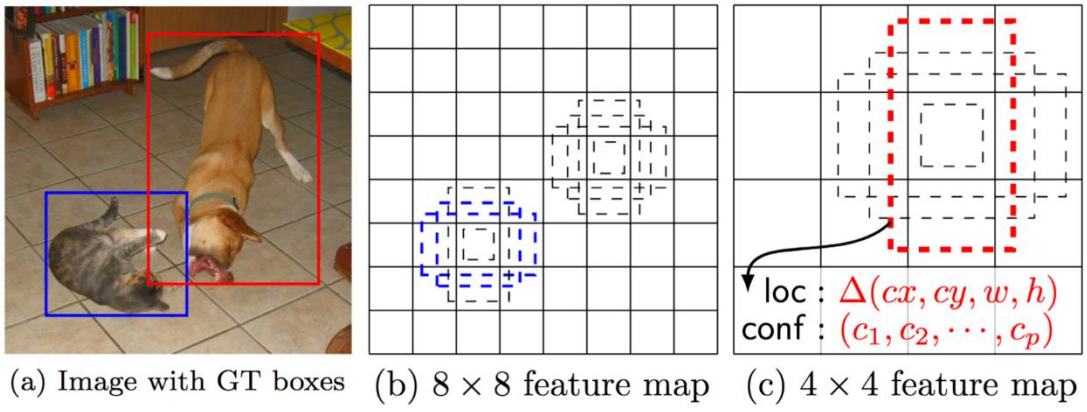

Figure 3.11: Single Shot Multibox Detector (SSD) model framework [11].

From Figure 3.11, in (a) the model takes, as input, an image including its ground truth (GT) bounding boxes. And in (b) and (c) the different feature map fixes sets of boxes

with different aspect ratios. Then, the training process will consist to best match the boxes localization to the ground truth.

The loss calculation for SSD model:

x_{ij}^{p}={1,0} is the indicator of maching *i-th* box to *j-th* ground truth box with *p* category. The loss function is calculated by averaging the sum of the localization loss(*ll*) and the confidence loss(*lc*) defined as follows:

$$L(x,c,l,g) = \frac{1}{N}(l_{ll}(x,c) + \rho l_{lc}(x,l,g)). \quad \textbf{(3.31)}$$

N the matched default box number, *l* is predicted box and *g* the ground truth box. *L* is the loss.

$$l_{ll}(x,l,g) = \sum_{i\in Pos}^{L} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m),$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \text{ and } \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h,$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \text{ and } \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h}),$$

$$l_{lc}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i\in Neg} \log(\hat{c}_i^0),$$

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)},$$

ρ is the weight term, (*cx*, *cy*) are center coordinates, and (*w*,*h*) are the bounding box (*d*) width and height.

To use *m* feature maps for prediction, the scale of the default boxes is calculated for each feature map. This scale calculation is defined as follows:

$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}$(k-1), k$\in [1, m]$ with $s_{min} = 0.2$ and s_{max}=0.9 for lowest layer and highest layer, respectively.

3.2.2 Faster Region-bassed Convolutional Network (Faster R-CNN) model

The faster R-CNN model (Figure 3.12) is built of convolutional neural networks. The model makes region proposals and at the end classifies the object by a bounding box. The bounding box shows the class of the object and the score in percentage to be an element of that class.

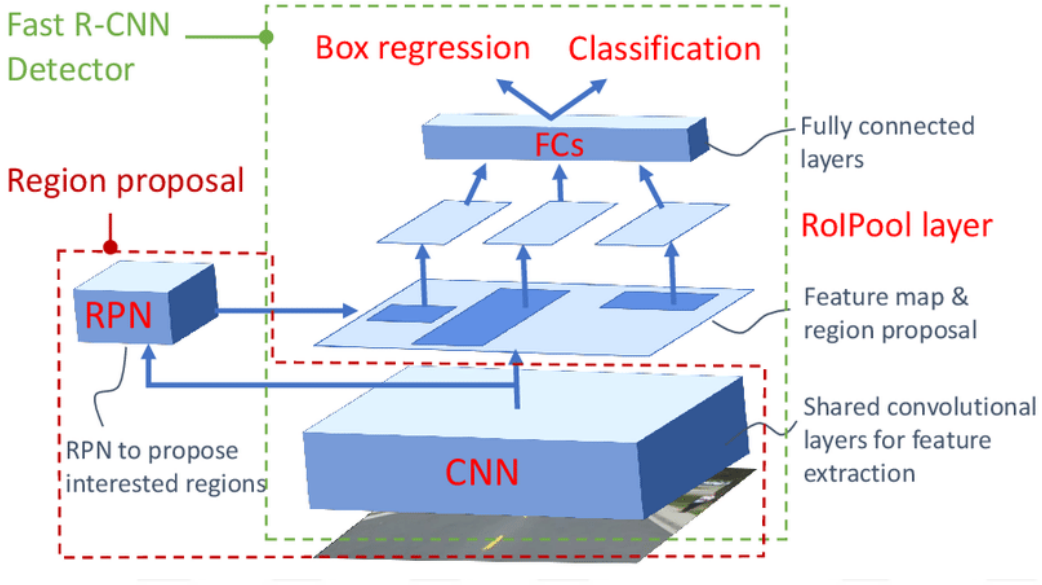

Figure 3.12: The framework of detection with Faster Region-bassed Convolutional Neural Network **(**Faster R-CNN) model [78].

Figure 3.12 shows the framework of Faster R-CNN. The Faster R-CNN is composed of a Fast R-CNN model and a Region Proposal Network (RPN). Fast R-CNN model is a model where a main CNN including several convolutional layers is used to take an entire image as input to produced feature maps. Then, a selective search method is used on the feature maps to generate Region of Interests (RoIs). Once RoIs are obtained, a RoI pooling layer is employed to reduce the feature maps size in order to get valid RoIs with fixed width and heigh and after these RoIs are fed into fully connected layers. Finally, objects are detected by a softmax classifier and the coordinates of the bounding box are modified by a linear regressor to match the ground truth.

RPN is a well-known fully-convolutional network used to generate region proposals, predict object at each position, and detect objects.

In Faster R-CNN model, a CNN model also takes the entire image as input to produce feature maps. A sliding window of size 3x3 outputs a features vector from the produced feature maps, then link them to two fully connected layers. One fully connected layer is for box-regression and other one for box-classification. These fully connected layers predict multiple region proposals. From a fixed number k of regions called anchors, the output of the box-regression layer gets a size of 4k including height, width, and coordinates of the boxes and a size of 2k for the box-classification layer output with objectness scores of detecting object or not.

The relevant anchor boxes are kept after they are detected and selected by following a threshold over the objectness score. A Fast R-CNN model is then fed by these anchor boxes and feature maps obtained from the initial CNN model.

The selective search method is avoided in Faster R-CNN by the usage of RPN, which improves the model performances, and speeds up the training and test processes. A pre-trained model from ImageNet dataset is used by the RPN for classification and fine-tuned on the PASCAL VOC dataset[52]. The Fast R-CNN is then trained using the generated region proposals with anchor boxes.

Loss function for Faster-RCNN model is:

$$L(\{p_i\},\{t_i\})=\frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \varphi \frac{1}{N_{reg}}\sum_i, p_i^* L_{reg}(t_i, t_i^*). \quad \textbf{(3.32)}$$

i is anchor index, predicted probability is p_i, p_i^* is the ground-truth. t_i is the vector for representing 4 coordinates of the predicted bounding box and t_i^* is for positive vector. L_{reg} is the regression loss and L_{cls} is the classification loss.

.

4. AIRPLANE DETECTION

In our experiments, we present airplane detection from very high-resolution satellite images using deep learning techniques with short training time and we detect targets regardless of their features with high speed detection. The deep learning models, we use, are the pre-trained model SSD and faster RCNN. Using an open-source software library, called TensorFlow with python. This flexible architecture, TensorFlow, allows deploying computation to one or more Central Processing Units (CPUs) or GPUs on a server, desktop, or mobile devices.

4.1 Dataset

The datasets used in this research were collected from open source data sets including some images from Northwestern Polytechnical University-REmote Sensing Image Scene Classification (WPU-RESISC45) dataset, WHURS19 dataset, Aerial Image Dataset (AID), and added to ITU-CSCRS dataset (Table 4.1). We used images from these datasets because there are composed of images with different resolutions, different scenes with different backgrounds, from different sensors and earth's regions as explained in the coming sub-sections (4.1.1 to 4.1.4). Also, the datasets were used in many scene classification and object detection projects [1,3,25,79,80,81,82,83,84] and they produced state-of-the-art results. All images from these sources are very high-resolution satellite images and mainly created for scene classification and object detection. Our dataset, with this variety of sources, is composed of 1402 images in total with 7 701 airplane objects in 1167 images other images contain non-airplane objects such as Jet Plane, High Building, Residence, Storage Tank, and Pool. Table 4.1 illustrates the dataset repartition per source and Figure 4.1 illustrates some samples. We used 80% of the dataset for training with 6 287 plane objects in 1 119 images and the remaining for testing including 1 414 plane objects in 283 images.

Table 4.1: Dataset sources.

Name of Dataset Source	Number of Images from the Dataset	Number of Airplane Object	Number of Images Containing Airplane Object	Resolution	Sensors
ITU-CSCRS dataset	262	222	114	0.5m; 300x300pixels	Pleiades 1B
AID dataset	360	4715	360	8m; 600x600pixels	Google Earth
WHU-RS19 dataset	53	896	53	0.5m; 600x600pixels	Google Earth
WPU-RESISC45 dataset	727	1868	640	30m; 256x256pixels	Google Earth

Figure 4.1: Some samples of our dataset for airplane detection.

The Table 4.1 illustrates our total data set reparation per source. We can observe that the 360 images collected from AID dataset have the largest number of airplane with 4715 airplane objects and the largest number of image was obtained from WPU-RESISC45 dataset with 727 images. In Figure 4.1, some samples of our data set are illustrated. We can see that the dataset is composed of images with airplane objects of different scales, orientrations, and different positions. The images have complex background, a variety of scenes. Once trained and tested on this dataset, the models will be able to produce the similar performance on different dataset.

4.1.1 NWPU-RESISC45 dataset [84]

NWPU-RESISC45 dataset is a collection of public data available for Remote Sensing Image Scene Classification (RESISC). It was built by Northwestern Polytechnical University (NWPU). This benchmark contains 31 500 images divided into 45 scene classes cropped from Google Earth. Some samples are shown in Figure 4.2.

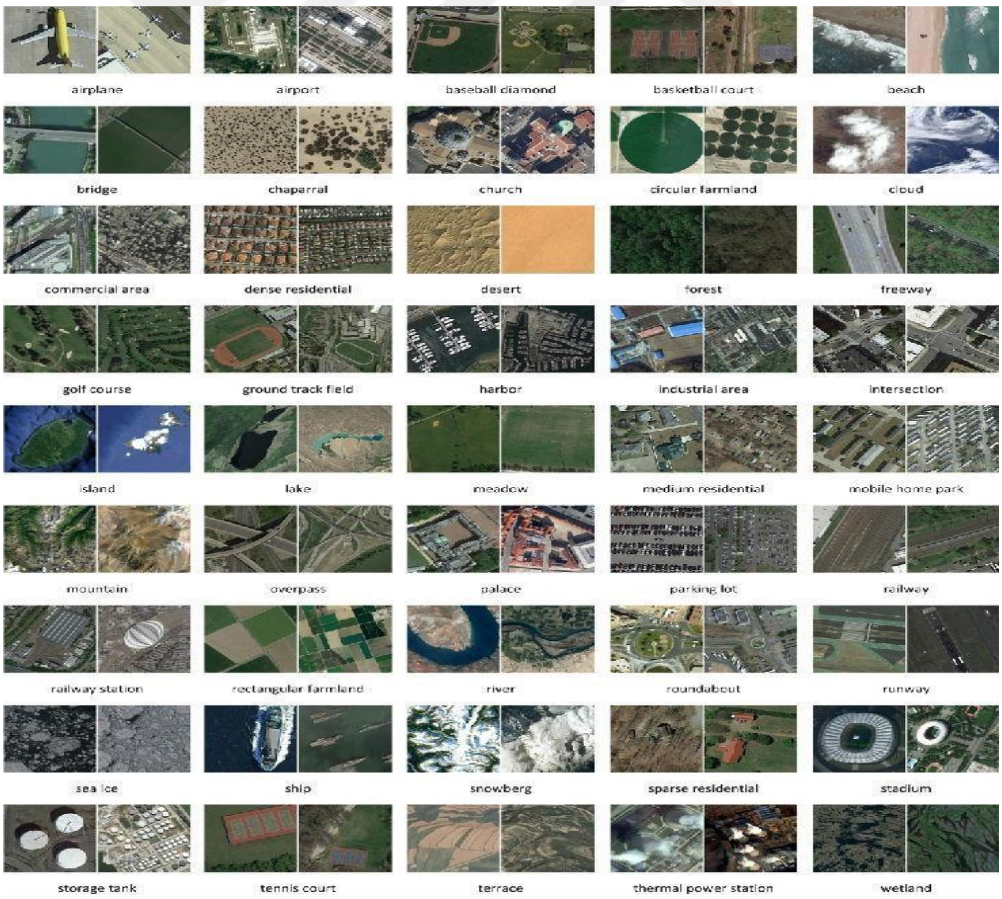

Figure 4.2: Some samples of NWPU-RESISC45 dataset [84].

Figure 4.2 has some images of NWPU-RESISC45 dataset, about 45 scene categories images including airplane, airport, baseball diamond, basketball court, beach, bridge, chaparral, church, circular farmland, cloud, commercial area, dense residential, desert, forest, freeway, golf course, ground track field, harbor, industrial area, intersection, island, lake, meadow, medium residential, mobile home park, mountain, overpass, palace, parking lot, railway, railway station, rectangular farmland, river, roundabout, runway, seaice, ship, snowberg, sparse residential, stadium, storage tank, tennis court, terrace, thermal power station, and wetland. In each category, there are 700 images with a size of 256 ×256 and their spatial resolution varies from 30 to 0.2 m per pixel. Here too, we only collected images regarding airplane and some images from airport category since in some airport images we have some airplane objects.

4.1.2 AID dataset [80]

The dataset is built for scene classification for remote sensing images. AID dataset is a collection of images of large-scale from Google Earth Imagery. The dataset contains 30 aerial scene categories such as Airport, Bare land, Baseball field, Beach, Bridge, Center, Church, Commercial, Dense residential, Desert, Farmland, Forest, Industrial, Meadow, Medium residential, Mountain, Park, Parking, Playground, Pond, Port, Railway station, Resort, River, School, Sparse residential, Square, Stadium, Storage tanks, and Viaduct. AID dataset includes a variety of scene image with 10 000 images within all these 30 categories. An illustration can be found in Figure 4.3. We only collected airport images since in some airport images we have some airplane objects.

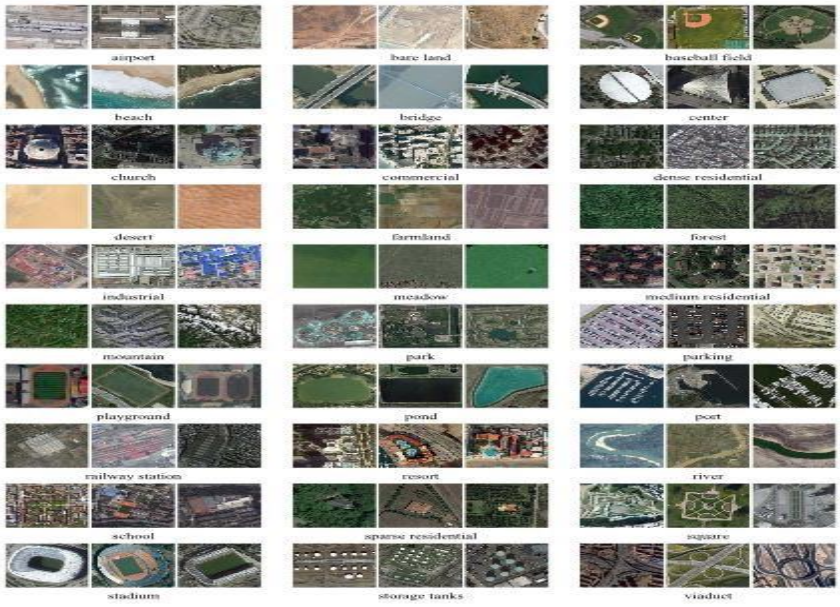

Figure 4.3: Some samples of AID dataset [80].

Figure 4.3 shows us some AID images. These images are from Google Earth. They have specific features. The dataset is taken from many regions and countries around the world such as England, Japan, and the United States. AID images are extracted at different time and seasons under different imaging conditions that affect the data by increasing their intra-class diversities. AID dataset images have their pixel resolution of 8 meters to 0.5 meters and the size of each is 600x600 pixels to cover a scene with various resolutions. The dataset was built in 2017.

4.1.3 WHU-RS19 dataset [25,85]

WHU-RS19 dataset has 19 classes of different scenes and each class has 55 images, with size of 600×600 pixels and pixel resolutions up to half a meter from Google Earth imagery. Those 19 categories include airport, bridge, river, forest, meadow, pond, parking, port, viaduct, residential area, industrial area, and commercial area, beach, desert, farmland, football field, mountain, park, and railway station. They are collected from different regions all around the world and the aerial scenes appear at different orientations, scales, and with different lighting conditions. We only selected images of airport since in some airport images we have some airplane objects.

4.1.4 ITU-CSCRS datasets

ITU-CSCRS dataset is built of very high-resolution remote sensing images from Pleiades satellites with 50 cm as spatial resolution, image taken at an altitude of 694km, and Spectral Bands (Pan: 0.47-0.83 μm; Blue=0.43-0.55 μm, Green = 0.50-0.62 μm, Red = 0.59-0.71μm, Near Infrared = 0.74-0.94 μm (NIR)). The dataset is built for scene classification and object detection. ITU-CSCRS dataset is not publicly open dataset. The dataset contains a large number of different scene images with these regions of Ataturk, Esenboga, Izmir, and Sabiha with a size of 300x300. The dataset includes the labeled files. Figure 4.4 illustrates some samples ITUCSCRS dataset. From ITU-CSCRS data set, we used 6 categories such as Plane, Jet Plane, High Building, Residence, Storage Tank, and Pool. However the dataset contains additional classes such as industrial areas, annual agricultural, roads bridges, forests natural lands, permanent agricultural, water bodies, parking lots, sport facilities.

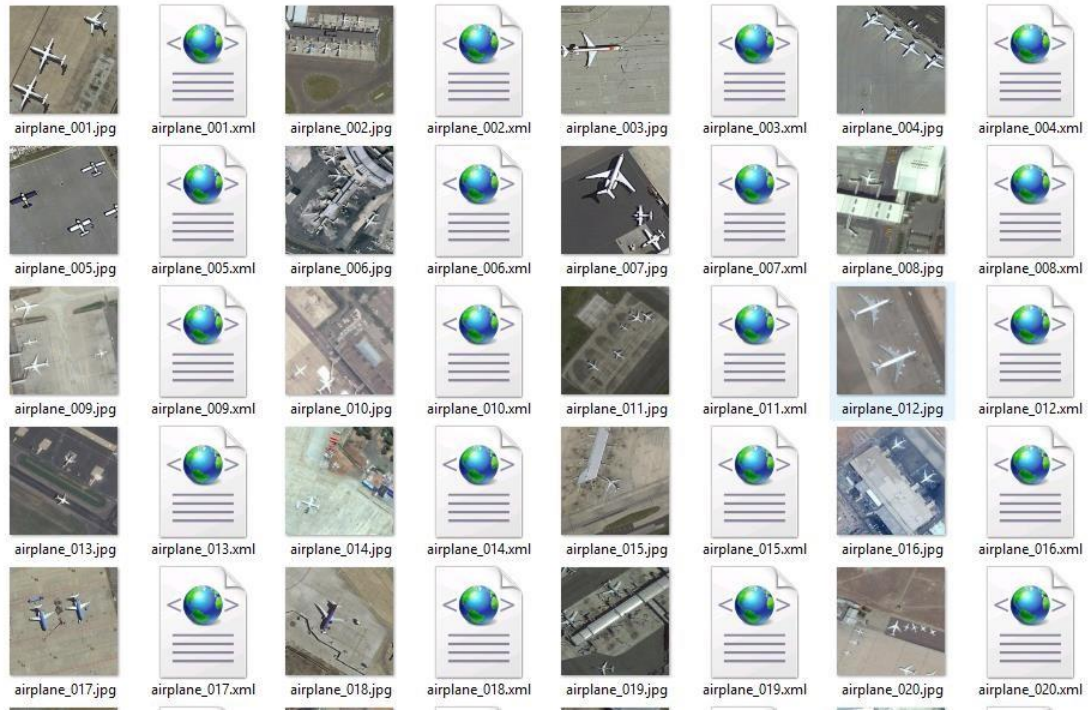

Figure 4.4: Some samples of ITU-CSCRS dataset with their labels.

4.2 Image Labeling

The images collected online from NWPU-RESISC45 dataset, AID dataset, and WHU-RS19 dataset, were not labeled. We labeled them with airplane class (Plane) and non-airplane classes. An image contains planes or non-planes with different size, different

shapes, different colors, and different orientations. The final images in the final dataset are regrouped images with different sizes and resolution like 256x256, 300x300, and 600x600, since they are from a variety of sources. Thus, the dimensions of the shape resizer of SSD models were fixed to 300x300 and for Faster R-CNN models aspect ratio resizer dimensions were kept to 200x250.

The image with a different type of class, of scene and object, are labeled. Figure 4.5, below, illustrates the labeling process.

Figure 4.5: Labelling sample.

The desired object belonging to a class is selected using a bounding box with a rectangular shape. The dimensions of this bounding box depend on the dimensions of the object being selected as belonging to a class. The final dataset contains plane and

non-plane objects of different scales, small with area less than 1024 pixels, medium with area between 1024 and 9216 pixels, and large scale with area greater than 9216 pixels, according to COCO challenge object size. Figure 4.6 presents the bounding box areas of plane and non-plane objects in the dataset.

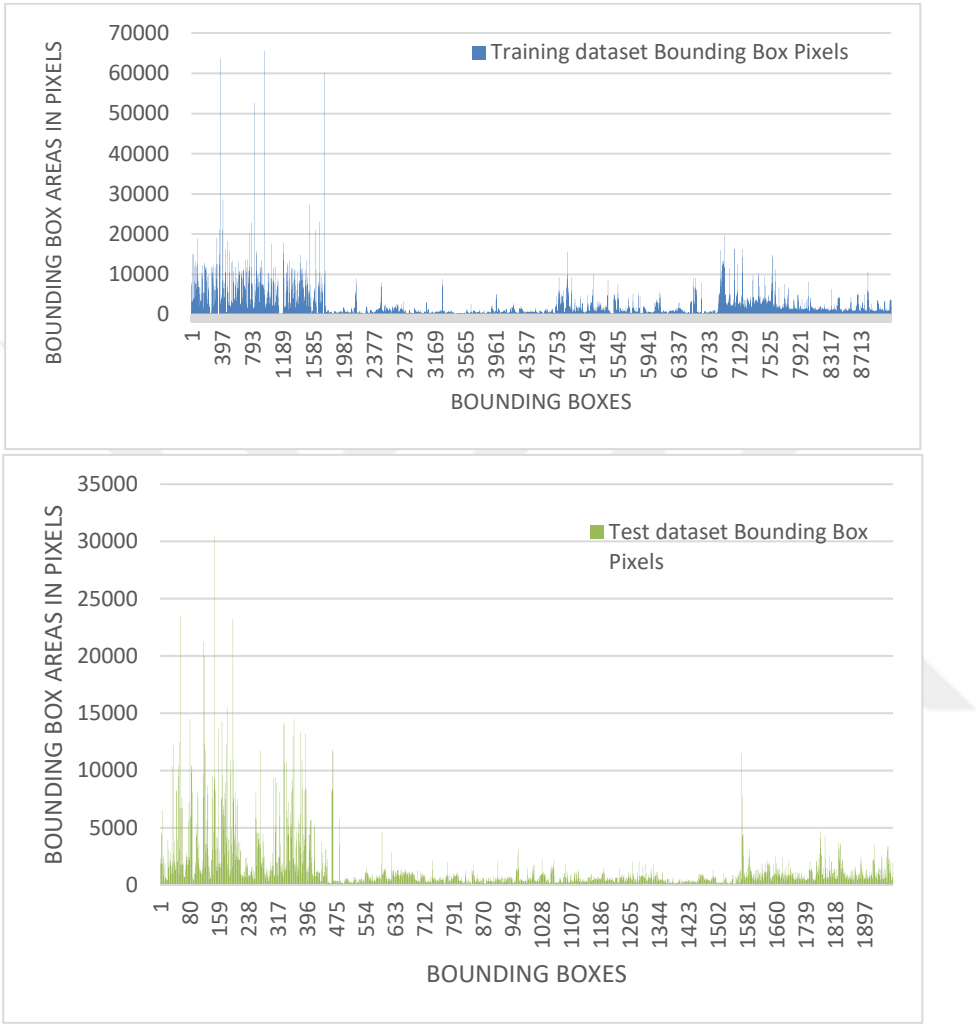

Figure 4.6: Number of pixels of bounding boxes in the training and test datasets.

In Figure 4.6, the grapth in blue represents our training dataset bounding box areas in pixels and the one in green represents our test dataset bounding box areas in pixels. The value of pixels of the bounding boxes varies from 0 to 65 000 pixels and 0 to 30 000 pixels for training and test samples, respectively. In total, there are 5336 airplane samples of small scale, 2210 airplane samples of medium scale, and 155 airplane samples of large scale within these 7701 airplane objects.

4.3 Methods

Airplane detection is done in this thesis using two main models, the SSD model and the Faster R-CNN model. They are pre-trained models of CNN and are the most used for object detection.

We have many experiments processed under different environments and conditions.

Environment 1 is in ITU-CSCRS, using a computer with following properties: GeForce GTX 1050 Ti, RAM: 4GB, Windows 10, 3TB, Python 3.6.8, TensorFlow-gpu 1.13.1.

Environment 2 is a different environment. It is on a computer with the following properties: GeForce 940MX, RAM: 12GB, Windows 10, 1TB, Python 3.6.8, TensorFlow-gpu 1.13.1, Anaconda3 python 3.7.

Environment 3 is a different environment. It is on a computer with the following properties: DELLWS Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz(2 units)x40 Thread Quadro P6000 24GB(2 units) 128 Gb.

Some SSD models such SSD 1 to 3 were trained in the environment 1 and all other models were trained in environment 2 including Faster RCNN models. And all evaluations were done in the environment 2, except the trainings of Faster RCNN 6 to 8 which were done in the environment 3.

We used the pretrained models of Microsoft COCO [86,87]. Microsoft COCO is a large-scale segmentation, object detection, and captioning dataset. It has several features regarding detection such as recognition in context, object segmentation, Super-pixel stuff segmentation, 330 000 images with more than 200 000 labeled, 80 object categories (plane, person, etc.), 1.5 million object instances, 91 stuff categories, 5 captions per image, and 250,000 people with key-points. For more detail about Microsoft COCO, refer to [86].

4.3.1 Training models

We have fifteen models with seven models based on SSD and eight based on Faster RCNN (FRCNN). Each model has specific configurations. Below tables, from Table 4.2 to 4.6, represent networks and configurations of these 7 SSD models, and from

Table 4.7 to 4.12, we have the networks and configurations of the four Faster RCNN models. Anchor generator properties, training configuration, box predictor properties, feature extractor properties, loss type, and post processing parameters can be found in these tables.

4.3.1.1 SSD models

Below tables from Table 4.2 to 4.6, anchor generator properties, box predictor properties, feature extractor properties, loss type, and training configuration of SSD models can be found, respectively. In the configuration of the models, all the parameters were not changed at the same time. To evaluate the effect of a parameter, we changed one parameter and others remained unchanged and the changes can be seen in bold in the tables.

Table 4.2: Anchor generator of SSD models.

	SSD 1/2	**SSD 3/4/5/6/7**
Number layers	6	6
Min scale	0.2	0.2
Max scale	0.95	0.95
Reduce boxes in lowest layer	false	**true**

Table 4.3: Box predictor of SSD models.

	SSD 1/2	**SSD 3/4/7**	**SSD 5**	**SSD 6**
Min depth	0	0	0	0
Max depth	0	0	**6**	0
Number of layers before predictor	0	0	**2**	0
Kernel size	**5**	3	3	3
Activation Function	RELU_6	RELU_6	RELU_6	**RELU**
L2 regularizer (weight)	0.00004	0.00004	0.00004	0.00004
Truncated normal initializer (stddev)	0.03	0.03	0.03	0.03
Truncated normal initializer (mean)	0.0	0.0	0.0	0.0

Table 4.4: Feature extractor of SSD models.

	SSD 1/2/3/5/7	SSD 4	SSD 6
Type	inception v2	inception v2	inception v2
Min depth	16	16	16
Depth multiplier	1.0	1.0	1.0
Activation Function	RELU_6	RELU_6	**RELU**
L2 regularizer (weight)	0.00004	0.00004	0.00004
Truncated normal initializer (stddev)	0.03	**0.003**	0.03
Truncated normal initializer (mean)	0.0	0.0	0.0
Batch norm (decay)	0.9997	0.9997	0.9997
Batch norm (epsilon)	0.001	0.001	0.001

Table 4.5: Loss type of SSD models.

	SSD 1/2/3/4/5/6	SSD 7
Classification loss	Weighted sigmoid	Weighted sigmoid
Localization loss	Weighted smooth l1	Weighted smooth l1
Classification weight	1.0	1.0
Localization weight	1.0	1.0
Normalize loss by num matches	true	true
Post processing (score converter)	SIGMOID	**SOFTMAX**

Table 4.6: Training configuration of SSD models.

	SSD 1	SSD 2	SSD 3	SSD 4	SSD 5/6/7
Batch size	8	8	**24**	8	8
Optimizer	RMS prop optimizer	RMS prop optimizer	RMS prop optimizer	RMS prop optimizer	RMS prop optimizer
Initial learning rate	0.002	0.002	**0.004**	**0.00004**	0.004
Decay steps	800720	800720	800720	**10000**	10000
Decay factor	0.95	0.95	0.95	0.95	0.95
Momentum optimizer value	0.9	0.9	0.9	0.9	0.9
RMS prop optimizer (decay)	0.9	0.9	0.9	0.9	0.9
RMS prop optimizer (epsilon)	1.0	1.0	1.0	1.0	1.0
Number of steps	**1000000**	**800000**	**1700667**	**210000**	**210000**

For SSD 3 to 7, we reduced boxes in lowest layer to speed up the training (Table 4.2). For the SSD 5 model, we added 2 layers before predictor with maximum depth of 6 to see the effet of a SSD model with extra layers and also changed RELU_6 to RELU in

SSD 6 model (Table 4.3). RELU as explained in sub-section 3.1.4.2, has a general range of [0, x], where x can take any value from 0 to infinite. RELU_6 has a rang of [0, 6] with x equaling to 6 and RELU has a range of [0, inf). We changed RELU_6 to RELU to see how this infinite range of RELU can bring a plus in our airplane detection although it can blow up the activation. Also, we reduced the standard deviation (stddev) value of truncated normal to 0.003 for SSD 4 model (Table 4.3). Truncated normal is used to overcome saturation during the model training. In the score converter level, we changed in SSD7 the activation function to the softmax function in order to apply what was discussed in sub-section 3.1.4.2, aslo to analyze the effet of softmax function on the scores comparing to sigmoid activation function (Table 4.5). From the SSD 3 to SSD 7 models, we fixed kernel size to 3x3. For the trainings, we had different learning rates and step sizes as it can be observed from Table 4.6. We had a decay learning where the learning rates were decreasing at a determined step. For instance, in SSD 5, the total step was setted to 210 000 with initial learning rate value of 0.004, and a decay factor and step of 0.95 and 10 000, respectively. The batch size for all the models was 8, except for SSD 3 model which was 24, to analyze the effect of batch size.

4.3.1.2 Faster R-CNN models

In the tables, below, from Table 4.7 to 4.12, anchor generator properties, box predictor properties, feature extractor properties, loss type, and training configuration of Faster R-CNN models can be found, respectively. In the configuration of the models, all the parameters were not changed at the same time as in SSD models. To evaluate the effect of a parameter, we only changed one parameter as in SSD models and others remained unchanged and the changes can be seen in bold in the below tables.

Table 4.7: First stage feature extractor of Faster RCNN models.

	Faster RCNN 1/2/5/6/7/8	**Faster RCNN 3/4**
Type	Faster RCNN Inception v2	**Faster RCNN Resnet50**
Features stride	16	16

Table 4.8: First stage box predictor of Faster RCNN models.

	Faster RCNN 1/3/7/8	**Faster RCNN 2/4**	**Faster RCNN 5/6**
Predictor type	CONV	CONV	CONV
Regularizer	L2 regularizer (weight=0/0/**0**.04/0)	L1 regularizer (weight=0.0)	**L1 regularizer (weight=0.004/0.04/0.04)**
Truncated normal initializer (stddev)	0.01	0.01	0.01

Table 4.9: First stage post processing parameters of Faster RCNN models.

	Faster RCNN 1/2/3/4/5/6/7/8
NMS score threshold	0.0
NMS IOU threshold	0.7
Localization weight	2.0
Objectness weight	1.0
Max-pool kernel size	2

Table 4.10: Second stage box predictor of Faster RCNN models.

	Faster RCNN 1/3/7/8	**Faster RCNN 2/4**	**Faster RCNN 5/6**
predictor type	Mask RCNN box predictor (FC)	Mask RCNN box predictor (FC)	Mask RCNN box predictor (FC)
Regularizer	L2 regularizer (weight=0/0/0.04/0)	L1 regularizer (weight=0.0)	**L1 regularizer (weight=0.004/0.04)**
variance_scaling_initializer (factor)	1.0	1.0	1.0

Table 4.11: Second stage post processing parameters of Faster RCNN models.

	Faster RCNN 1/2/3/4/5/6/7/8
Batch non max suppression (score threshold)	0.0
Batch non-max suppression (iou threshold)	0.6
Localization weight	2.0
Classification weight	1.0
Score converter	SOFTMAX

Table 4.12: Training configuration of Faster RCNN models.

	Faster RCNN 1/2/3/4	**Faster RCNN 5**	**Faster RCNN 6/7**	**Faster RCNN 8**
Batch size	1	2	1	1
Optimizer	Momentum optimizer	Momentum optimizer	Momentum optimizer	RMS prop optimizer
Initial learning rate	0.002	0.00002	0.0002	0.002
Learning rate from 100000 steps	0.0002	-	0.00002	0.0002
Learning rate from 200000 steps	0.00002	-	0.000002	0.00002
Learning rate from 260000 steps	-	0.000002	-	-
Learning rate from 500000 steps	-	0.0000002	-	-
Momentum optimizer value	0.9	0.9	0.9	0.9
Number of steps	210000	340279	210000	210000

In the seek of a good performance in our airplane detection, we also used different properties in Faster R-CNN models. The inception v2 feature extractor in the model of Faster R-CNN 3 and 4 was changed to Faster R-CNN Resnet_50 feature extractor(Table 4.7). In the models of Faster R-CNN 2 and 4, we changed L2 regularizer, used in the models of Faster R-CNN 1 and 3 , to L1 regularizer (Table 4.8 and Table 4.10). Regularization is used for feature selection and to avoid overfitting. The difference between the L2 regularizer and L1 regularizer is that the first one is used to add a squared magnitude of coefficient called penalty to the loss functions. In Table 4.9 and Table 4.11, the default values were used as in Microsoft COCO [76,77] . For the training of Faster R-CNN 1/2/3/4/8 models, we fixed the training step to 210 000 steps with an initial learning rate of 0.002, and from the 100 000th and 200 000th steps, a learning rate value of 0.0002 and 0.00002, respectively, were setted. The batch size was 1 (Table 4.12) as in the original document. With our above defined parameters, any batch size bigger than 1 that was tried, leaded to overfitting in the training or to memory allocation errors as mentioned in [12] with Faster R-CNN models. However, we were able to train and detect our targets with low detection performance using batch size of 2 and with different value of L1 regularizer weight and also learning rate (Table 4.10, Table 4.12). In the Faster R-CNN models, we used momentum optimizer, whereas in SSD models and in Faster R-CNN 8 model, we used Root Mean Square Propagation optimizer (RMS Prop optimizer). Momentum optimizer is used for guiding the search during the training by using the actuel step gradient and also used to accumulate the preview step gradients in order to find the next direction. On the other hand, RMS Prop selects each parameter learning rate. The difference between both optimizers momentum and RMSProp optimizers, is that momentum is used to accelerate the search in the minima direction, whereas RMSProp is to impede the search during the training in the oscillation directions. Table 4.13 is the summary of all the models including Faster R-CNN and SSD models.

Table 4.13: Summary of all models.

Model	Model based on	Optimizer	Batch size	Initial Learning rate	Decay factor-step	Number of Steps	Main Change
SSD 1	SSD - inception v2	RMS prop optimizer	8	0.002	0.95-800720	**1 000 000**	**In bold**
SSD 2						**800 000**	**In bold**
SSD 3			**24**	0.004		**1 700 667**	**In bold**
SSD 4			8	**0.00004**	0.95-10000	210 000	Truncated normal initializer (stddev) =0.003
SSD 5				0.004			Added 2 layers before predictor and max depth=6
SSD 6							RELU
SSD 7							Softmax
FRCNN 1	Faster RCNN Inception v2	Momentum optimizer	1	0,002	0.0002-100 000& 0.00002-200 000		L2 regularizer
FRCNN 2							L1 regularizer
FRCNN 3	Faster RCNN Resnet50						L2 regularizer
FRCNN 4							L1 regularizer
FRCNN 5	Faster RCNN Inception v2		**2**	**0.00002**	**0.000002-260 000 & 0.0000002-500 000**	**340 279**	**In bold**
FRCNN 6			1	0.0002	0.00002-100 000 & 0.000002-200 000	210 000	L1 regularizer
FRCNN 7							L2 regularizer
FRCNN 8		**RMS prop optimizer**		0,002	0.0002-100 000& 0.00002-200 000	210 000	**In bold**

This Table 4.13 summarizes all the models. The difference between each model is showing in the last column of the Table 4.13 with the name "Main change". "In bold" means that in the same row the value or values which is/are in bold represent the main change. To get rid of overfitting, in FRCNN 5 to 6, we used L1 and L2 Regularizers with different weight values (Table 4.10, Table 4.12). For FRCNN 6 training, we made the training up to 106 7425 steps where the initial learning rate was 0,002, L1 regularizer (weight = 0,004), with no dropout, and after 100 000th step the learning rate was reduced to 0.0002. From 106 7425th step, initial learning rate was setted to 0,0002, L1 regularizer (weight = 0,04), with dropout, and after 200 000th step the learning rate

was reduced to 0.00002. the same configuration as FRCNN 6 model was used for FRCNN 7, however for FRCNN 7, L2 regularizer was used and the best metric results for FRCNN 7 was observed at 51 502th step after the 210 000th step. These properties of our models helped them to boost their performances in airplane detection.

4.3.2 Evaluation configuration

For all models, the same evaluation configuration was used. The number of examples was 1 119 from training set and 283 from test set. In total, there are 5 336 airplane samples of small scale, 2210 airplane samples of medium scale, and 155 airplane samples of large scale within these 7 701 airplane objects (Table 4.14). The COCO detection metrics were taken for metric set by including metrics per category since we had plane and non-plane categories. All these examples were evaluated successfully for the validation of our models.

Table 4.14: Number of airplane objects per scale.

	Small	**Medium**	**Large**
Training Data Set	4313	1857	121
Test Data Set	1023	353	34

We observe from Table 4.14 that the data sets mostly contain small scale airplane objects compared to medium and large scale objects.

5. EXPERIMENTAL RESULTS

Here we present our airplane detection results. The results from the training and from the evaluation are given followed by the analysis and classification of the models in terms of their performances.

5.1 Training Results

Here we have the total loss of eleven experiments with different models. The total loss includes the classification loss, localization loss, clone loss, and the regularization loss (Figure 5.1 and Figure 5.2).

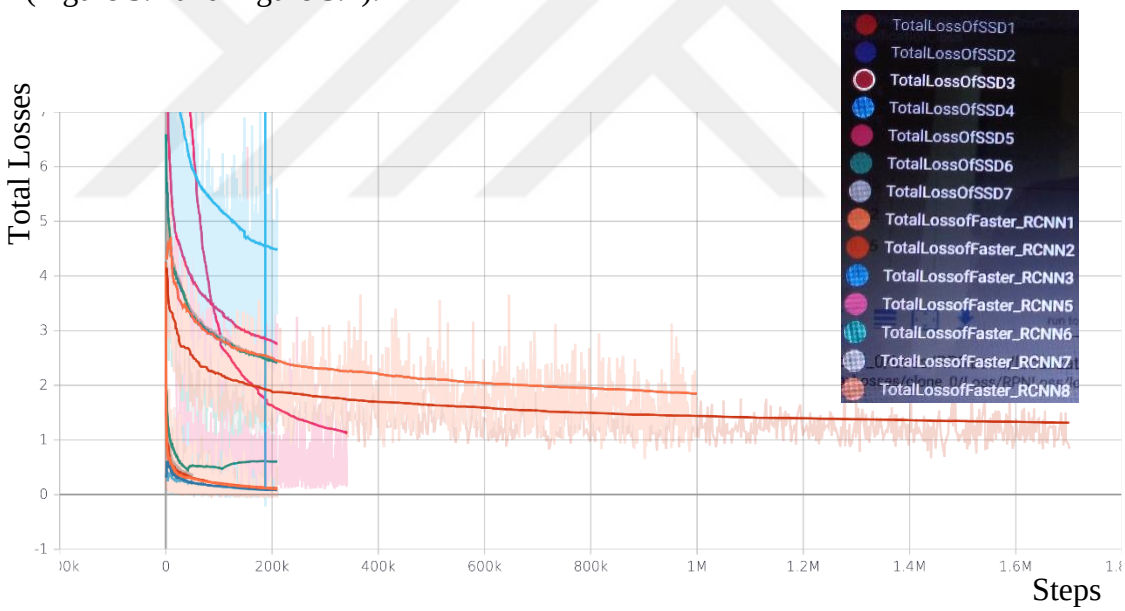

Figure 5.1: Total loss graph in terms of steps.

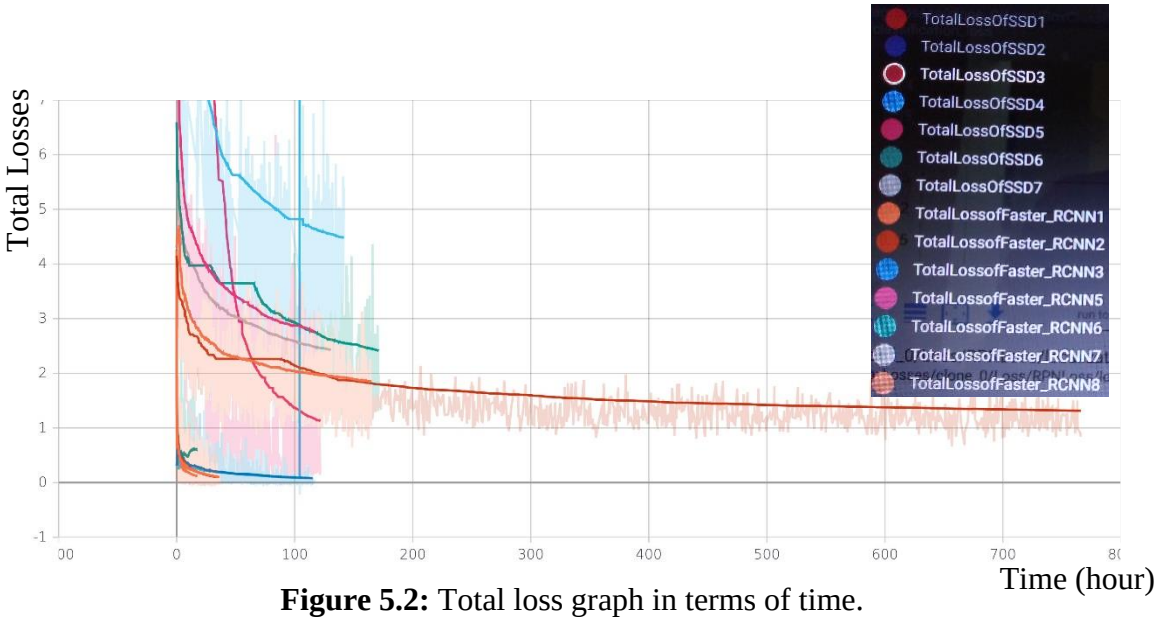

Figure 5.2: Total loss graph in terms of time.

The Figure 5.1 and 5.2 represent the total loss graph of all trainings with SSD 1 to SSD 7 models and with Faster_RCNN 1 to 8. The grapths with total loss value below 1 are the grapths of the eight Faster_R-CNN models. Figure 5.1 shows that among the models, SSD 4 models has the highest total loss value (5.398) and the lowest total loss value (0.0021) going to Faster_RCNN 1 model. SSD 1 and SSD 2 models have the same graph with 1 million steps. On the other hand, the Figure 5.2 represents the total losses in terms of elapsed time for each model training. We also observe that from the Figure 5.2, SSD 3 model training took more time compared to others, about 31d20h48m37s, and the Faster_RCNN models present the shortest training time, specially Faster_RCNN 7 model with 4h44m53s.

5.2 Evaluation Results

Table 5.1 presents the detection times of all models on test and training sets.

Table 5.1: Total detection time in second for all models.

Models	Test Set	Training Set
SSD 1	3.4	**14.61**
SSD 2	4.53	22.8
SSD 3	3.68	15.2
SSD 4	6.23	19.08
SSD 5	4.49	20.45
SSD 6	4.39	17.98
SSD 7	3.98	15.97
FRCNN 1	5.81	23.25
FRCNN 2	5.67	23.48
FRCNN 3	5.96	23.16
FRCNN 4	6.67	23.73
FRCNN 5	**2.9**	24.97
FRCNN 6	5.82	25.13
FRCNN 7	6.17	29.02
FRCNN 8	6.01	25.85

From the Table 5.1, we observe that Faster_RCNN 5 model has the lowest detection time (fastest detection time) in test set and SSD 1 model has the lowest detection time in training set. FRCNN 4 has the highest detection time in test set, whereas FRCNN 7 has the highest detection time in training set. The highest detection time observed in training set compared to test set is due to the fact that training set contains 1119 images, whereas test set contains 283 images.

5.2.1 Performance metrics

The object detection regroups many challenging tasks such as classification and regression tasks. During the object detection process, the models generate many bounding boxes with different confidence values. Among these boxes, the boxes with low score are eliminated for the computation of the spatial precision, used to evaluate and determine the metric values which are between 0 and 1. Addition to that, the Intersection over Union (IoU) area is also used, with its value located between 0 and 1. IoU area is the overlapping area between the ground-truth box and the predicted box. A higher IoU is a sign of a better predicted location of the bounding box. Usually, all bounding box candidates are kept with an IoU area equal or greater than some threshold values. In our airplane detection, we take this threshold to be 0.5 as defined

in COCO object detection challenge. Then, when a box is detected with this threshold, is considered as a true prediction.

True positive (TP), False positive (FP), and False negative (FN) are used in the calculation of precision and recall to determine the performance of a model (Eq. 5.1 and Eq. 5.2).

TP is correctly identified prediction for each class, that is for IoU > 0.5, the object is truly detected as the aimed object. FP is incorrectly identified predictions for certain class, in another term FP is for IoU < 0.5, the detected object which is defined to not be the target is not truly the target object. The FN is incorrectly rejected for certain class, in another term FN is for IoU > 0.5 with a miss classification of the object, in this situation, another object was detected as the target object.

Precision is used to determine how accurate is the predictions or data point proportions that are said to be relevant by the model and are actually relevant. Recall is used to determine how a model is able to find all the data points of relevant cases or of interest.

$$Precision = \frac{TP}{TP+FP} \quad \textbf{(5.1)}$$

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(5.2)}$$

F1 Score is relative to both precision and the recall. For imbalanced data, F1 Score is used to compare different models. F1 Score is said to be the harmonic mean of precision and the recall, it is located between them and it describes better the performance of the models using the recall and the precision than the accuracy for imbalanced data. Then, the model with the highest F1 Score is considered to be the best model.

$$F1\ Score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \quad \textbf{(5.3)}$$

The Average Precision (AP) metric summarizes the precision-recall curve. In general, the mean Average Precision (mAP) metric, which is the mean of the AP metrics, is used for object detection challenges. The mAP score is computed by taking account certain IoU values, for instance 0.5, 0.75. An official metric was developed by the COCO challenge that helps to avoid box over-generations, where a mean of the mAP

is computed for different sets of IoU (Table 5.2). It also helps to avoid many wrong classifications[11]. Average precision and Average Recall (AR) represent averaged over multiple Intersection over Union (IoU) values.

5.2.2 Metric results and performance analysis

We determined the performance of our models using COCO performance metrics [88] calculation methods. It includes 12 metrics shown in Table 5.2.

Table 5.2: The 12 performances metrics of COCO [88].

Metric Name	Calculation methods
Metric1	Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]
Metric2	Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]
Metric3	Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]
Metric4	Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]
Metric5	Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]
Metric6	Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]
Metric7	Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]
Metric8	Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]
Metric9	Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]
Metric10	Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]
Metric11	Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]
Metric12	Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]

10 IoU is used with thresholds starting from 0.50 to 0.95 with increment value of 0.05. The average precision is calculated for IoU of 0.5 and 0.75. Also, the AP and AR are calculated for IoU varying from 0.5 to 0.95 with increment of 0.05 for all object scales and for each of the three object scales separately. In accordance with COCO object scale, the small objects are object with area less than 32x32 (area < 32^2) pixels., medium objects are objects with area between 32x32 and 96x96 (32^2 < area < 96^2) pixels, and large objects are objects with area bigger than 96x96 (area > 96^2) pixels. AR is the maximum recall taking a fixed averaged over categories, number of detections per image, and IoUs (Table 5.2). The obtained metric values from our models are presented from Table 5.3 to Table 5.7.

Table 5.3: The 12 metric values obtained from our models using validation set.

	Metric1	**Metric2**	**Metric 3**	**Metric 4**	**Metric 5**	**Metric 6**	**Metric 7**	**Metric 8**	**Metric 9**	**Metric 10**	**Metric 11**	**Metric 12**
SSD 1	0.548	0.788	0.655	0.450	0.643	0.819	0.167	0.564	0.593	0.502	0.668	0.880
SSD 2	0.565	**0.794**	0.691	0.459	0.628	**0.838**	0.167	0.577	0.605	0.505	0.656	**0.884**
SSD 3	**0.593**	**0.794**	**0.723**	**0.492**	**0.687**	0.826	**0.175**	**0.600**	**0.634**	**0.542**	**0.713**	0.847
SSD 4	0.144	0.296	0.127	0.104	0.279	0.652	0.066	0.186	0.236	0.210	0.369	0.739
SSD 5	0.221	0.494	0.149	0.196	0.406	0.587	0.084	0.293	0.348	0.342	0.480	0.806
SSD 6	0.246	0.501	0.190	0.222	0.453	0.575	0.112	0.293	0.352	0.350	0.521	0.841
SSD 7	0.258	0.528	0.209	0.234	0.473	0.728	0.116	0.290	0.349	0.343	0.538	0.802
FRCNN 1	0.125	0.264	0.122	0.064	0.378	0.553	0.062	0.171	0.207	0.140	0.453	0.773
FRCNN 2	0.125	0.247	0.111	0.062	0.349	0.523	0.058	0.182	0.217	0.163	0.423	0.717
FRCNN 3	0.136	0.255	0.121	0.080	0.276	0.574	0.070	0.206	0.251	0.201	0.362	0.761
FRCNN 4	0.129	0.282	0.113	0.073	0.308	0.576	0.054	0.204	0.238	0.180	0.401	0.765
FRCNN 5	0.014	0.033	0.012	0.005	0.041	0.329	0.023	0.049	0.054	0.026	0.103	0.553
FRCNN 6	0.093	0.172	0.091	0.031	0.232	0.582	0.044	0.147	0.175	0.101	0.328	0.731
FRCNN 7	0.128	0.256	0.112	0.066	0.392	0.610	0.072	0.165	0.198	0.136	0.470	0.771
FRCNN 8	0.115	0.222	0.111	0.070	0.272	0.531	0.054	0.172	0.216	0.154	0.390	0.759

Table 5.4: The 12 metric values obtained from our models using training set.

	Metric 1	**Metric 2**	**Metric 3**	**Metric 4**	**Metric 5**	**Metric 6**	**Metric 7**	**Metric 8**	**Metric 9**	**Metric 10**	**Metric 11**	**Metric 12**
SSD 1	0.546	0.755	0.647	0.476	0.583	0.792	0.184	0.568	0.607	0.539	0.643	0.852
SSD 2	0.533	0.745	0.620	0.473	0.594	0.779	0.181	0.555	0.597	0.532	0.656	0.842
SSD 3	0.564	0.744	0.651	0.509	0.620	0.823	0.182	0.585	0.635	0.567	0.690	0.876
SSD 4	0.317	0.584	0.285	0.183	0.396	0.797	0.133	0.376	0.453	0.336	0.510	0.830
SSD 5	0.612	0.931	0.720	0.512	0.719	0.882	0.197	0.607	0.686	0.595	0.779	0.902
SSD 6	0.656	0.914	0.774	0.582	0.802	0.798	0.210	0.670	0.759	0.661	0.865	0.945
SSD 7	0.693	**0.954**	0.810	0.596	0.804	0.936	0.208	0.662	0.747	0.669	0.829	0.951
FRCNN 1	0.753	0.849	0.802	0.632	**0.908**	0.966	0.234	0.721	0.792	0.684	**0.933**	0.977
FRCNN 2	0.758	0.842	0.807	0.627	0.895	0.981	0.235	0.718	0.792	0.679	0.910	0.988
FRCNN 3	0.739	0.813	0.782	**0.647**	0.883	0.900	0.238	0.727	0.801	0.687	0.917	0.986
FRCNN 4	**0.771**	0.856	**0.817**	0.645	0.899	**0.985**	**0.241**	**0.732**	**0.804**	**0.696**	0.916	**0.989**
FRCNN 5	0.016	0.041	0.011	0.006	0.034	0.250	0.038	0.075	0.098	0.048	0.104	0.581
FRCNN 6	0.215	0.381	0.192	0.081	0.345	0.715	0.099	0.252	0.281	0.116	0.430	0.759
FRCNN 7	0.365	0.591	0.419	0.179	0.526	0.808	0.132	0.397	0.442	0.256	0.605	0.844
FRCNN 8	0.675	0.757	0.721	0.590	0.880	0.980	0.233	0.702	0.773	0.649	0.917	0.986

Table 5.5: mAP at small, medium and large plane detection using validation and training sets.

	Test Set			**Training Set**		
	Small	**Medium**	**Large**	**Small**	**Medium**	**Large**
SSD 1	0.449609	0.760926	0.814312	0.368176	0.735843	0.835452
SSD 2	0.403835	0.742038	0.815370	0.340572	0.687708	0.824858
SSD 3	**0.475712**	**0.785016**	**0.831881**	0.405855	0.746742	0.852958
SSD 4	0.095438	0.509707	0.727744	0.103117	0.580517	0.839742
SSD 5	0.145255	0.568625	0.708467	0.243196	0.733639	0.916648
SSD 6	0.158505	0.567684	0.565078	0.309269	0.789808	0.626850
SSD 7	0.144763	0.583679	0.778105	0.275339	0.764882	0.955378
FRCNN 1	0.062302	0.496406	0.740782	0.383810	0.907741	0.995662
FRCNN 2	0.062694	0.480138	0.701221	0.384946	0.907599	0.996928
FRCNN 3	0.080742	0.489953	0.733614	**0.460786**	0.916084	**1.000000**
FRCNN 4	0.075191	0.482776	0.726055	0.456671	**0.918071**	0.999010
FRCNN 5	0.005021	0.104581	0.615217	0.005791	0.108266	0.602770
FRCNN 6	0.018885	0.443814	0.746535	0.057485	0.632313	0.880226
FRCNN 7	0.037759	0.463065	0.763737	0.081005	0.677328	0.859757
FRCNN 8	0.061688	0.475882	0.774254	0.357120	0.888888	0.987505

Table 5.6: Recall AR100 at small, medium, and large plane detection using validation and training sets.

Models	Test Set			Training Set		
	Small	**Medium**	**Large**	**Small**	**Medium**	**Large**
SSD 1	0.548610	0.825074	0.859375	0.485115	0.794125	0.880315
SSD 2	0.509012	0.801180	**0.868750**	0.457498	0.750881	0.860630
SSD 3	**0.581592**	**0.840708**	0.865625	**0.518855**	0.800646	0.886614
SSD 4	0.215724	0.628024	0.778125	0.224401	0.654113	0.873228
SSD 5	0.269895	0.666667	0.796875	0.388820	0.793361	0.940157
SSD 6	0.275072	0.670501	0.853125	0.445098	0.840071	0.952756
SSD 7	0.268360	0.672271	0.818750	0.406699	0.813337	0.970079
FRCNN 1	0.118121	0.574926	0.803125	0.430080	0.919859	0.997638
FRCNN 2	0.117641	0.571976	0.818750	0.433474	0.918860	0.998425
FRCNN 3	0.135858	0.587316	0.821875	0.495253	0.926557	**1.000000**
FRCNN 4	0.128380	0.580826	0.815625	0.493323	**0.927203**	0.999213
FRCNN 5	0.007766	0.187021	0.706250	0.012955	0.184959	0.725984
FRCNN 6	0.045254	0.535398	0.790625	0.077130	0.674559	0.919685
FRCNN 7	0.066539	0.560767	0.828125	0.105590	0.714689	0.890551
FRCNN 8	0.117737	0.569912	0.818750	0.410448	0.908989	0.992126

Table 5.7: F1 Score at small, medium, and large plane and non-plane detections of validation and training sets.

Models	Test Set			Training Set			Category
	Small	**Medium**	**Large**	**Small**	**Medium**	**Large**	
SSD 1	0.494200	0.791703	0.836237	0.418633	0.763874	0.857297	**Plane**
SSD 2	0.450364	0.770476	0.841214	0.39047	0.717907	0.842364	
SSD 3	**0.523351**	**0.811908**	**0.848418**	0.455451	0.772755	0.86946	
SSD 4	0.132332	0.562713	0.752092	0.141303	0.615121	0.856158	
SSD 5	0.145255	0.613755	0.750075	0.299231	0.762332	0.928254	
SSD 6	0.201119	0.614824	0.679849	0.364955	0.814164	0.756182	
SSD 7	0.188073	0.62485	0.797910	0.328369	0.788366	0.962672	
FRCNN 1	0.081577	0.532789	0.770695	0.40563	0.91376	0.996649	
FRCNN 2	0.081796	0.522049	0.755442	0.407771	0.913195	0.997676	
FRCNN 3	0.101288	0.534235	0.77524	**0.477398**	0.921291	**1**	
FRCNN 4	0.094837	0.527282	0.768238	0.474290	**0.922614**	0.999111	
FRCNN 5	0.006099	0.134148	0.657598	0.008004	0.136583	0.658664	
FRCNN 6	0.026644	0.485323	0.767948	0.065874	0.652753	0.899523	
FRCNN 7	0.048178	0.507254	0.794629	0.091678	0.695507	0.874883	
FRCNN 8	0.080958	0.518670	0.795881	0.381931	0.898826	0.989810	
SSD 1	0.474478	0.655170	0.848487	0.505699	0.611366	0.820589	**All Categories**
SSD 2	0.481325	0.641780	**0.860632**	0.500955	0.623688	0.809393	
SSD 3	**0.515888**	**0.699845**	0.836334	0.536364	0.653116	0.848861	
SSD 4	0.208008	0.317363	0.692699	0.236788	0.446003	0.812984	
SSD 5	0.249193	0.440008	0.678940	0.550408	0.748117	0.892054	
SSD 6	0.271748	0.484961	0.682768	0.619237	0.831988	0.865393	
SSD 7	0.278008	0.503159	0.763287	0.630422	0.816116	0.943446	
FRCNN 1	0.087489	0.412111	0.644948	0.656973	**0.920125**	0.971509	
FRCNN 2	0.089568	0.382454	0.604870	0.65181	0.902235	0.984829	
FRCNN 3	0.114557	0.313047	0.654398	0.666321	0.89976	0.941191	
FRCNN 4	0.103609	0.348151	0.657446	**0.669295**	0.907782	**0.98694**	
FRCNN 5	0.008478	0.058753	0.412946	0.011144	0.051451	0.349663	
FRCNN 6	0.047534	0.271688	0.648217	0.095485	0.382360	0.736565	
FRCNN 7	0.088822	0.427192	0.681150	0.210579	0.562667	0.825599	
FRCNN 8	0.096270	0.320773	0.624790	0.6178492	0.898167	0.983123	

Following the calculation of these performance metrics with the values obtained, it is noticed that SSD 3 model presents the highest metric values except with metric6 and metric12 where SSD 2 overpassed (Table 5.3). On the other hand, Faster_R-CNN 5 gave the lowest metric results regarding 6 metrics, from metric1 to metric4, metric6 and metric12. All the models gave their highest metric value at average recall for large

object of 100 maximum large object detections at IoU from 0.50 to 0.95 (metric12) and their lowest at AR given one detection per image (metric7) at IoU from 0.50 to 0.95 (Table 5.3). Regarding training dataset (Table 5.4), the highest metric values are going to Faster_R-CNN models, except at average precision for all object scales of 100 maximum large object detections at IoU of 0.50 (metric2) where SSD 7 outperformed. The highest metric values were obtained at (metric12) for all the models, except SSD 5 and 7 their highest at metric2. However, all models presented their lowest metric values at metric7 on training dataset, except the Faster_R-CNN 5 to 7 which present their lowest values at metric4. For mean Average Precision (mAP) and Average Recall given 100 detection per image (AR100) at small, medium and large plane detection using validation set, SSD 3 gave the best results, except for AR100 at large scale planes where SSD 2 gave the best result. And concerning training set, Faster_R-CNN models, specially Faster_R-CNN 3, overpassed SSD models, except for AR100 at small scale planes where SSD 3 out-performed (Table 5.5 and Table 5.6). In addition to that, in Table 5.7, we observe that F1 score values increased from small to large plane and non-plane detections for all models. This is because the test dataset contains many small and medium scale plane and non-plane objects than large scale plane and non-plane objects. That also means SSD and Faster_R-CNN models perform better on large scale objects than on small and medium objects since we had a large number of small and medium airplane and non-airplane objects in our training dataset and several different model configurations. SSD models produced better results by comparing to Faster_R-CNN models at small, medium, and large scale object detections on test set. The SSD 3 generally outperformed other models in terms of the F1 Score at plane and non-plane detections therefore it is the best model on test dataset (Table 5.7). However, Faster_R-CNN models over-passed SSD models on training set. This domination of Faster_R-CNN over SSD on training set explains the lowest total losses obtained from Faster_R-CNN models during the training process. And the high performance obtained on training set by Faster_R-CNN may be the result of overfitting some models such as Faster_R-CNN 1 to 4 including Faster_R-CNN 8.

During the model evaluations, 20 images were visualized from each model. Below figures, from Figure 5.3 to 5.12, are some visualized evaluation images of SSD 3 and

Faster_R-CNN 4 models. And more visual results on test set from all models are shown in appendix A.

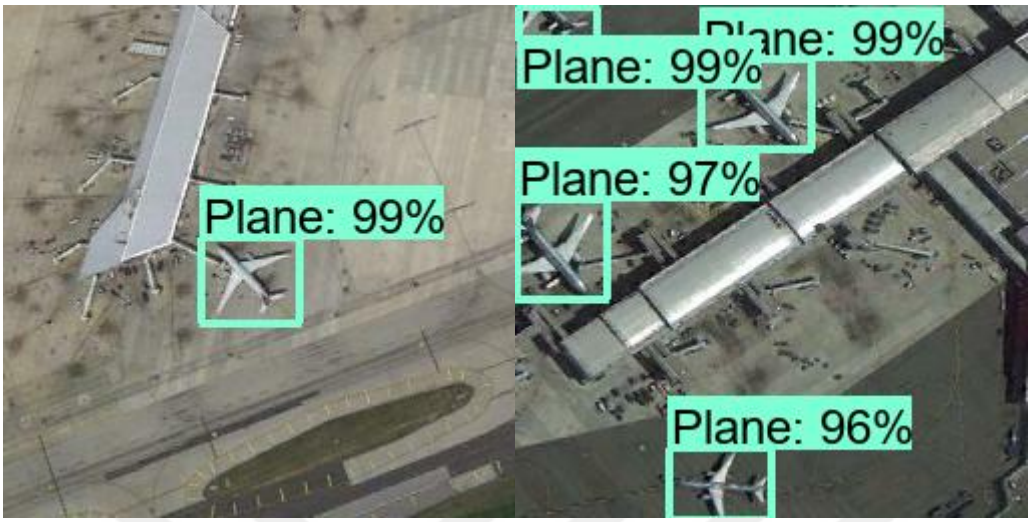

Figure 5.3: Sample 1 of visualized evaluation images of test dataset of SSD 3.

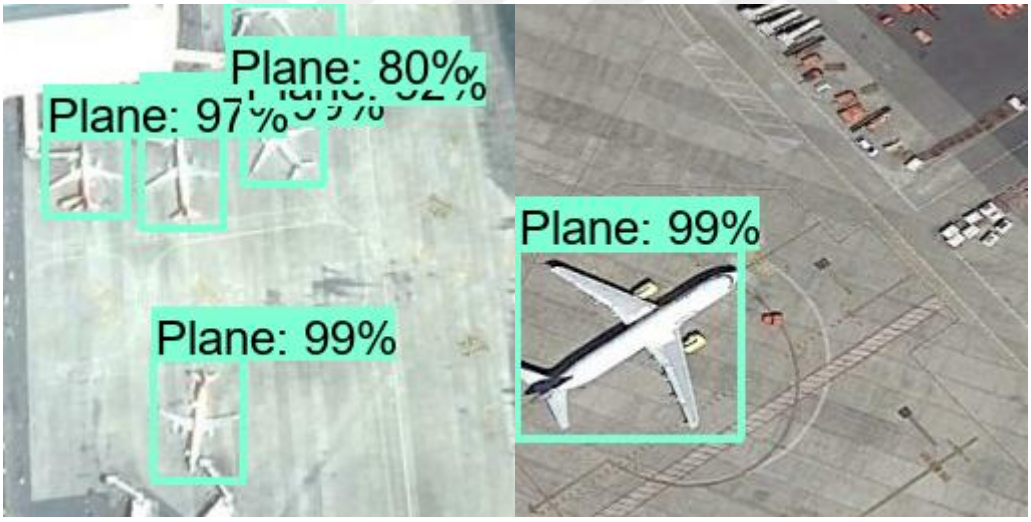

Figure 5.4: Sample 2 of visualized evaluation images of test dataset of SSD 3.

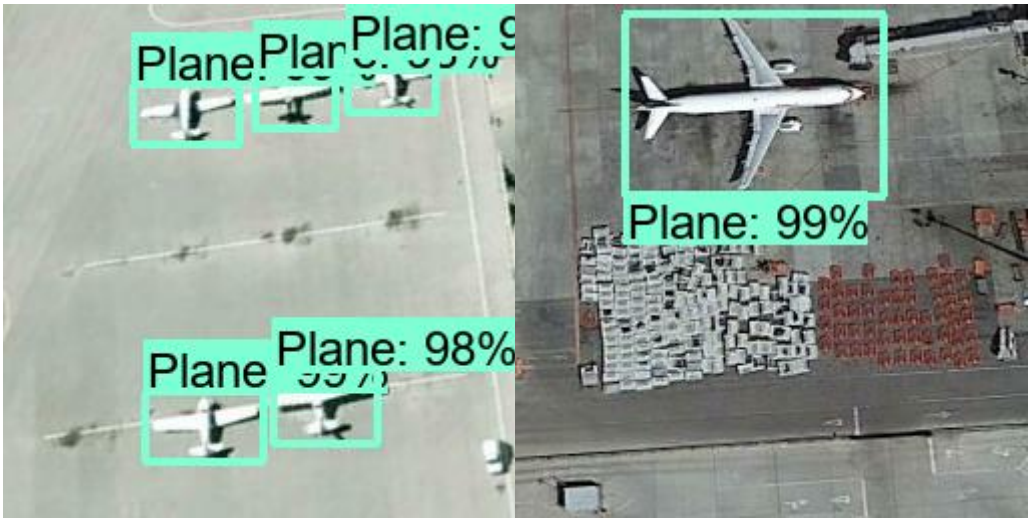

Figure 5.5: Sample 3 of visualized evaluation images of test dataset of SSD 3.

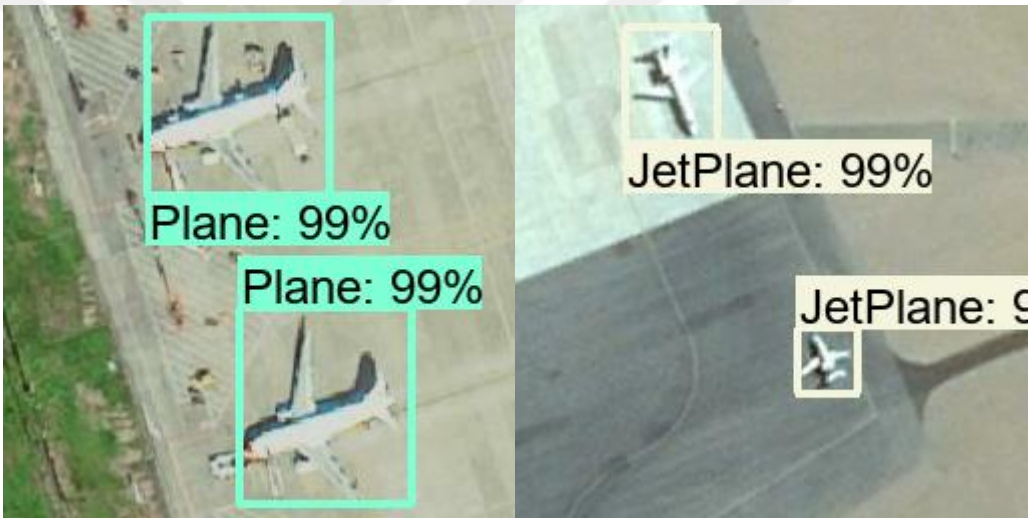

Figure 5.6: Sample 4 of visualized evaluation images of test dataset of SSD 3.

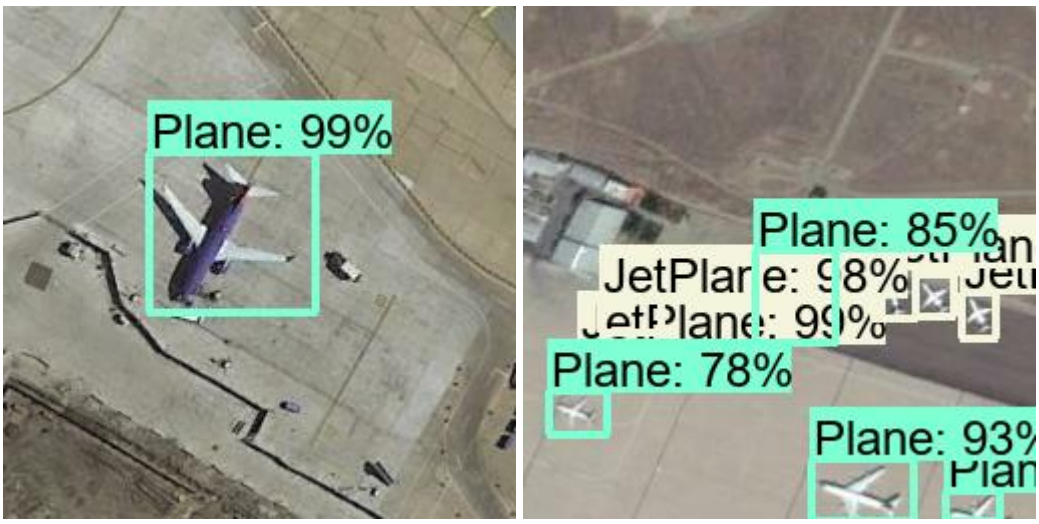

Figure 5.7: Sample 5 of visualized evaluation images of test dataset of SSD 3.

Figure 5.8: Sample 6 of visualized evaluation images of test dataset of SSD 3.

Figure 5.9: Sample 7 of visualized evaluation images of test dataset of SSD 3.

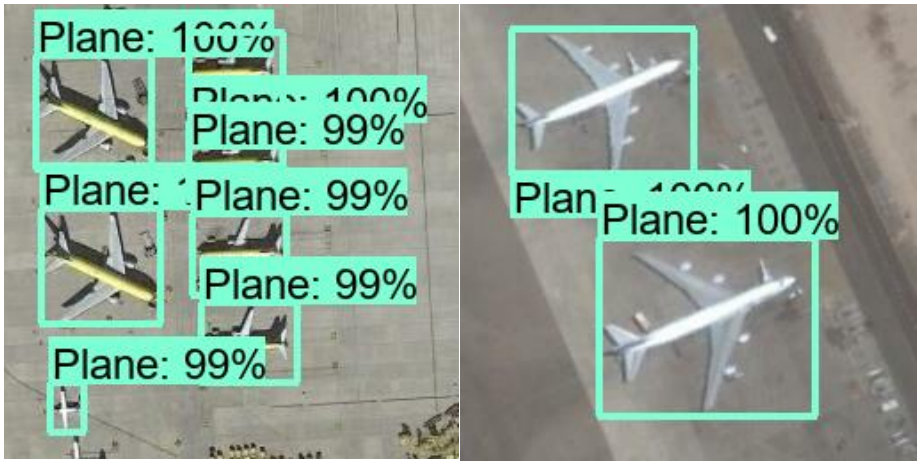

Figure 5.10: Sample 1 of visualized evaluation images of training dataset of Faster_R-CNN 4.

Figure 5.11: Sample 2 of visualized evaluation images of training dataset of Faster_R-CNN 4.

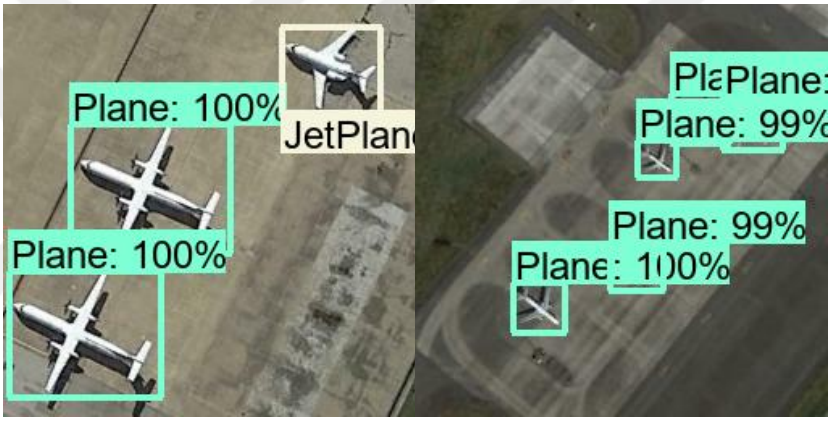

Figure 5.12: Sample 3 of visualized evaluation images of training dataset of Faster_R-CNN 4.

Figure 5.13: Sample 4 of visualized evaluation images of training dataset of Faster_R-CNN 4.

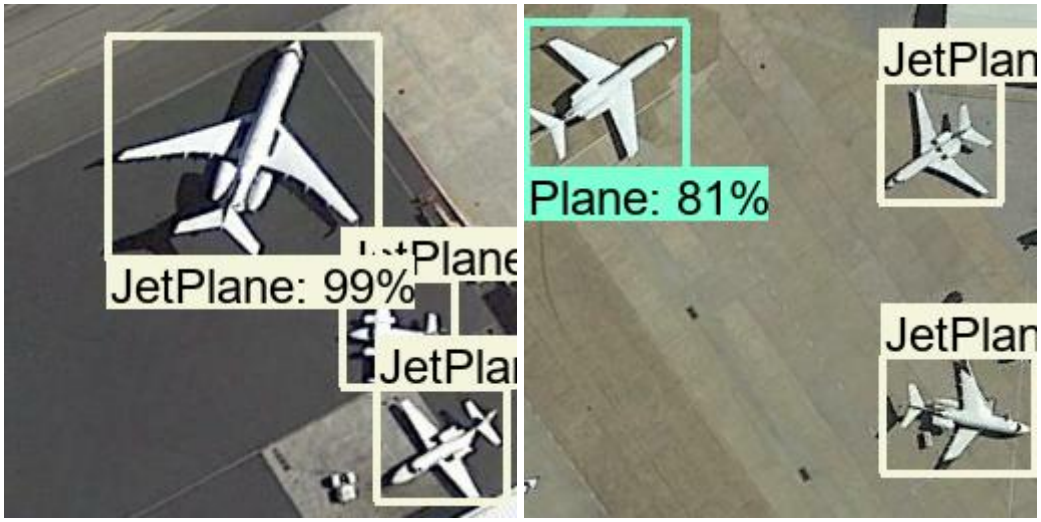

Figure 5.14: Sample 5 of visualized evaluation images of training dataset of Faster_R-CNN 4.

From these visualized evaluation images, we can see that our models can detect objects of different spatial and spectral resolutions.

6. CONCLUSIONS AND FUTURE WORKS

Our concern in this thesis was to detect airplanes using different deep learning techniques and very high-resolution satellite images and to analyze the effet of hyperparameters for a better future object detection. SSD and Faster R-CNN models were used for airplane detection in very high-resolution satellite images obtained from various sources; WPU-RESISC45 dataset, WHURS19 dataset, Aerial Image Dataset (AID), and ITU-CSCRS dataset. From the obtained metric values and the visual results, we retain in general from test dataset that Incepttion v2 made our training faster than ResNet_50, but ResNet_50 models presented the best airplane detection. We have seen also that L1_regularizer models learnt faster and better than L2_regularizer, however L2_regularizer models presented a better performance during the airplane detection in test dataset. The regularizers helped to reduce overfitting from Faster R-CNN by setting specific weight values. For score convertor, softmax function generated the best output compared to sigmoid function in our airplane detection. Also, we observed from our trainings that a low learning rate lead to a slow training compared to a high learning rate. During the training, when we reduce the learning rate in time, we observed better metric values for the Faster R-CNN models. A deep network can give a good performance, however it has a slow training and detection times. A large number of training steps may lead to a good performance, however for a good performance a large number of training steps is not essential. Also, a large batch size slows down the training speed in our case. RELU performs better on small scale object than RELU_6. Momentum optimizer leads to a better learning compared to RMSProp optimizer, however RMSProp optimizer is good for airplane detection. DL training from scratch using this SSD model can take long or short time depending on the configuration. The SSD models outperformed Faster R-CNN models on small scale airplane objects detection. The Faster R-CNN model with batch size of 2 gave low detection values. The data labeling needs time. The environment configuration is a complex task because since each time there are updates to the system software like

TensorFlow, CUDA, CUDNN, etc. So, compatibility between them fails sometimes if we update one and ignore other software updates. Our work helps to reply to some open questions such:

. transfer pre-trained network to other images, we apply two different datasets to the same model, and it works;

. the complexity of RS images (a problem in high spatial resolution, size differences, colors, locations, and rotations in a single scene);

. training SSD model with images of different sizes;

. training Faster R-CNN with batch size of 2.

The model configuration affects the training speed, the loss, and the detection result. The target localization, detection, and classification depend not only on the model but also on the dataset.

The future of deep learning in the field of remote sensing is open to many challenges such as:

. training with few images: the number of training samples even with the increased number of remote sensing data acquisition devices, access to most of the data is still not open;

. reducing training and test dataset labeling time;

. depth of the DL model (many DL systems have a large number of parameters, and require a significant amount of training data);

. creating an efficient network capable to be train and evaluated on images of different resolutions and type of bands.

In our experiments, we observed that some Faster R-CNN models performed better on training set than the test dataset. That may be caused by overfitting. So to solve this issue, we may use images of same size larger than or equal to 600x600 pixels and/or try to use different values for regularizer weight than what we have used in this thesis in the future studies.

REFERENCES

[1] Cheng, G.; Han, J. (2016), A survey on object detection in optical remote sensing images. ISPRS J. of Photogramm. and Remote Sens. 2016, 117, 11-28, DOI: 10.1016/j.isprsjprs.2016.03.014.

[2] L. ZHANG, L. ZHANG, BODU (2016): 'Deep Learning for Remote Sensing Data ', IEEE Geoscience and remote sensing 40 magazine June 2016, Digital Object Identifier 10.1109/MGRS.2016.2540798.

[3] Ying Li et al. (2018): 'Deep learning for remote sensing image classification: A survey', WIREs Data Mining Knowl Discov. 2018; 8:e1264. DOI: 10.1002/widm.1264.

[4] A. Zisserman and J. Sivic (2003**)**: 'Video Google: A Text Retrieval Approach to Object Matching in Videos', in Proc. IEEE Int. Conf. Comput. Vis., pp1470–1477.

[5] A. Cheriyadat (2014): 'Unsupervised Feature Learning for Aerial Scene Classification', IEEE Remote Sensing, vol. 52, pp439–451, **DOI:** 10.1109/TGRS.2013.2241444 .

[6] S. Xiang et al. (2014): 'Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks', IEEE Geosci. Remote Sens. Lett., vol. 11, no. 10, pp1797–1801.

[7] S. Xiang et al. (2015): 'Aircraft Detection by Deep Convolution Neural Networks', IPSJ Transactions on Computer Vision and Application Vol.7 10-17, DOI: 10.2197/ipsjtcva.7.10.

[8] W. Diao et al. (2015): 'Object Recognition in Remote Sensing Images Using Sparse Deep Belief Networks', Remote Sens. Lett. 6(10), 745–754.

[9] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. in Comput Vis ECCV, Amsterdam, Netherlands, 2016, 21-37.

[10] Ren, S.; He, K. M.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. in Proc. 28th Int. Conf. Comput. Vis., Montreal, QC, Canada, 2015, 91-99.

[11] Arthur OUAKNINE (2018), "Review of Deep Learning Algorithms for Object Detection",https://medium.com/zylapp/review-of-deep-learning-algorithms for-object- detection-c1f3d437b852

[12] Ugur Alganci, Mehmet Soydas and Elif Sertel (2020)," Comparative Research on Deep Learning Approaches for Airplane Detection from Very High-Resolution Satellite Images", Remote Sens. 2020, 12, 458; doi:10.3390/rs12030458.

[13] Redmon J., Farhadi A.,(2018): "YOLOv3: An Incremental Improvement", University of Washington, https://pjreddie.com/media/files/papers/YOLOv3.pdf.

[14] Common Objects in Context Challenge, http://cocodataset.org/#detection-eval.

[15] Wu Zhihuan et al. (2018): 'Rapid Target Detection in High Resolution Remote Sensing Images Using YOLO Model', DOI:10.5194/isprs-archives-XLII-3-1915-2018.

[16] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. (2014): Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on [16N]Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 580–587. [Google Scholar].

[17] Girshick, R. (2015): R. Fast R-CNN. in Proc. IEEE Int. Conf. Comput. Vis., Santiago, Chile, 2015, 1440–1448.

[18] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. (2016): A. You only look once: Unified, real-time object detection. in Proc IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Seattle, WA, USA, 2016, 779-788.

[19] Shiqi Chen, Ronghui Zhan and Jun Zhang (2018): 'Geospatial Object Detection in Remote Sensing Imagery Based on Multiscale Single-Shot Detector with Activated Semantics', Science and Technology on Automatic Target Recognition Laboratory, National University of Defense Technology, Changsha 410073, China, Remote Sens. 2018, 10(6), 820; https://doi.org/10.3390/rs10060820.

[20] S. Dorling and M. Gardner (1998): 'Artificial Neural Networks (The Multilayer Perceptron): A Review of Applications in The Atmospheric Sciences', Atmos. Environ., vol. 32, no. 14-15, pp2627–2636, Aug.

[21] G. Wu et al. (2017): 'Hyperspectral Image Classification Using Deep Pixel-Pair Features', IEEE Trans. Geosci. Remote Sens., vol. 55, no. 2, pp844–853.

[22] C. Gatta et al. (2016): 'Unsupervised Deep Feature Extraction for Remote Sensing Image Classification', IEEE Trans. Geosci. Remote Sens., vol. 54, no. 3, pp1349–1362.

[23] P. Ghamisi et al. (2017): 'Fully Conv-Deconv Network for Unsupervised Spectral-Spatial Feature Extraction of Hyperspectral Imagery Via Residual Learning' DOI: 10.1109/IGARSS.2017.8128169.

[24] P. Ghamisi et al. (2017): 'Unsupervised Spectral-Spatial Feature Learning Via Deep Residual Conv-Deconv Network for Hyperspectral Image Classification', IEEE Trans. Geosci. Remote Sens. doi:10.1109/ TGRS.2017.2748160.

[25] G. Sheng et al. (2012): 'High-Resolution Satellite Scene Classification Using A Sparse Coding Based Multiple Feature Combination', International Journal of Remote Sensing, vol. 33, no. 8, pp2395–2412.

[26] Tao Chen et al. (2017): 'Unsupervised Feature Learning for Land-Use Scene Recognition', Institute for Infocomm Research, Nanyang Technological University, https://www.researchgate.net/publication/312185111, Article in

IEEE Transactions on Geoscience and Remote Sensing · January, DOI: 10.1109/TGRS.2016.2640186.

[27] F. Zhang et al. (2015): 'Saliency-Guided Unsupervised Feature Learning for Scene Classification', IEEE Trans. Geosci. Remote Sens., vol. 53, no. 4, pp2175-2184, April.

[28] D. Dai et al. (2011): 'Satellite Image Classification Via Two-Layer Sparse Coding with Biased Image Representation', IEEE Geoscience and Remote Sensing Letters, vol. 8, pp173-176, January.

[29] E. R. Keydel et al. (1996): 'MSTAR Extended Operating Conditions: A Tutorial', Proc. SPIE-Int. Soc. Opt. Eng., vol. 2757, 1996. doi: 10.1117/12.242059.

[30] F. Xu et al. (2016): 'Target classification using the deep convolutional networks for SAR images', IEEE Trans. Geosci. Remote Sens., vol. 54, no. 8, pp4806–4817.

[31] D. Morgan (2015): 'Deep Convolutional Neural Networks for ATR From SAR Imagery', Processing SPIE-Int. Soc. Opttical Eng., vol. 9475. doi: 10.1117/12.2176558.

[32] C. Kreucher et al. (2016): 'Modern Approaches in Deep Learning for SAR ATR', Processing SPIE-Int. Soc. Opttical Eng., vol. 9843. doi: 10.1117/12.2220290.

[33] S. A. Wagner (2016): 'SAR ATR by A Combination of Convolutional Neural Network and Support Vector Machines', IEEE Trans. Geosci. Remote Sens., vol. 52, no. 6, pp2861–2872.

[34] Cortes, C.; Vapnik, V. (1995): Support-Vector Networks. Mach. Learn. 1995, 20, 273–297. [Google Scholar] [CrossRef].

[35] J. Yang et al. (2015): 'Hierarchical Recognition System for Target Recognition from Sparse Representations', Math. Problems Eng., vol. 2015. doi: 10.1155/2015/527095.

[36] X. X. ZHU et al. (2017), 'Deep Learning in Remote Sensing: A Review', IEEE Geoscience and remote sensing magazine December 2017, DOI 10.1109/MGRS.2017.2762307, Date of publication: 27 December.

[37] Han, J.; Zhang, D.; Cheng, G.; Guo, L.; Ren, J. (2014); Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning. IEEE Trans. Geosci. Remote Sens. 2014, 53(6),3325-3337, DOI: 10.1109/TGRS.2014.2374218.

[38] John E. Et al. (2017): "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community,", J. of Applied Remote Sensing, 11(4), 042609 (2017). doi: 10.1117/1.JRS.11.042609.

[39] Shaoming Zhang, Ruize Wu, Kunyuan Xu, Jianmei Wang, and Weiwei Sun (2019) 'R-CNN-Based Ship Detection from High Resolution Remote Sensing Imagery', Remote Sens. 2019, 11(6), 631; https://doi.org/10.3390/rs11060631.

[40] Adam Van E.en (2018) ‘You Only Look Twice: Rapid Multi-Scale Object Detection In Satellite Imagery’, arXiv:1805.09512v1 [cs.CV] 24 May 2018.

[41] Xue Yang et al. (2018) ‘Automatic Ship Detection in Remote Sensing Images from Google Earth of Complex Scenes Based on Multiscale Rotation Dense Feature Pyramid Networks’, Remote Sens. 2018, 10, 132; doi:10.3390/rs10010132.

[42] Laila Bashmal et al. (2018) ‘Siamese-GAN: Learning Invariant Representations for Aerial Vehicle Image Categorization’, Remote Sens. 2018, 10, 351; doi:10.3390/rs10020351.

[43] Yuhang Zhang et al. (2018) ‘Aircraft Type Recognition in Remote Sensing Images Based on Feature Learning with Conditional Generative Adversarial Networks’, Remote Sens. 2018, 10, 1123; doi:10.3390/rs10071123.

[44] Joseph Redmon, Ali Farhadi (2016) ‘YOLO9000: Better, Faster, Stronger’, https://arxiv.org/abs/1612.08242.

[45] T. Nathan Mundhenk et al.(2016) ‘A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning’, 2016 (arxiv.org/abs/1609.04453).

[46] Moein Zalpour et al.(2019) ‘A New Approach for Bright Circular Oil Tanks Detection in High Resolution Optical Imagery’, 4th International congress on engineering technology and applied sciences, New Zealand-Auckland, June, 2019.

[47] Muhammad Jaleed Khan et al.(2017) ‘Automatic Target Detection in Satellite Images using Deep Learning’, Article in Journal of Space Technology · July 2017, https://www.researchgate.net/publication/319313161.

[48] Zhou, P.; Cheng, G.; Liu, Z.; Bu, S.; Hu, X. **(2015)**Weakly supervised target detection in remote sensing images based on transferred deep features and negative bootstrapping. Multidim. Syst. Sign. Process. 2015, 27, 925-944, DOI: 10.1007/s11045-015-0370-3**.**

[49] Hu, G.; Yang, Z.; Han, J.; Huang, L.; Gong, J.; Xiong, N. (2018); Aircraft detection in remote sensing images based on saliency and convolution neural network. EURASIP J. Wirel. Commun. Netw. 2018, 26, 1-16, DOI: 10.1186/s13638-018-1022-8.

[50] Tang, T.; Zhou, S.; Deng, Z.; Zou, H.; Lei, L. (2017); Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining, Sensors 2017, 17(2), 1-17, DOI: 10.3390/s17020336.

[51] Gidaris, S.; Komodakis, N. Locnet (2016); Locnet: Improving localization accuracy for object detection. in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Las Vegas, NV, USA, 2016, 789-798.

[52] Everingham, M.; Gool, L. V.; Williams, C.K. I.; Winn, J.; Zisserman, A. (2010); The PASCAL visual object classes (VOC) challenge. Int. J. Comput. Vis. 2010, 88(2), 303-338, DOI:10.1007/s11263-009-0275-4.

[53] Polat, E.; Yildiz, C. (2012); Stationary aircraft detection from satellite images. IU-JEEE. 2012, 12(2), 1523-1528.

[54] Sun, H.; Sun, X.; Wang, H.; Li, Y.; Li, X. (2012); Automatic target detection in high-resolution remote sensing images using spatial sparse coding bag-of-words model. IEEE Trans. Geosci. Remote Sens. 2012, 9(1), 109-113, DOI: 10.1109/LGRS.2011.2161569.

[55] Atakan Körez and Necaattin Barışçı (2019); Object Detection with Low Capacity GPU Systems Using Improved Faster R-CNN. Appl. Sci. 2020, 10, 83; doi:10.3390/app10010083.

[56] Cheng, G.; Han, J.; Guo, L.; Qian, X.; Zhou, P.; You, X.; Hu, X. (2013); Object detection in remote sensing imagery using a discriminatively trained mixture model. ISPRS J. of Photogramm. and Remote Sens. 2013, 85, 32-43, DOI: 10.1016/j.isprsjprs.2013.08.001.

[57] Han, J.; Zhou, P.; Zhang, D.; Cheng, G.; Guo, L.; Liu, Z.; Bu, S.; Wu, J. (2014); Efficient, simultaneous detection of multi-class geospatial targets based on visual saliency modeling and discriminative learning of sparse coding. ISPRS J. of Photogramm. and Remote Sens. 2014, 89, 37-48, DOI: 10.1016/j.isprsjprs.2013.12.011.

[58] D. Yu and L. Deng (2015): 'Automatic Speech Recognition: a deep learning approach', Signals and Communication Technology, DOI 10.1007/978-1-4471-5779-3, pp. 57-317, © Springer-Verlag London.

[59] H.-C. Shin et al. (2016): 'Deep Convolutional Neural Networks for Computer aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning', IEEE T. M. Images, vol. 35, no. 5, pp 1285–1298, May.

[60] Yuichi Konishi et al. (2018), 'Detection of Target Persons Using Deep Learning and Training Data Generation for Tsukuba Challenge', Journal of Robotics and Mechatronics Vol.30 No.4.

[61] J. Suto et al. (2018): 'Efficiency investigation from shallow to deep neural network techniques in human activity recognition', Action editor: Lorenzo Jamone Jozsef Suto, Stefan Oniga, Department of Informatics Systems and Networks, accepted 22 November 2018, Available online 29 Novembe, Cognitive Systems Research 54 (2019) pp37–49.

[62] Y.Guoetal et al. (2016): 'Deep learning for visual understanding: A review', The Netherlands, College of Information Systems and Management, National University of Defense Technology, Changsha, China, Neuro computing187 pp27–48.

[63] Wudi Zhao et al. (2019): 'Hyperspectral Images Classification with Convolutional Neural Network and Textural Feature Using Limited Training Samples', Remote Sensing Letters, 10:5, 449-458, DOI: 10.1080/2150704X.2019.1569274.

[64] D. Yu and L. Deng (2015): 'Automatic Speech Recognition: a deep learning approach', Signals and Communication Technology, DOI 10.1007/978-1-4471-5779-3, pp. 290, © Springer-Verlag London.

[65] Di Wu et al. (2019): 'A deep model with combined losses for person re-identification', Guangxi Teachers Education University, Nanning, Guangxi 530001, China, Cognitive Systems Research 54 pp74–82, https://doi.org/10.1016/j.cogsys.2018.04.003.

[66] Christof Angermueller et al. (2016): 'Deep learning for computational biology', Molecular Systems Biology, Published online: July 29.

[67] Y. Bengio et al. (2016): 'Deep Learning', http://www.deeplearningbook.org, MIT Press.

[68] M. Zeiler (2014): 'Hierarchical Convolutional Deep Learning in Computer Vision', NY University.

[69] Hayit Greenspan et al. (2016): 'Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique', IEEE Transactions on Medical Imaging, VOL. 35, NO. 5, MAY.

[70] He, K.; Zhang, X.; Ren, S.; Sun, J. (2015): 'Deep residual learning for image recognition," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Boston, MA, USA, 2015, 1–9.

[71] Deng, J.; Dong, W.; Socher, R.; Li, L. J.; Li, K.; Li, F.F. (2009): Imagenet: a large-scale hierarchical image database. in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Miami, FL, USA, 2009, 248–255.

[72] Peng Zhao (2016): 'R for Deep Learning (I): Build Fully Connected Neural Network from Scratch', https://www.r-bloggers.com/r-for-deep-learning-i-build-fully-connected-neural-network-from-scratch/, Reached on 08/12/2019.

[73] M. Talo et al. (2019): 'Application of deep transfer learning for automated brain abnormality classification using MR images', School of Medicine, Faculty of Health and Medical Sciences, Taylor's University, 47500 Subang Jaya, , Cognitive Systems Research 54, pp176–188.

[**74] Jonathan Huang et al. (2017)**, "Speed/accuracy trade-offs for modern convolutional object detectors", arXiv:1611.10012v3 [cs.CV] 25 Apr 2017.

[**75] Szegedy, C.; Vanhoucke, C.; Ioffe, S.; Shlens, J.; Wojna, Z. (2016)**, Rethinking the inception architecture for computer vision. in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Seattle, WA, USA, 2016, 2818-2826.

[76] Bharath RajA (2018), "Simple Guide to the Versions of the Inception Network, https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the inception-network-7fc52b863202, reached on 09/11/2019.

[77]MATLAB,resnet50,"https://www.mathworks.com/help/deeplearning/ref/resnet50.html, reached on 09/11/2019.

[78] Qi Liao (2018), "A Secure End-to-End Cloud Computing Solution for Emergency Management with UAVs", DOI: 10.1109/GLOCOM.2018.8648094.

[79] Hu, F.; Xia, G.S.; Hu, J.; Zhang, L. (2015): Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. Remote Sens. 2015, 7(11), 14680-14707, DOI: 10.3390/rs71114680.

[80] XIA et al. (2017): 'AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification' Senior Member, IEEE Transactions on Geoscience and Remote Sensing, Vol. 55, No. 7.

[81] Gong Cheng et al. (2014), "Multi-class geospatial object detection and geographic image classification based on collection of part detectors", ISPRS Journal of Photogrammetry and Remote Sensing, 98: 119-132, 2014.

[82] Cheng, G.; Zhou, P.; Han, J. (2016), Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images. IEEE Trans. Geosci. Remote Sens. 2016, 54(12), 7405-7415, DOI: 10.1109/TGRS.2016.2601622.

[83] W. Shao et al. (2013), "A hierarchical scheme of multiple feature fusion for high-resolution satellite scene categorization," in Computer Vision Systems. Berlin, Germany: Springer, 2013, pp. 324–333.

[84] Cheng, G.; Han, J.; Lu, X. (2017), Remote sensing image scene classification: benchmark and state of the art. Proc. IEEE 2017, 105(10), 1865-1883.

[85]Y. Wen, "DataSet:WHU-RS19",http://www.escience.cn/people/yangwen/WHU-RS19.html,reached on 01.11.2019.

[86] Lin, T. Y.; Marie, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. (2014), "Microsoft COCO: Common Objects in Context. in Comput Vis ECCV, Zurich, Switzerland, 2014, 740-755.

[87] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean,J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. et al. (2016), TensorFlow: A System for Large-Scale Machine Learning. in OSDI, Savannah, GA, USA, 2016, 265-283, arXiv:1605.08695.

APPENDICES

APPENDIX A.1 : Visual Results

APPENDIX A.1

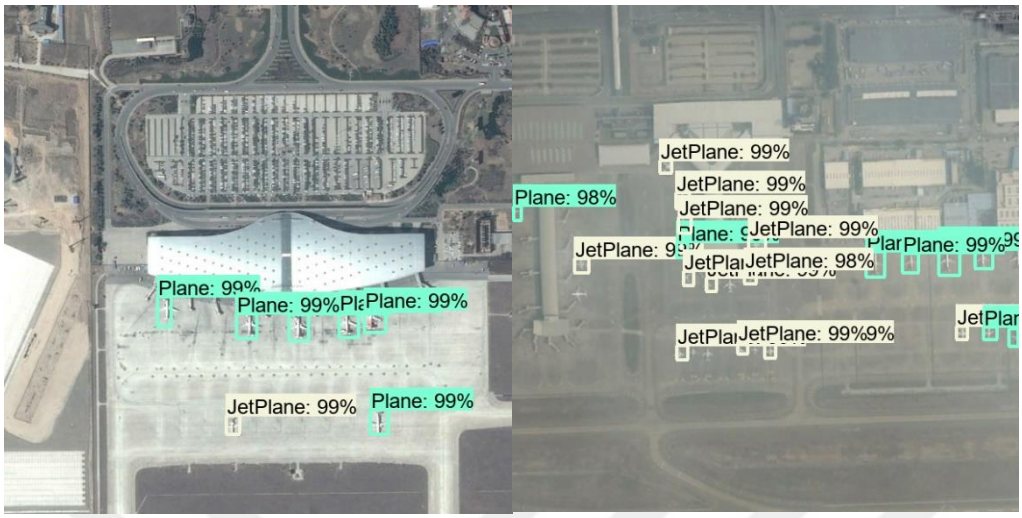

Figure A.1: Sample 1 of visualized evaluation images of test dataset of SSD 3.

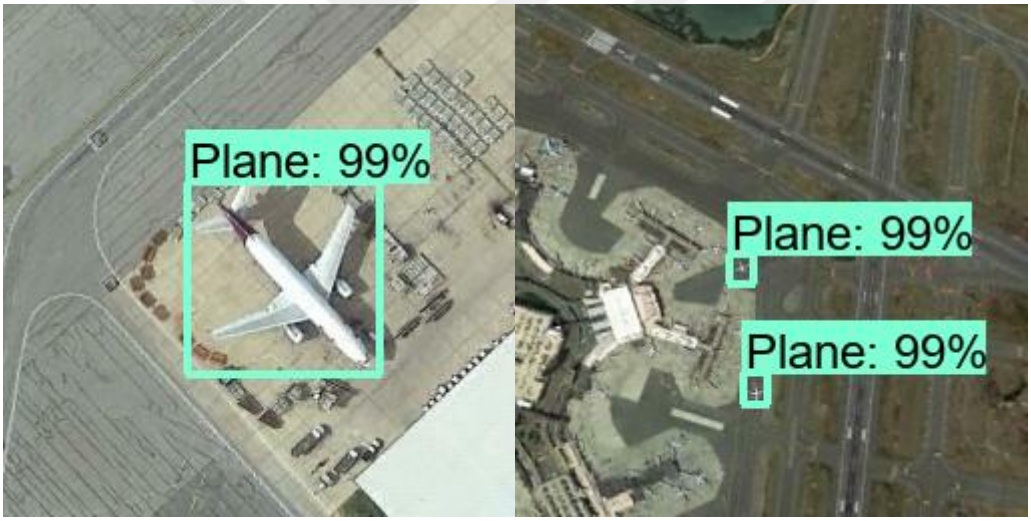

Figure A.2: Sample 2 of visualized evaluation images of test dataset of SSD 3.

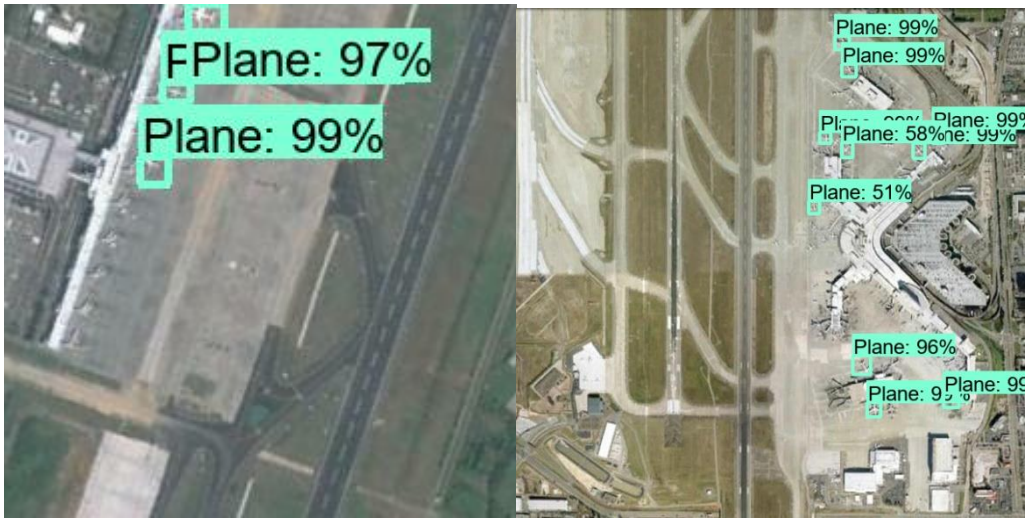

Figure A.3: Sample 3 of visualized evaluation images of test dataset of SSD 3.

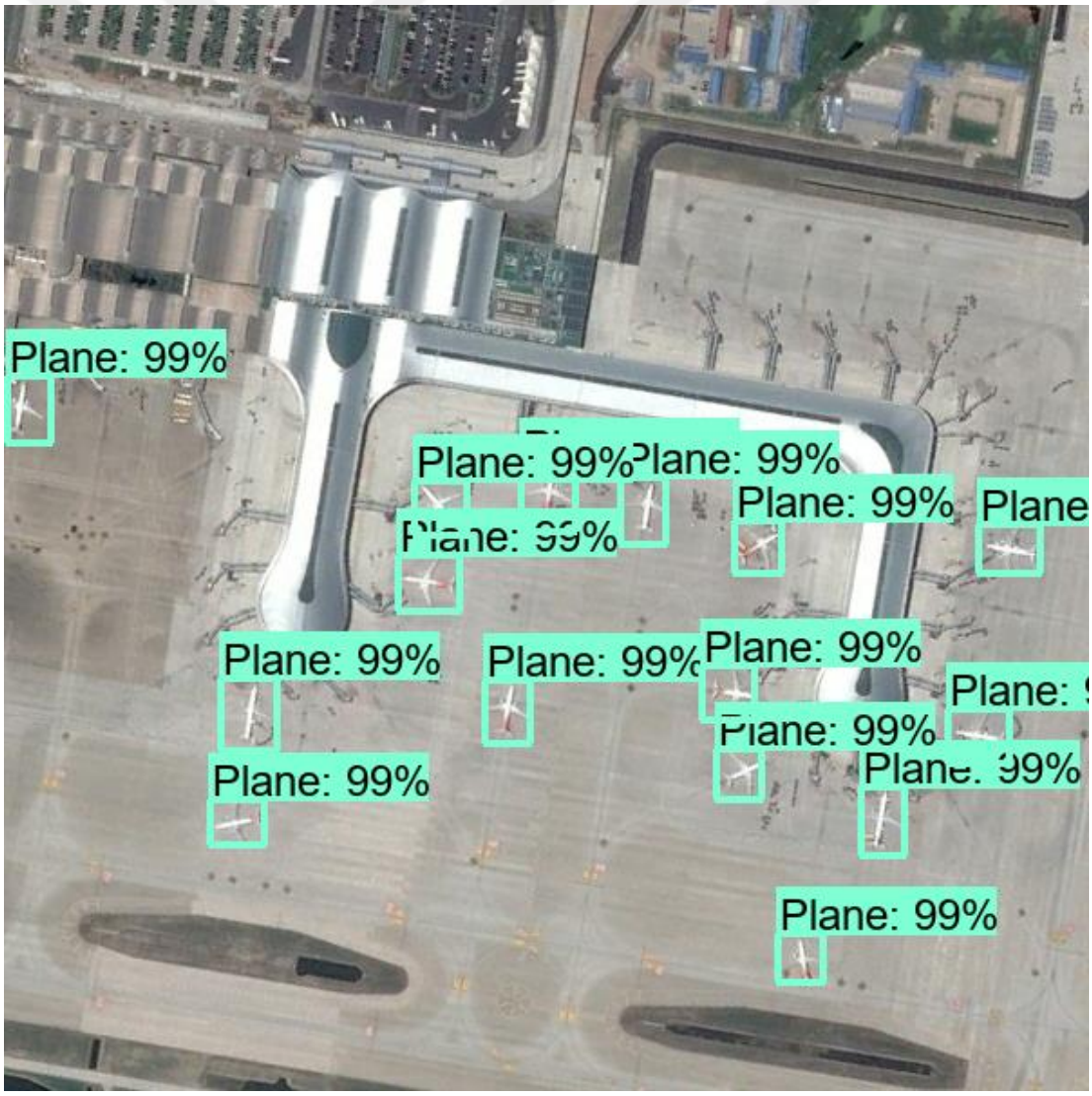

Figure A.4: Sample 4 of visualized evaluation images of test dataset of SSD 3.

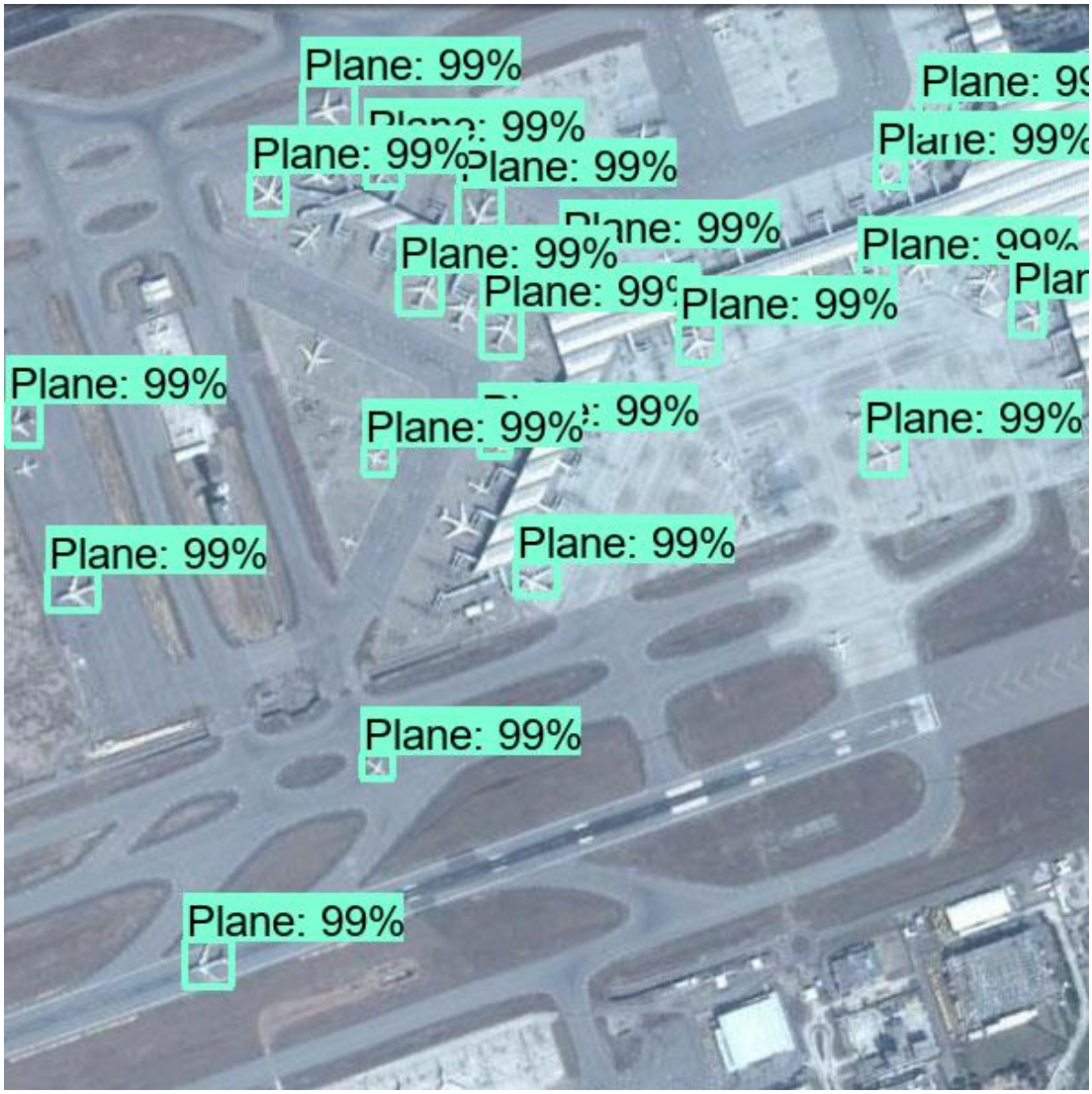

Figure A.5: Sample 5 of visualized evaluation images of test dataset of SSD 3.

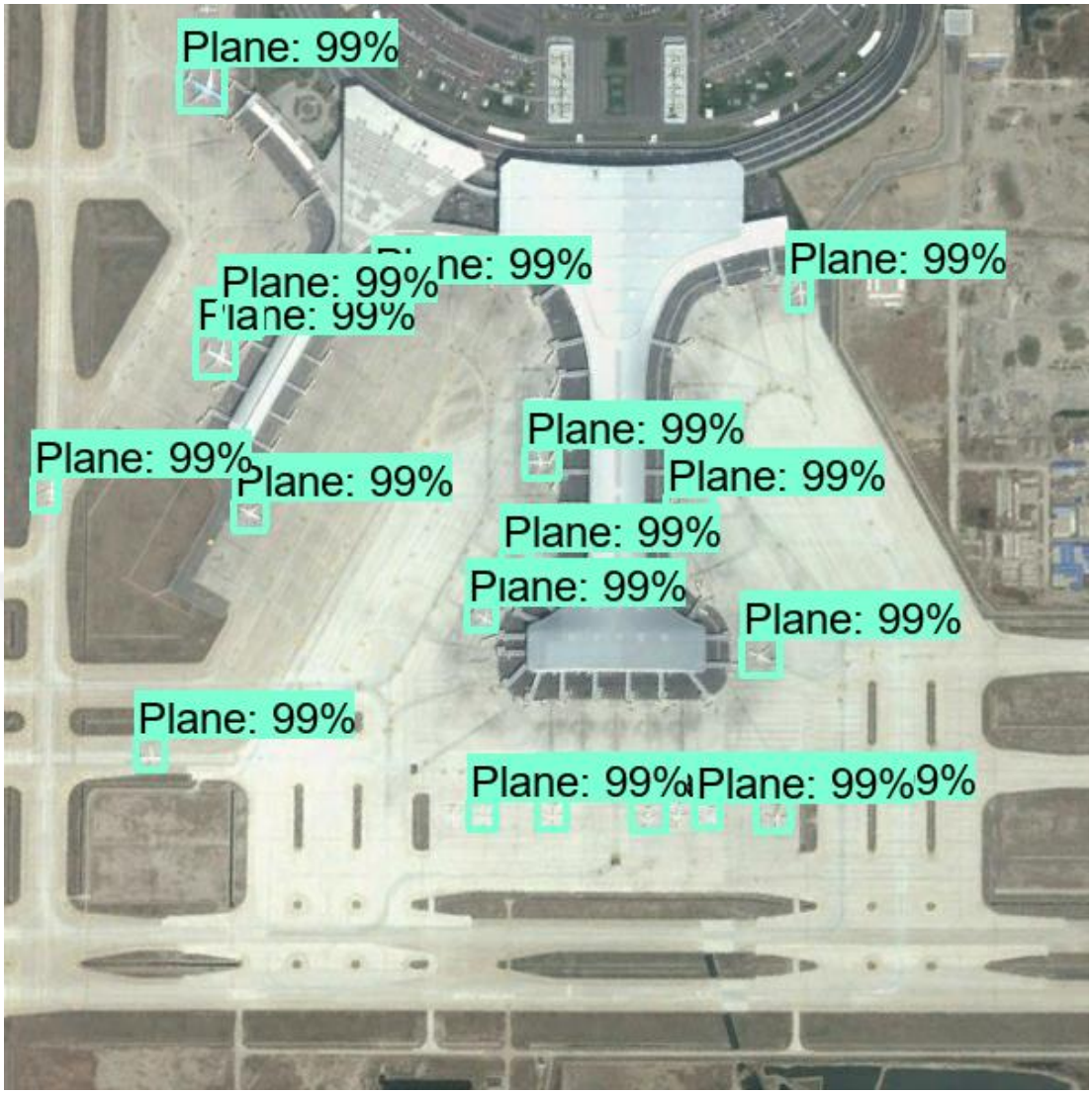

Figure A.6: Sample 6 of visualized evaluation images of test dataset of SSD 3.

Figure A.7: Sample 7 of visualized evaluation images of test dataset of SSD 3.

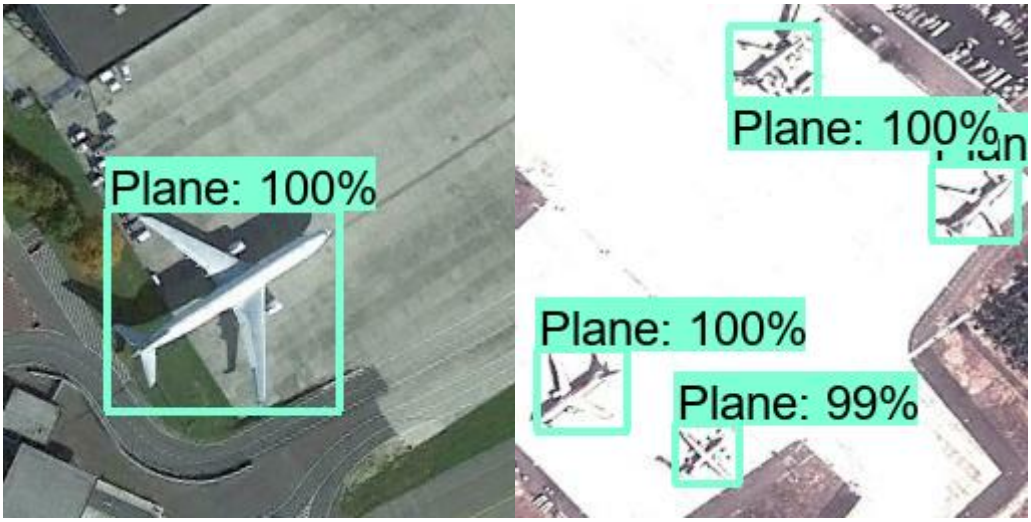

Figure A.8: Sample 1 of visualized evaluation images of training dataset of Faster_R-CNN 4.

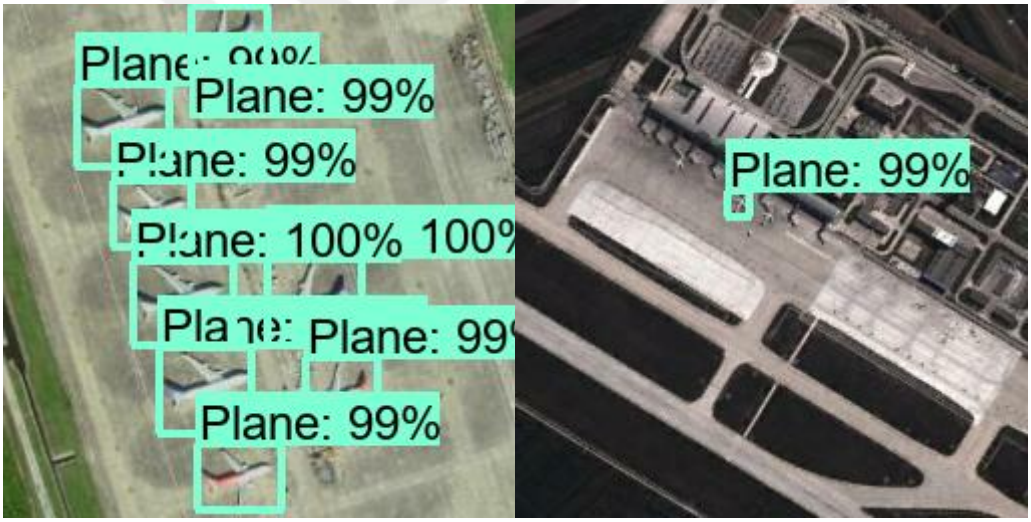

Figure A.9: Sample 2 of visualized evaluation images of training dataset of Faster_R-CNN 4.

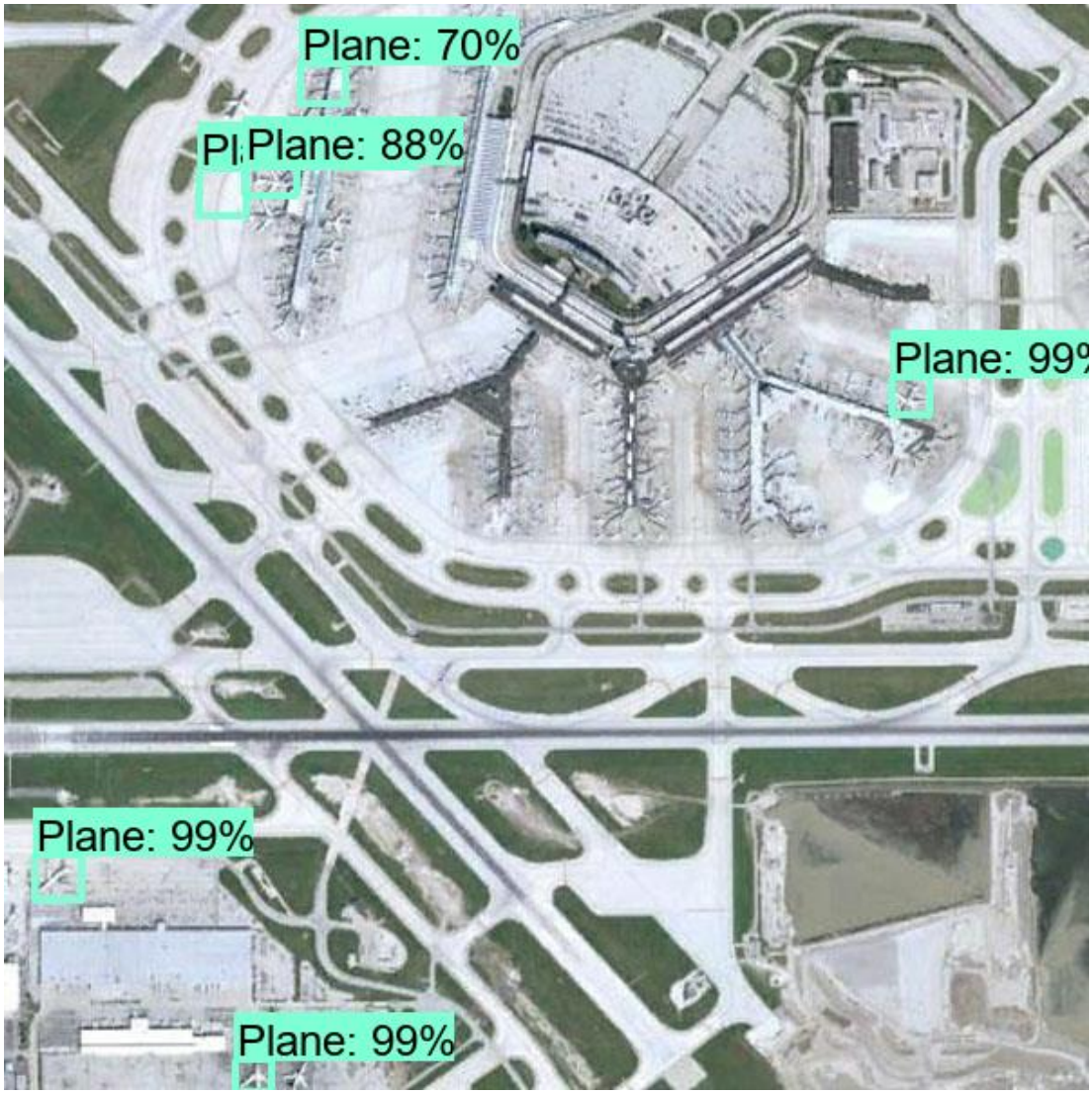

Figure A.10: Sample 2 of visualized evaluation images of training dataset of Faster_R-CNN 4.

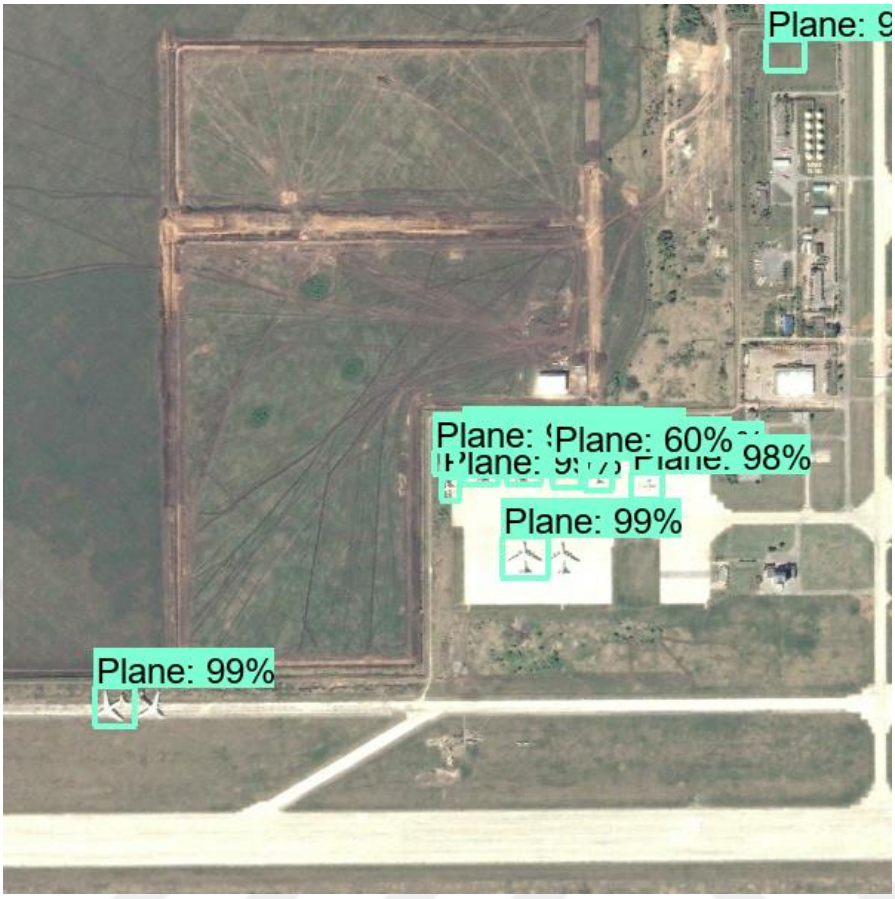

Figure A.11: Sample 3 of visualized evaluation images of training dataset of Faster_R-CNN 4.

Figure A.12: Sample 4 of visualized evaluation images of training dataset of Faster_R-CNN 4.

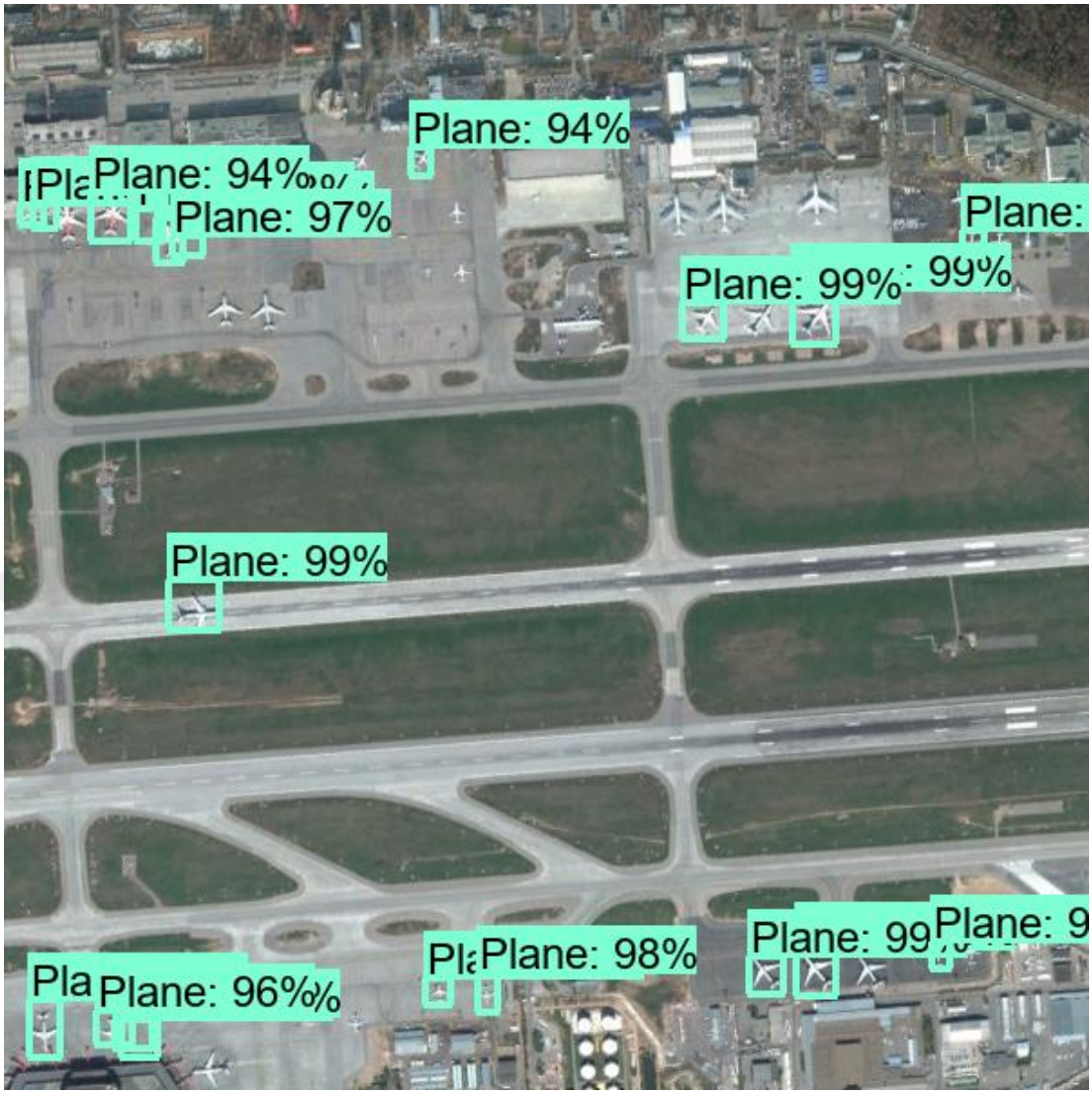

Figure A.13: Sample 5 of visualized evaluation images of training dataset of Faster_R-CNN 4.

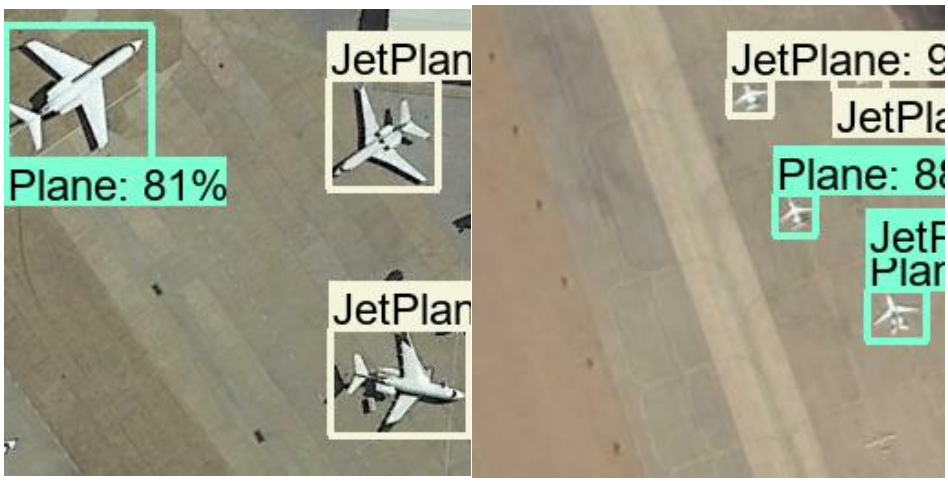

Figure A.14: Sample 5 of visualized evaluation images of training dataset of Faster_R-CNN 4.

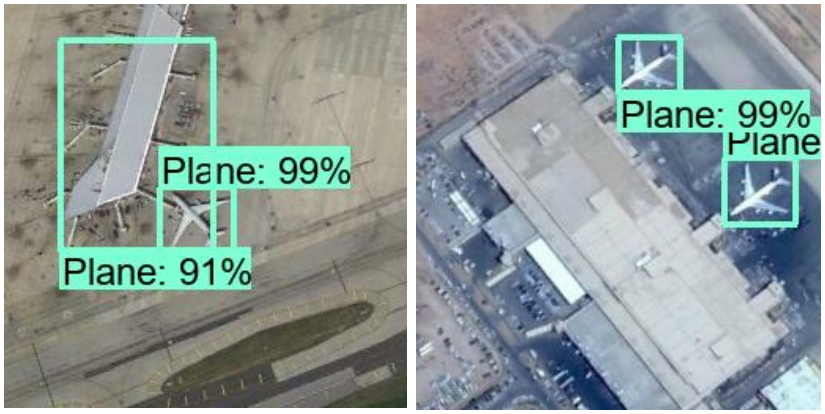

Figure A.15: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 1.

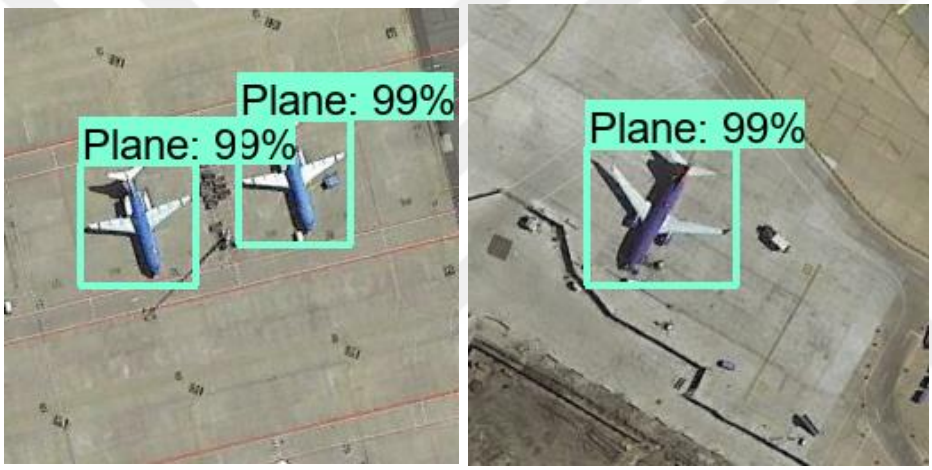

Figure A.16: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 1.

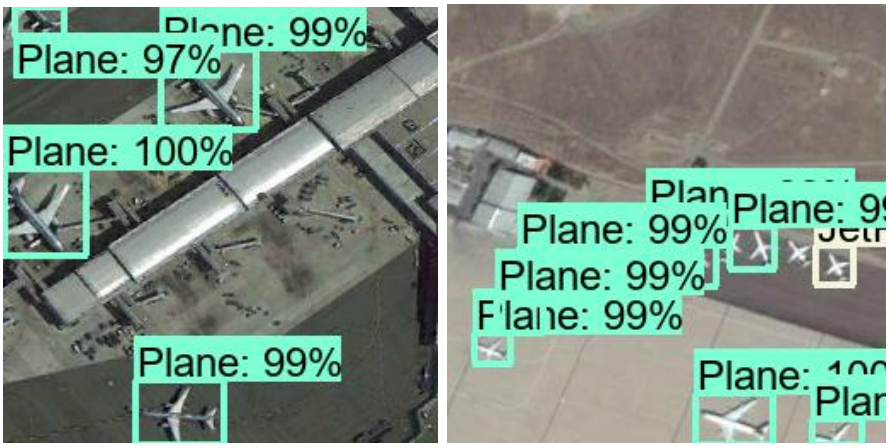

Figure A.17: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 1.

Figure A.18: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 1.

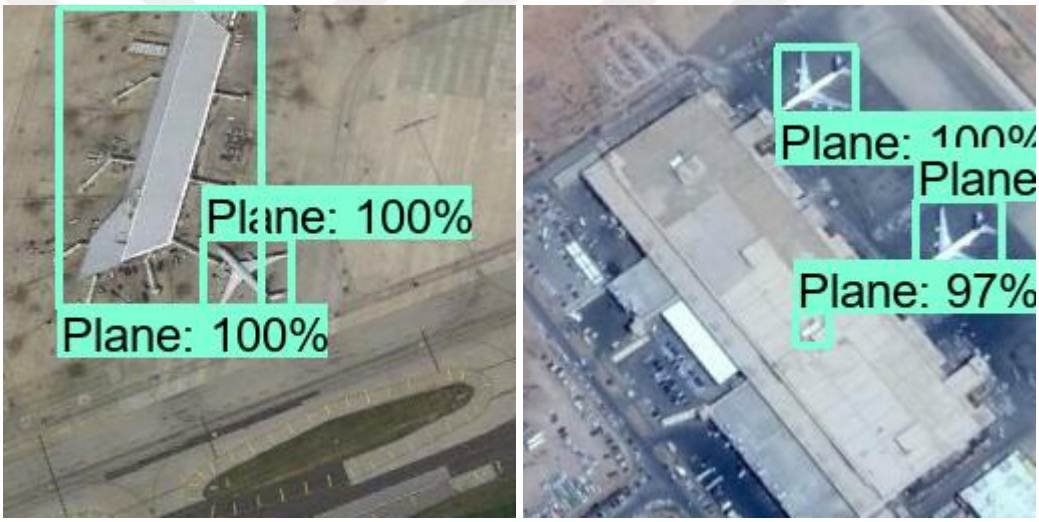

Figure A.19: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 2.

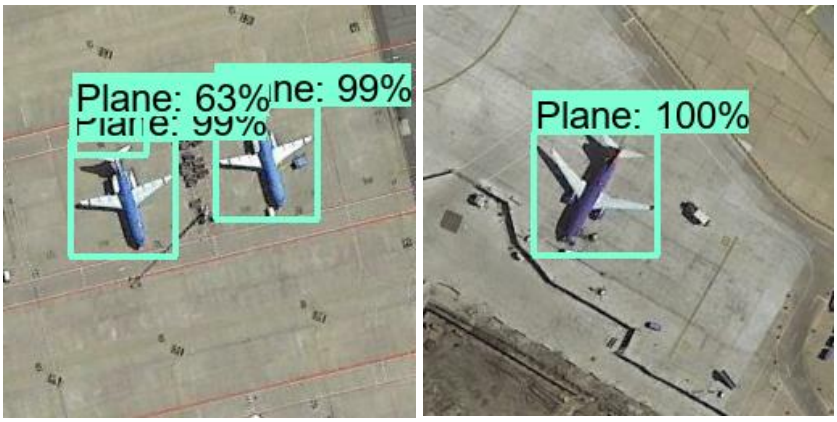

Figure A.20: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 2.

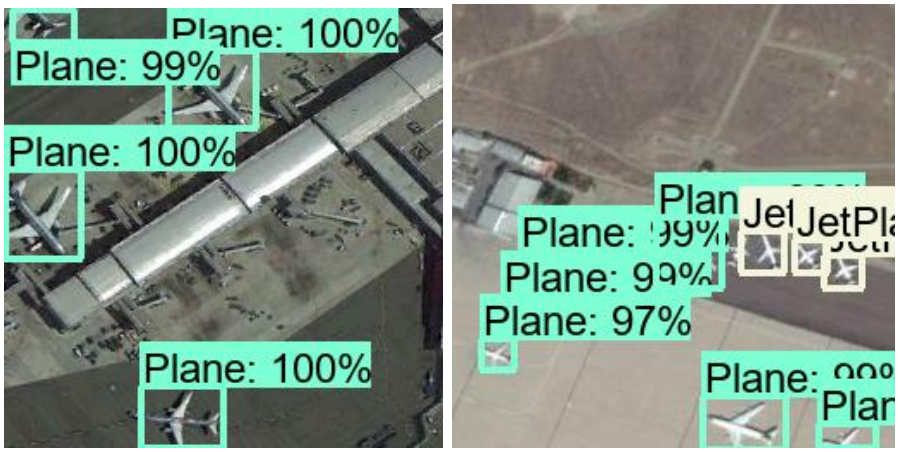

Figure A.21: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 2.

Figure A.22: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 2.

Figure A.23: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 3.

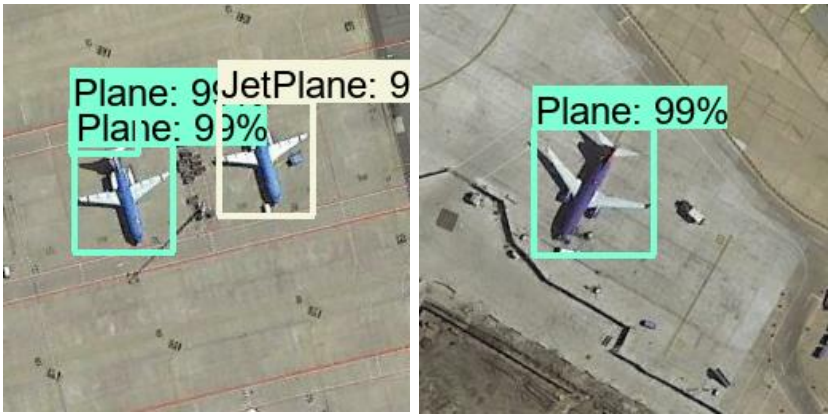

Figure A.24: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 3.

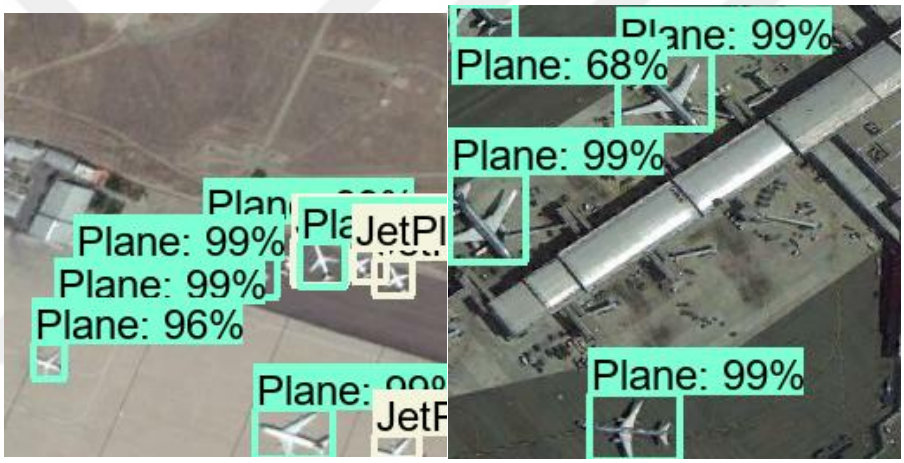

Figure A.25: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 3.

Figure A.26: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 3.

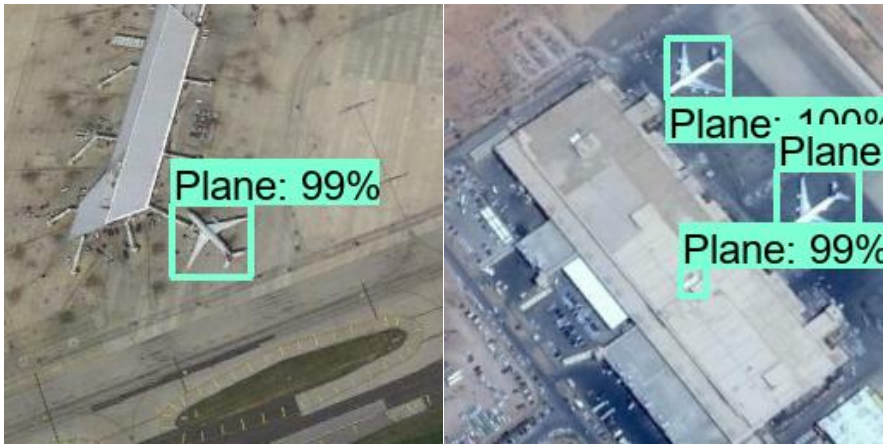

Figure A.27: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 4.

Figure A.28: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 4.

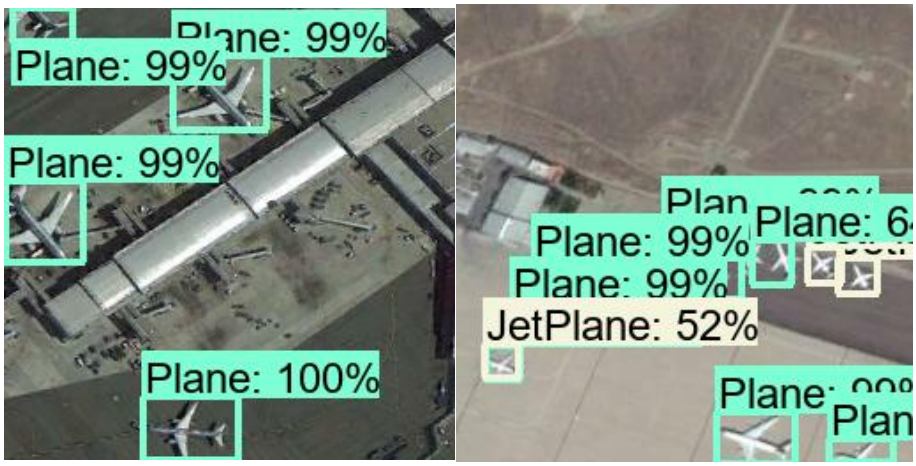

Figure A.29: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 4.

Figure A.30: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 4.

Figure A.31: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 5.

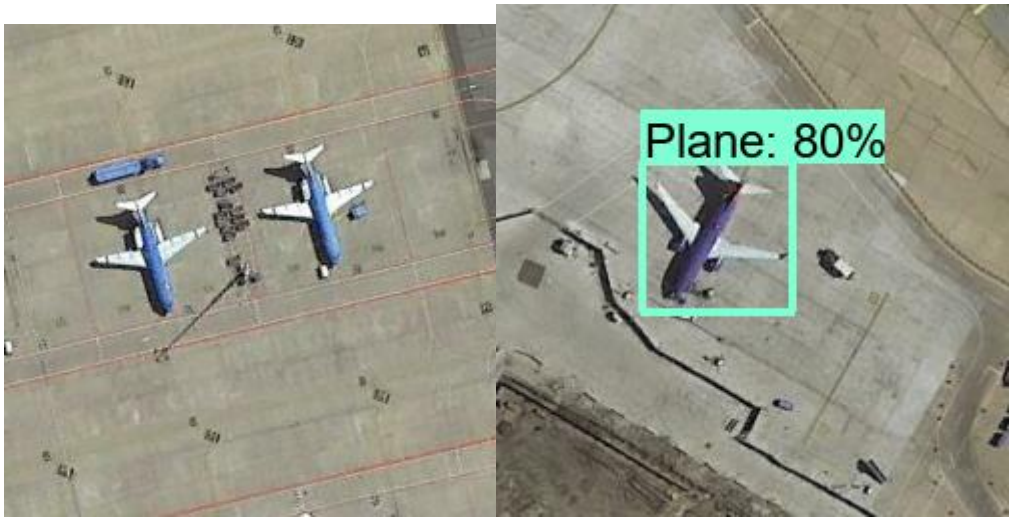

Figure A.32: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 5.

Figure A.33: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 5.

Figure A.34: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 5.

Figure A.35: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 6.

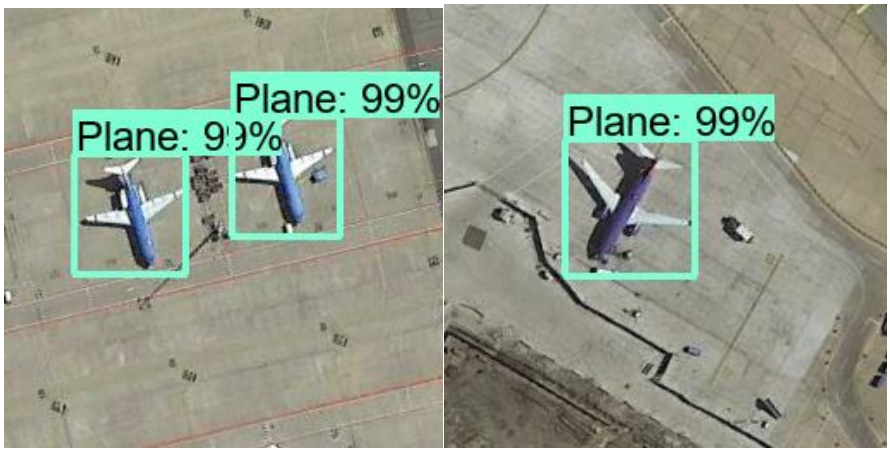

Figure A.36: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 6.

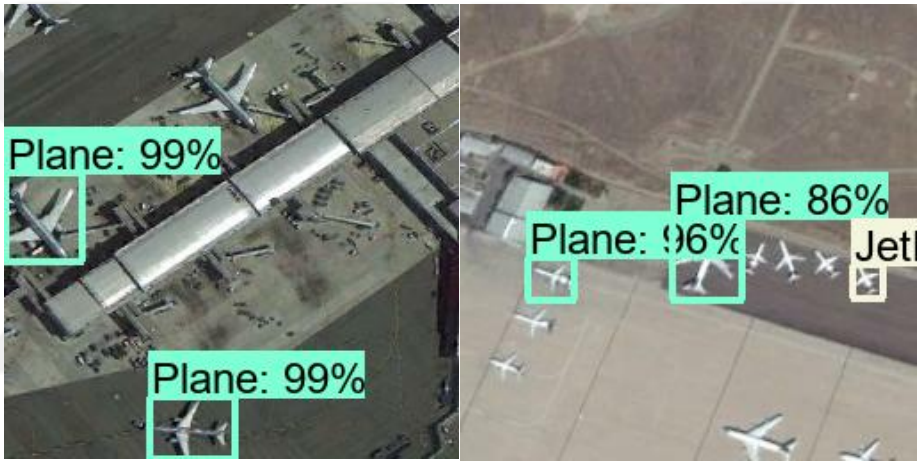

Figure A.37: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 6.

Figure A.38: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 6.

Figure A.39: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 7.

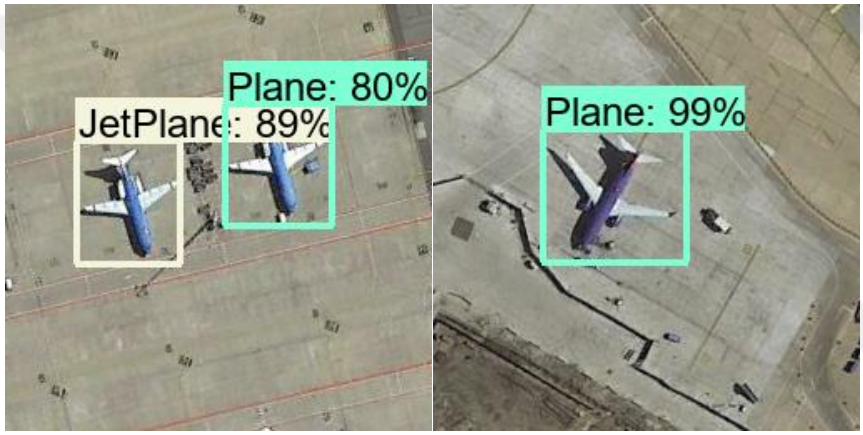

Figure A.40: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 7.

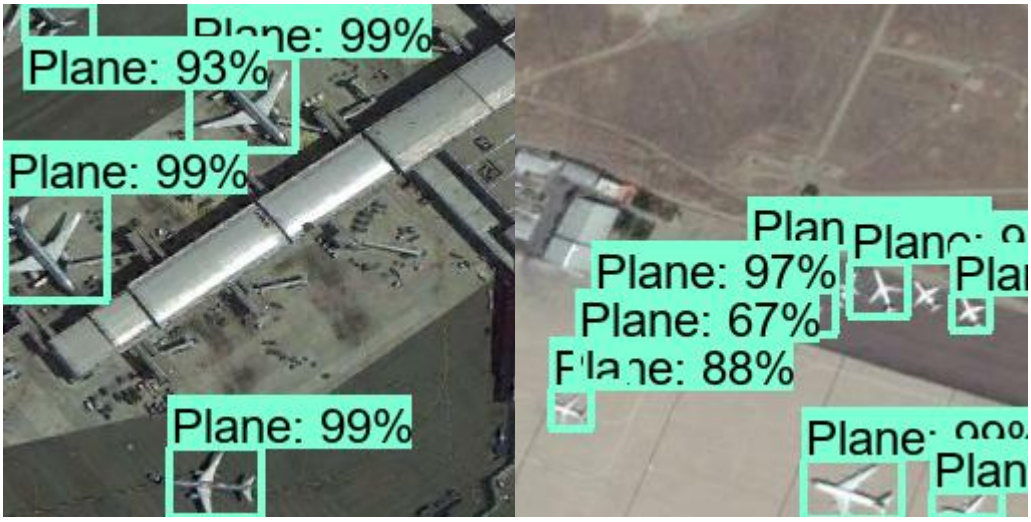

Figure A.41: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 7.

Figure A.42: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 7.

Figure A.43: Sample 1 of visualized evaluation images of test dataset of Faster_R-CNN 8.

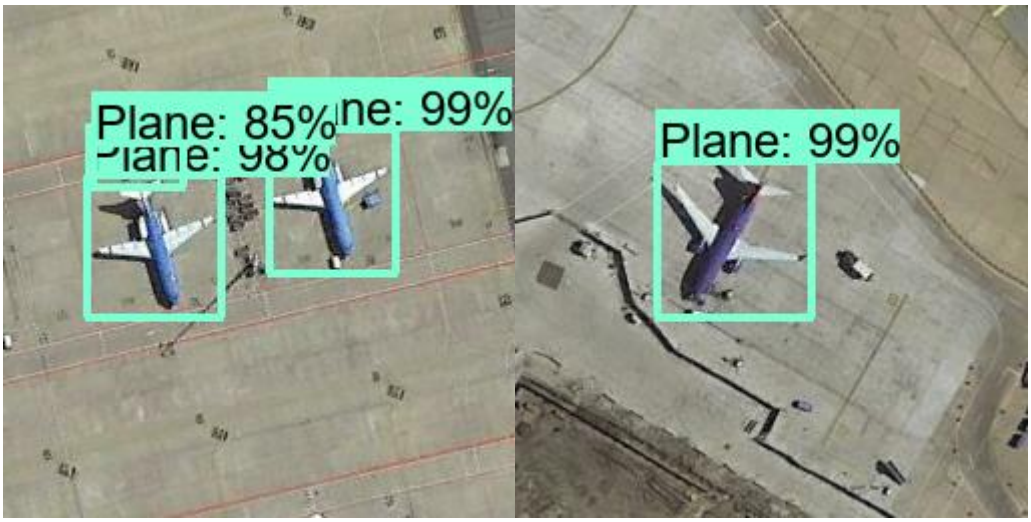

Figure A.44: Sample 2 of visualized evaluation images of test dataset of Faster_R-CNN 7.

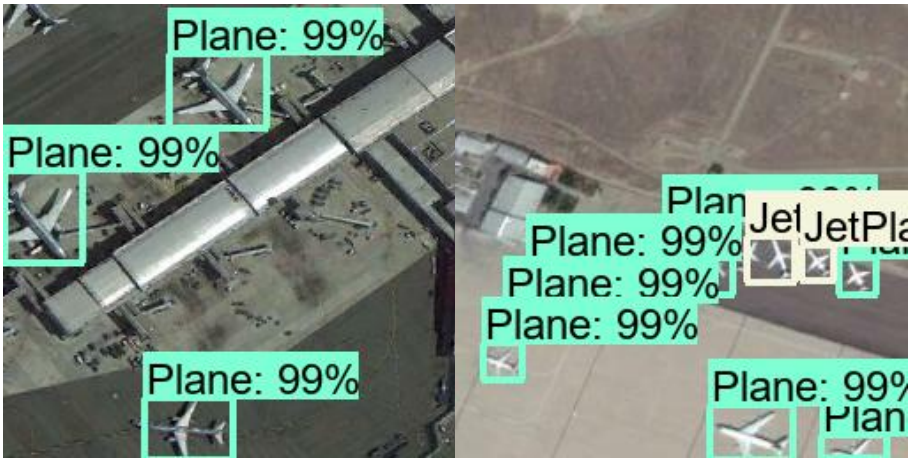

Figure A.45: Sample 3 of visualized evaluation images of test dataset of Faster_R-CNN 7.

Figure A.46: Sample 4 of visualized evaluation images of test dataset of Faster_R-CNN 7.

Figure A.47: Sample 1 of visualized evaluation images of test dataset of SSD 1.

Figure A.48: Sample 2 of visualized evaluation images of test dataset of SSD 1.

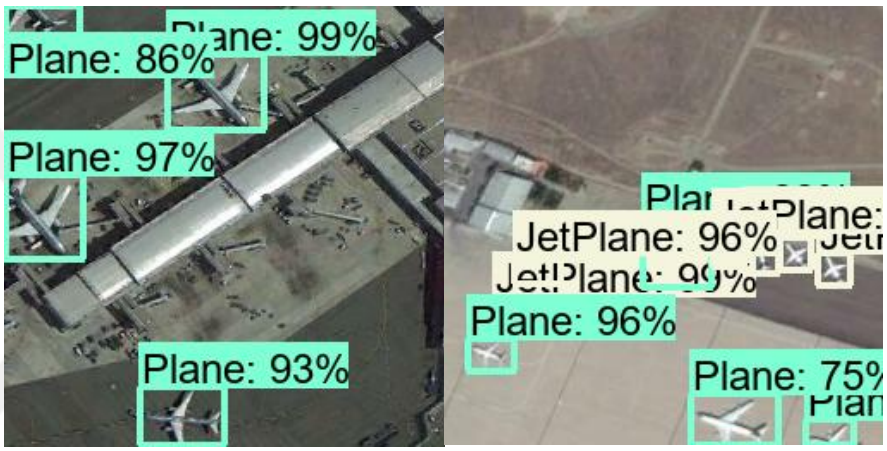

Figure A.49: Sample 3 of visualized evaluation images of test dataset of SSD 1.

Figure A.50: Sample 4 of visualized evaluation images of test dataset of SSD 1.

Figure A.51: Sample 1 of visualized evaluation images of test dataset of SSD 2.

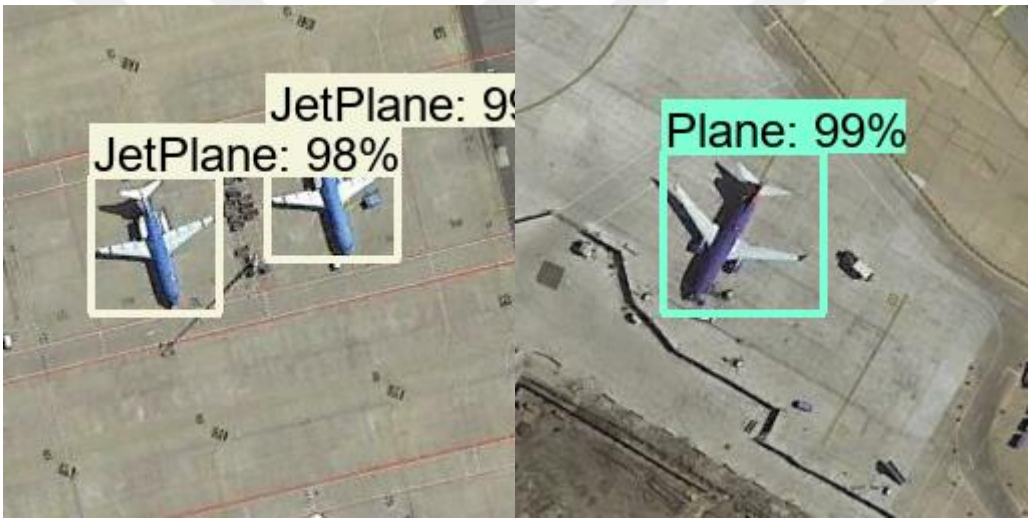

Figure A.52: Sample 2 of visualized evaluation images of test dataset of SSD 2.

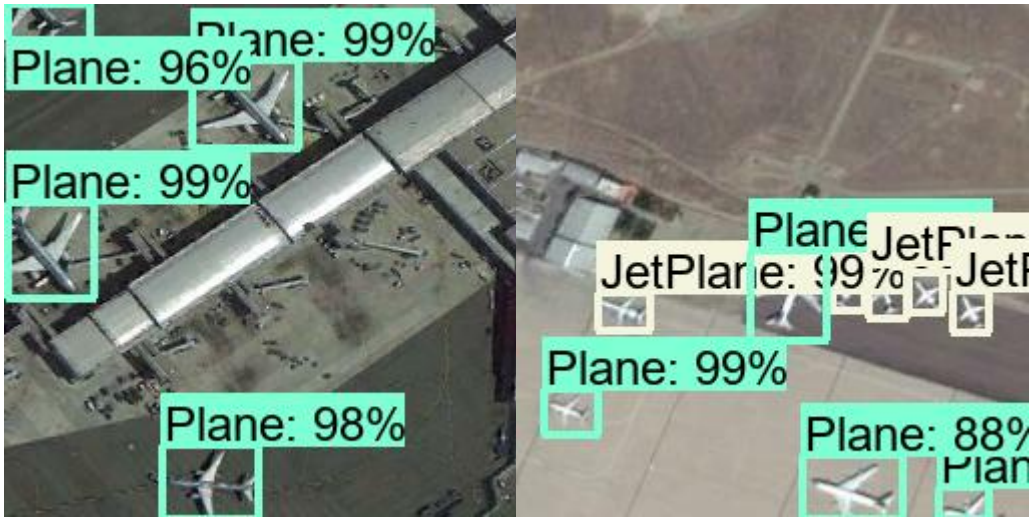

Figure A.53: Sample 3 of visualized evaluation images of test dataset of SSD 2.

Figure A.54: Sample 4 of visualized evaluation images of test dataset of SSD 2.

Figure A.55: Sample 1 of visualized evaluation images of test dataset of SSD 3.

Figure A.56: Sample 2 of visualized evaluation images of test dataset of SSD 3.

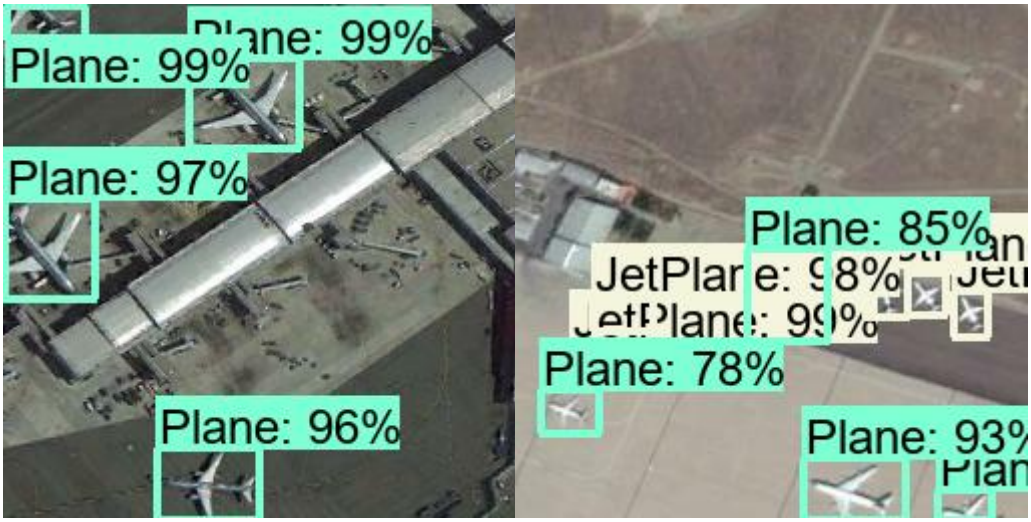

Figure A.57: Sample 3 of visualized evaluation images of test dataset of SSD 3.

Figure A.58: Sample 4 of visualized evaluation images of test dataset of SSD 3.

Figure A.59: Sample 1 of visualized evaluation images of test dataset of SSD 4.

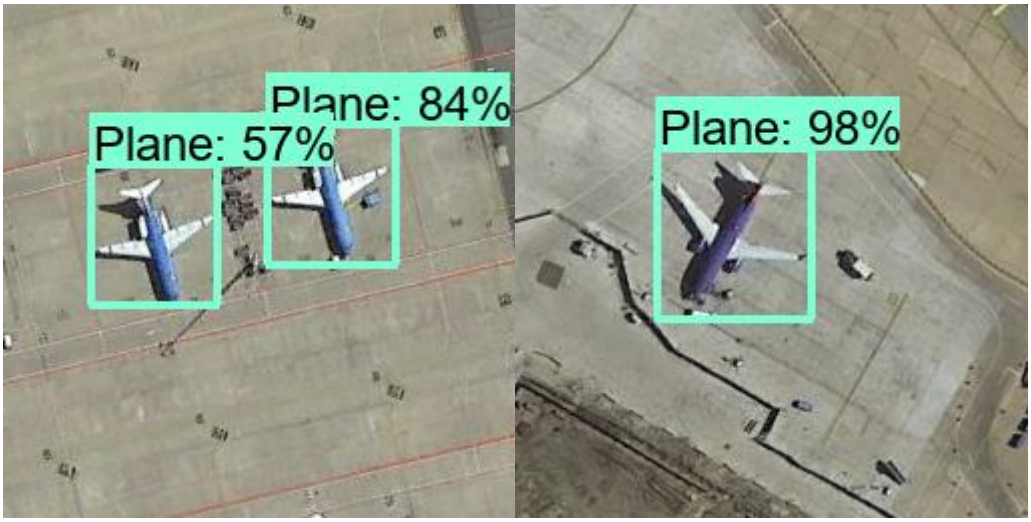

Figure A.60: Sample 2 of visualized evaluation images of test dataset of SSD 4.

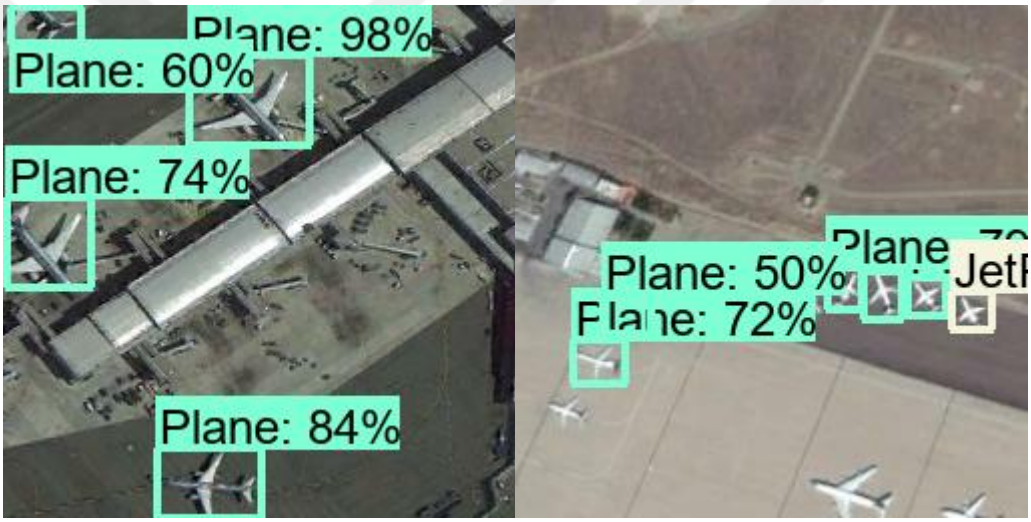

Figure A.61: Sample 3 of visualized evaluation images of test dataset of SSD 4.

Figure A.62: Sample 4 of visualized evaluation images of test dataset of SSD 4.

Figure A.63: Sample 1 of visualized evaluation images of test dataset of SSD 5.

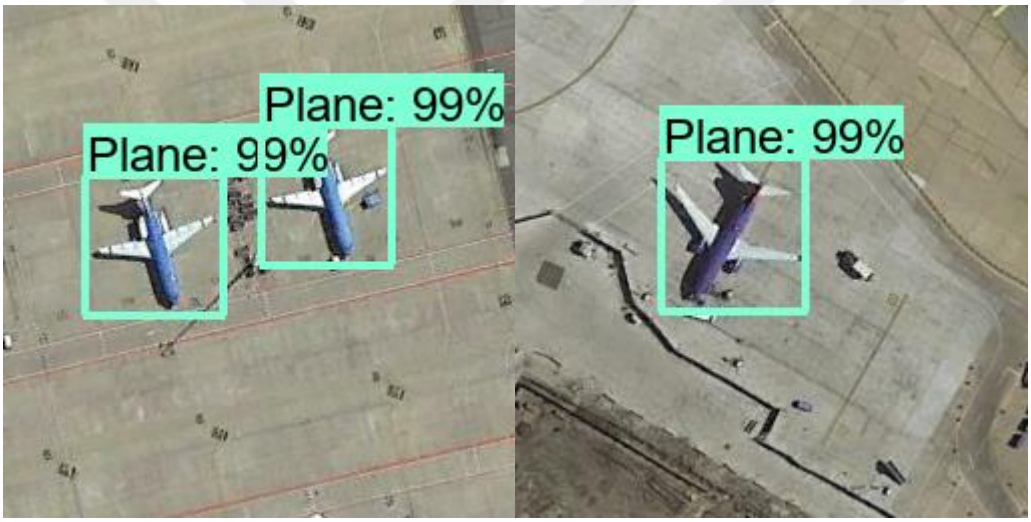

Figure A.64: Sample 2 of visualized evaluation images of test dataset of SSD 5.

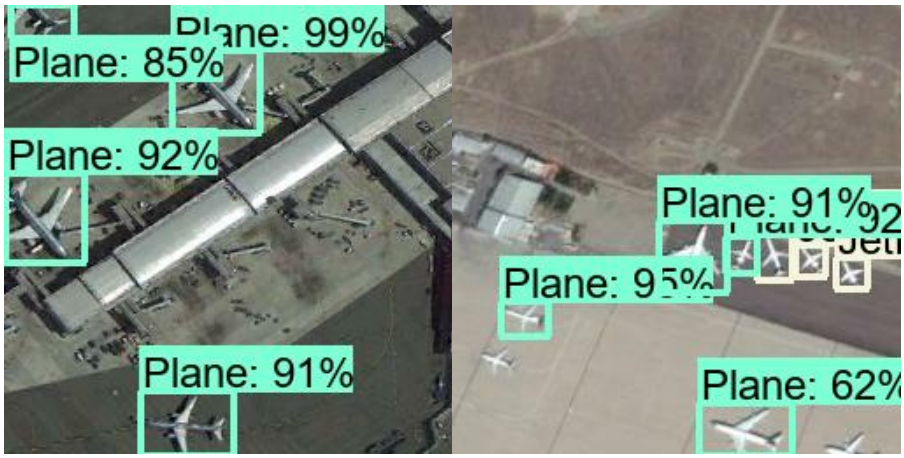

Figure A.65: Sample 3 of visualized evaluation images of test dataset of SSD 5.

Figure A.66: Sample 4 of visualized evaluation images of test dataset of SSD 5.

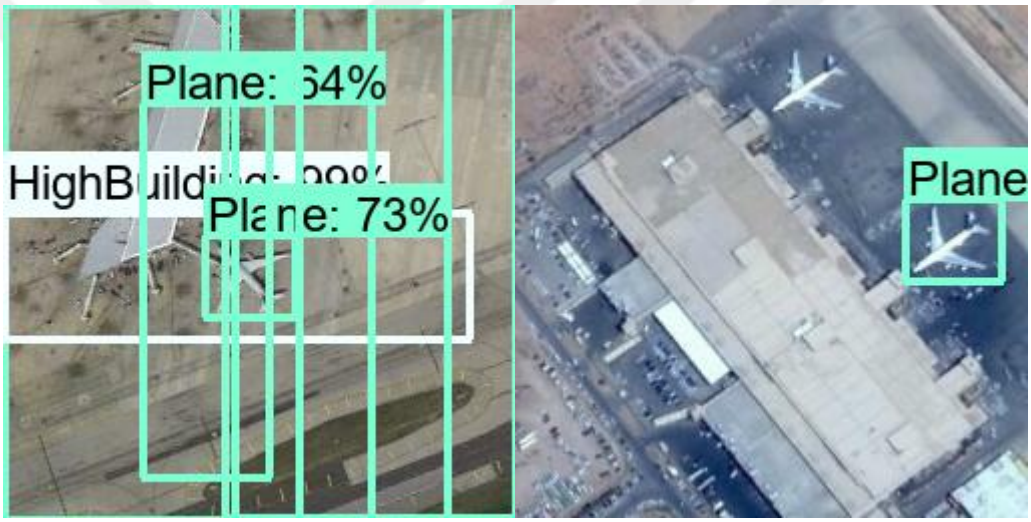

Figure A.67: Sample 1 of visualized evaluation images of test dataset of SSD 6.

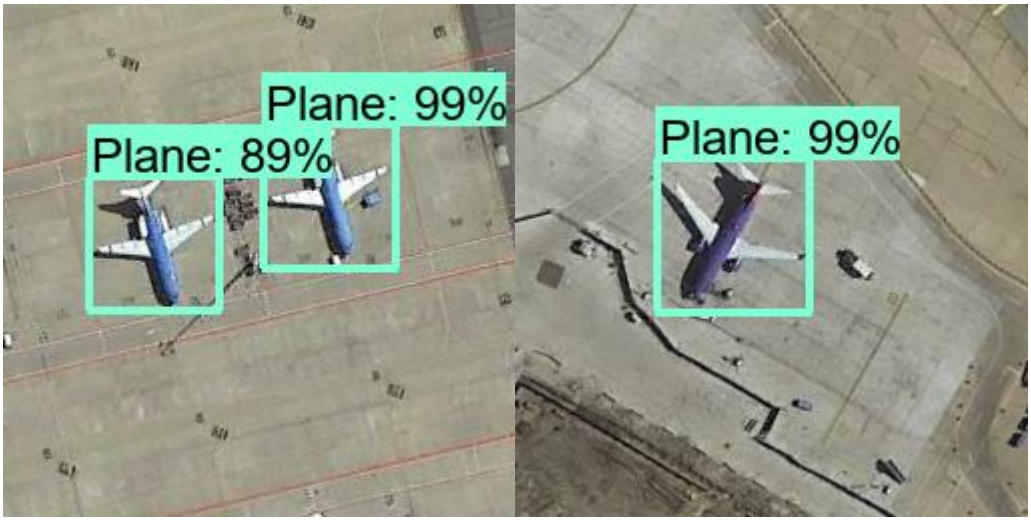

Figure A.68: Sample 2 of visualized evaluation images of test dataset of SSD 6.

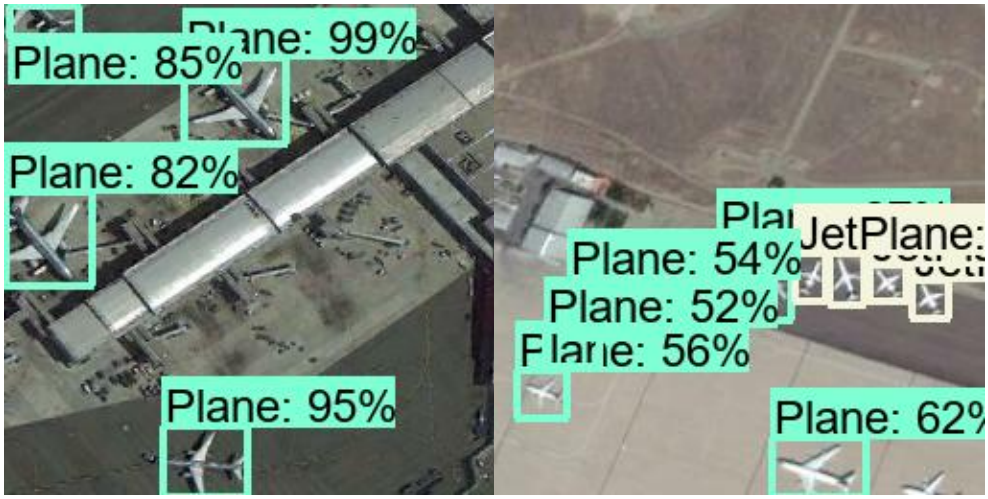

Figure A.69: Sample 3 of visualized evaluation images of test dataset of SSD 6.

Figure A.70: Sample 4 of visualized evaluation images of test dataset of SSD 6.

Figure A.71: Sample 1 of visualized evaluation images of test dataset of SSD 7.

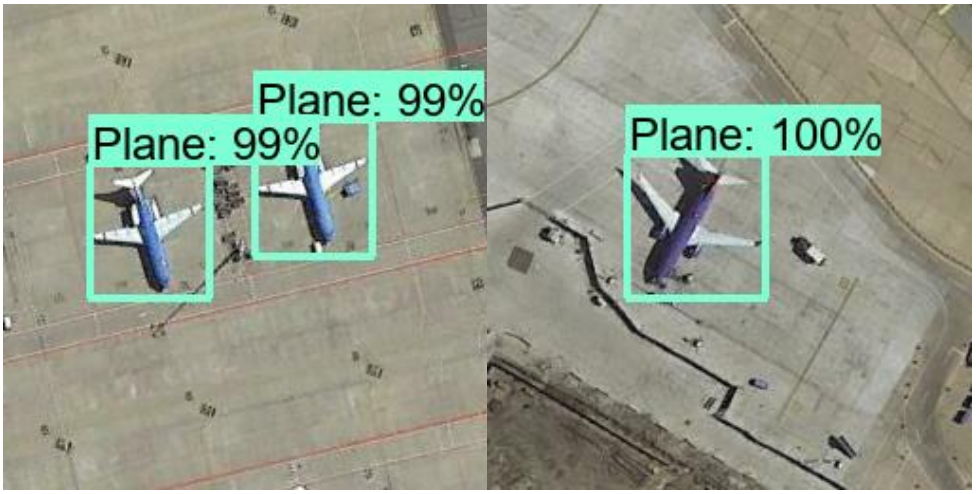

Figure A.72: Sample 2 of visualized evaluation images of test dataset of SSD 7.

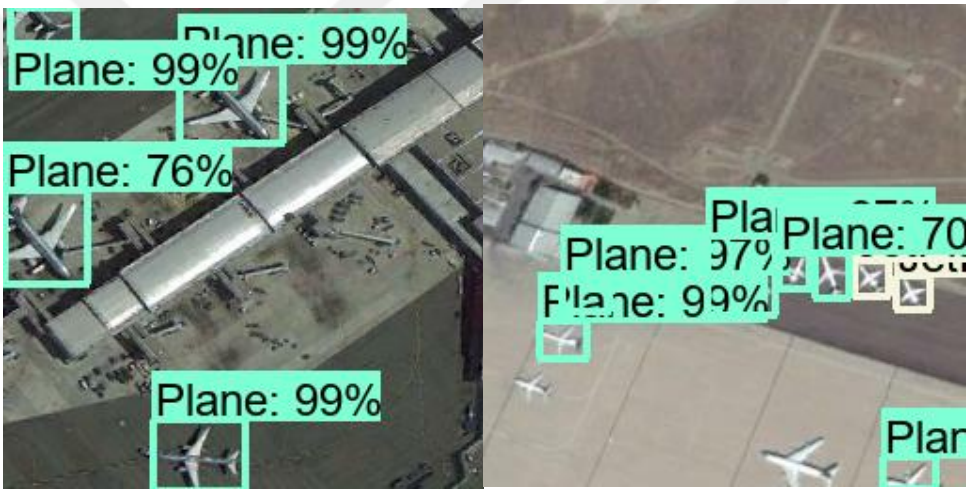

Figure A.73: Sample 3 of visualized evaluation images of test dataset of SSD 7.

Figure A.74: Sample 4 of visualized evaluation images of test dataset of SSD 7.

CURRICULUM VITAE

Name Surname: Bakary TRAORE

Place and Date of Birth: Zangue-Oume (Ivory Coast)/ 20-03-1991

Address: Celiktepe Mh. Mahmut Sevket Pasa Cad. Dogru Sk. ½ Kagithane/Istanbul 34413, Turkey

E-Mail: traoreb956@gmail.com, traore18@itu.edu.tr

EDUCATION:

B.Sc.: Mechatronics Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

Position	Company	Location	Period
Satellite Communication and Remote Sensing	ITU-CSCRS	Istanbul/Turkey	01/2019-**Ongoing**
Mechatronics Engineer (Training)	MIMAR SC	Sanok/Poland	07/2019-08/2019
Mechatronics Engineer (Internship)	Kenar Muhendislik Ltd	Ankara/Turkey	06/2017-07/2017
Mechanical Engineer (Internship)	Schaeffler Group Romania	Brasov/Romania	07/2016-08/2016
English And French Teacher	Baskent Yildirim Koleji	Ankara/Turkey	03/2017-05/2017
English Teacher	Ozuman Akademi	Ankara/Turkey	10/2016-01/2017
Import-Export	Meka	Ankara/Turkey	04/2015-09/2015
Import-Export	Herkul Pharma	Ankara/Turkey	04/2014-06/2014
Farmer	Cocoa&Coffee Farm	Oume/Ivory Coast	2009-2013