ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AIRCRAFT DETECTION FROM LARGE SCALE REMOTE SENSING IMAGES WITH DEEP LEARNING TECHNIQUES

M.Sc. THESIS

Mehmet SOYDAŞ

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

AIRCRAFT DETECTION FROM LARGE SCALE REMOTE SENSING IMAGES WITH DEEP LEARNING TECHNIQUES

M.Sc. THESIS

Mehmet SOYDAŞ
(705141023)

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Prof. Dr. Elif SERTEL

JUNE 2019

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

BÜYÜK ÖLÇEKLİ UZAKTAN ALGILAMA GÖRÜNTÜLERİNDEN DERİN ÖĞRENME TEKNİKLERİYLE UÇAK TESPİTİ

YÜKSEK LİSANS TEZİ

Mehmet SOYDAŞ
(705141023)

İletişim Sistemleri Anabilim Dalı

Uydu Haberleşmesi ve Uzaktan Algılama Programı

Tez Danışmanı: Prof. Dr. Elif SERTEL

HAZİRAN 2019

Mehmet Soydaş, a **M.Sc.** student of ITU Informatics Institute student ID **705141023**, successfully defended the **thesis** entitled "**Aircraft Detection from Large Scale Remote Sensing Images with Deep Learning Techniques**", 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** İstanbul Technical University	....
Jury Members :	**Prof. Dr. Şinasi KAYA** İstanbul Technical University	....
	Prof. Dr. Bülent BAYRAM Yıldız Technical University	....

Date of Submission : **3 May 2019**
Date of Defense : **11 June 2019**

To my precious family and close friends,

FOREWORD

Master education is a challenging process that requires very serious labor. At the time, while we are trying to regulate our lives after bachelor education, entering such a process additionally could be abrasive. Therefore, the contribution of your advisor, supports and trust of your relatives are becoming priceless values that help you to get up and go forward when your motivation falls. For these reasons, my relatives and advisors deserve to receive thanks.

First of all, I would like to thank my advisor Prof. Dr. Elif SERTEL who guided and supported me with sharing her valuable knowledge, experience and also allowing me to use resources of the ITU CSCRS. I would like to thank all of my friends, especially to Berk Güney and Okan Ulusoy who support me and make me forget the stressful conditions with their friendships while doing the thesis works.

Last, but most intense thanks go to my family; to my mother who is the biggest shareholder and motivation source of my life, to my father who always felt his presence and supports me with all my decisions. I would also thank my sisters separately for they have been enduring me.

Thanks, everyone who inspiring me and contributed my education life.

May 2019 Mehmet SOYDAŞ

TABLE OF CONTENTS

Page

FOREWORD .... **ix**
TABLE OF CONTENTS .... **xi**
ABBREVIATIONS .... **xiii**
LIST OF TABLES .... **xv**
LIST OF FIGURES .... **xvii**
SUMMARY .... **xxi**
ÖZET .... **xxiii**
1. INTRODUCTION **1**
  1.1 Purpose of Thesis 4
2. LITERATURE OVERVIEW ON GEOSPATIAL OBJECT ANALYSİS **5**
  2.1 Handcrafted Features 6
    2.1.1 Texture features 6
    2.1.2 Scale invariant transform features (SIFT) 6
    2.1.3 Histogram of oriented gradient features (HOG) 7
    2.1.4 Bag of words features (BoW) 8
  2.2 Unsupervised Learning Techniques 8
  2.3 Supervised Learning Techniques 9
3. FUNDAMENTALS OF MACHINE LEARNING **11**
  3.1 Unsupervised Learning Methods 11
    3.1.1 K-means clustering 11
    3.1.2 Hierarchical clustering 12
    3.1.3 Autoencoders 13
  3.2 Supervised Learning Methods 15
    3.2.1 Support vector machine (SVM) 15
    3.2.2 Adaboost algorithm 16
    3.2.3 K-nearest neighbor (k-NN) 17
    3.2.4 Artificial neural networks (ANN) 18
      3.2.4.1 Convolutional neural network (CNN) 19
        Convolutional layers 19
        Pooling filters 20
        Fully connected layers 21
4. METHODOLOGY AND ARCHITECTURE REVIEW **23**
  4.1 Region Proposal Network : Faster R-CNN 23
    4.1.1 Loss function 26
    4.1.2 Residual blocks 26
  4.2 Single Shot Multibox Detector (SSD) 27
    4.2.1 Default bounding boxes and negative sample generation 29
    4.2.2 Loss function 30
  4.3 You Look Only Once v3 (YOLO) 31
  4.4 Detection Flow 33
5. EXPERIMENTS AND RESULTS **35**

5.1 Dataset 35
5.2 Training of the Object Detection Networks 36
5.3 Evaluation Metrics 39
    5.3.1 Intersection over union (IOU) 39
    5.3.2 Performance metrics 40
5.4 Results 40
    5.4.1 SSD evaluation results 43
    5.4.2 Faster R-CNN evaluation results 47
    5.4.3 Yolo-v3 evaluation results 51
6. CONCLUSIONS 75
REFERENCES 77
CURRICULUM VITAE 81

ABBREVIATIONS

1D : 1 Dimensional
2D : 2 Dimensional
ANN : Artificial Neural Network
BOW : Bag-Of-Words
CNN : Convolutional Neural Network
COCO : Common Objects in Context
DOG : Difference of Gaussian
DOTA : A Large-scale Dataset for Object Detection in Aerial Images
DR : Detection Rate
ETM : Enhanced Thematic Mapper
FAR : False Alarm Rate
FC : Fully Connected
GEOBIA : Geographic Object Based Image Analysis
HOG : Histogram of Oriented Gradient
HSV : Hue Saturation Value
ILSVRC : ImageNet Large Scale Visual Recognition Challenge
K-NN : K-Nearest Neighbor
LBP : Local Binary Pattern
LSD : Line Segment Detector
LULC : Land Use Land Cover
NLP : Natural Language Processing
NMS : Non Maximum Suppression
OBIA : Object Based Image Analysis
PASCAL VOC : Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes
PCA : Principal Component Analysis
R-CNN : Regions with Convolutional Neural Networks
RELU : Rectified Linear Unit
RESNET : Residual Network
RMS : Root Mean Square
ROI : Region of Interest
RPN : Region Proposal Network
RSI : Remote Sensing Image
SAR : Synthetic Aperture Radar
SIFT : Scale-Invariant Feature Transform
SPP-NET : Spatial Pyramid Pooling Network
SSCBoW : Spatial Sparse Coding Bag of Words
SSD : Single Shot Multibox Detector
SVM : Support Vector Machine
UAV : Unmanned Aerial Vehicle
VGG : Visual Geometry Group
YOLO : You Look Only Once

API **:** Application Programming Interface
AP **:** Average Precision
AR **:** Average Recall
PR **:** Precision Recall

LIST OF TABLES

Page

Table 5.1 : Performance metrics with calculation rules. **41**
Table 5.2 : The ratios(%) of data sets according to object scale. **42**
Table 5.3 : Performance of DOTA training, validation and large scale image set according to COCO metrics.. **43**
Table 5.4 : mAP metrics of all test sets for all networks according to COCO Metric API.. **56**
Table 5.5 : Precision, Recall and F1 score of all datasets. **57**

LIST OF FIGURES

Page

Figure 3.1 : K-means clustering algorithm. Data samples are represented as dots and cluster centroids as shown as crosses; (a) Original dataset, (b) Random initialization of cluster centroids, (c-f) Iterations to find clusters. **12**
Figure 3.2 : Hierarchical agglomerative clustering dendrogram for the animal recognition. Axis y is the similarity metric. **13**
Figure 3.3 : An example of Autoencoder. **14**
Figure 3.4 : An example of Linear SVM classifier for two classes in 2D feature space. **15**
Figure 3.5 : Illustration of obtaining strong classifier with Adaboost algorithm. **17**
Figure 3.6 : Illustration of (a) biological and artificial neuron (b) artificial neural networks. **18**
Figure 3.7 : 2D convolution opertion with zero padding, one stride and 3x3 filter. **20**
Figure 3.8 : Rectified linear unit function. **20**
Figure 3.9 : 2x2x1 average and maximum pooling filters with 2 stride number. **21**
Figure 3.10 : Illustration of fully connected layers, last layer as the classifier layer. **21**
Figure 4.1 : Selective search algorithm, bottom-up segmentation, merging regions at multiple scales. **24**
Figure 4.2 : Processes of the Faster R-CNN. **24**
Figure 4.3 : Faster R-CNN detection network. **25**
Figure 4.4 : Smooth L1 loss curve. **26**
Figure 4.5 : Residual blocks. **27**
Figure 4.6 : InceptionV2 architecture. **28**
Figure 4.7 : Illustration of the inception modules. **28**
Figure 4.8 : SSD architecture uses Inception V2 as a base network with 16 of batch size for training. **29**
Figure 4.9 : Illustration of 5x5 feature map and generated default bounding box with 6 aspect ratio. **30**
Figure 4.10 : Yolo-v1, architecture has 24 convolutional layers followed by 2 fully connected layers. **31**
Figure 4.11 : Yolo v3 architecture. **32**
Figure 4.12 : Illustration of sliding detection flow. **33**
Figure 4.13 : Non maximum suppression algorithm (NMS). **34**
Figure 4.14 : Detection results of occluded objects (a) without NMS algorithm, (b) with NMS algorithm. **34**

Figure 5.1 : Patches from DOTA test set; (a) cropped, (b) very big, (c) very small, (d) complex background, (e) illuminance effect, (f) panchromatic samples. **36**
Figure 5.2 : Number of the pixel size of airplane bounding boxes in the dataset.................... **36**
Figure 5.3 : Yolo v3 training and validation loss graphic.................... **38**
Figure 5.4 : Illustration of data augmentations; (a) Original image patch, (b) Rotated and scaled in HSV, (c) Rotated, cropped and scaled in HSV.................... **39**
Figure 5.5 : Intersection over Union. **39**
Figure 5.6 : 12 performance metrics of COCO challenge.................... **41**
Figure 5.7 : SSD precision-recall curve of DOTA test set for all objects size...... **43**
Figure 5.8 : SSD precision-recall curve of DOTA test set for large objects size.................... **44**
Figure 5.9 : SSD precision-recall curve of DOTA test set for medium objects size. **44**
Figure 5.10 : SSD precision-recall curve of DOTA test set for small objects size.................... **45**
Figure 5.11 : SSD precision-recall curve of large scale test set for all objects size.................... **45**
Figure 5.12 : SSD precision-recall curve of large scale test set for large objects size. **46**
Figure 5.13 : SSD precision-recall curve of large scale test set for medium objects size. **46**
Figure 5.14 : SSD precision-recall curve of large scale test set for small objects size. **47**
Figure 5.15 : Faster R-CNN precision-recall curve of DOTA test set for all objects size. **47**
Figure 5.16 : Faster R-CNN precision-recall curve of DOTA test set for large objects size. **48**
Figure 5.17 : Faster R-CNN precision-recall curve of DOTA test set for medium objects size. **48**
Figure 5.18 : Faster R-CNN precision-recall curve of DOTA test set for small objects size.................... **49**
Figure 5.19 : Faster R-CNN precision-recall curve of large scale test set for all objects size. **49**
Figure 5.20 : Faster R-CNN precision-recall curve of large scale test set for large objects size. **50**
Figure 5.21 : Faster R-CNN precision-recall curve of large scale test set for medium objects size. **50**
Figure 5.22 : Faster R-CNN precision-recall curve of large scale test set for small objects size.................... **51**
Figure 5.23 : Yolo-v3 precision-recall curve of DOTA test set for all objects size.................... **51**
Figure 5.24 : Yolo-v3 precision-recall curve of DOTA test set for large objects size. **52**
Figure 5.25 : Yolo-v3 precision-recall curve of DOTA test set for medium objects size. **52**
Figure 5.26 : Yolo-v3 precision-recall curve of DOTA test set for small objects size. **53**

Figure 5.27 : Yolo-v3 precision-recall curve of large scale test set for all objects size. **53**
Figure 5.28 : Yolo-v3 precision-recall curve of large scale test set for large objects size. **54**
Figure 5.29 : Yolo-v3 precision-recall curve of large scale test set for medium objects size. **54**
Figure 5.30 : Yolo-v3 precision-recall curve of large scale test set for small objects size. **55**
Figure 5.31 : Detection results of some of the DOTA test set patches (a) Yolo-v3, (b) SSD, (c) Faster R-CNN. **58**
Figure 5.32 : Yolo-v3 evaluation for the Antalya Airport. **59**
Figure 5.33 : Yolo-v3 evaluation for the Istanbul Ataturk Airport. **60**
Figure 5.34 : Yolo-v3 evaluation for the Istanbul Sabiha Gökçen Airport. **61**
Figure 5.35 : Yolo-v3 evaluation for the Esenboga Airport. **62**
Figure 5.36 : Yolo-v3 evaluation for the Izmir Adnan Menderes Airport. **63**
Figure 5.37 : SSD evaluation for the Antalya Airport. **64**
Figure 5.38 : SSD evaluation for the Istanbul Ataturk Airport. **65**
Figure 5.39 : SSD evaluation for the Istanbul Sabiha Gökçen Airport. **66**
Figure 5.40 : SSD evaluation for the Esenboga Airport. **67**
Figure 5.41 : SSD evaluation for the Izmir Adnan Menderes Airport. **68**
Figure 5.42 : Faster R-CNN evaluation for the Antalya Airport. **69**
Figure 5.43 : SSD evaluation for the Istanbul Ataturk Airport. **70**
Figure 5.44 : SSD evaluation for the Istanbul Sabiha Gökçen Airport. **71**
Figure 5.45 : SSD evaluation for the Esenboga Airport. **72**
Figure 5.46 : SSD evaluation for the Izmir Adnan Menderes Airport. **73**

AIRCRAFT DETECTION FROM LARGE SCALE REMOTE SENSING IMAGES WITH DEEP LEARNING TECHNIQUES

SUMMARY

Computer vision and artificial intelligence are not new fields in people's lives. In order to automate the problems in our lives in a way that does not require human resources, problem-specific morphological methods were investigated and tried over the years. Recently, these morphological approaches have been replacing by deep learning methods in many fields thanks to the hardware which came up with high computational power, a vast amount of data in the digital world and rapid development of machine learning and deep learning algorithms.

As in every field, these methods are also used in the analysis of remotely sensed images and their usage is becoming more widespread. The analysis of satellite images plays a very important role in many areas such as defining forest areas and fires, monitoring of cultivated areas in agricultural lands, city and road planning, security and military surveillance, disaster and crisis management. Considering the satellite images, which can cover many square kilometers of areas, it is very costly and time-consuming to perform these analyzes by people. In addition, in order to obtain accurate results, it is also necessary, that the people to be used for these tasks, must be experts in their field. Regard all, it is expected that the obtained computer vision system should give both fast results and at least as much accurate as of the people.

In the analysis of satellite images, computer vision solutions are categorized into three main topics as classification, segmentation and object detection. Classification and segmentation are examined in the sub-topics as pixel-based classification, scene classification, semantic segmentation, and instance segmentation. In all these analyses, convolutional neural network(CNN), a deep learning architecture which utilizes the spatial and spectral correlations on the image can be used and high performances can be achieved.

In this study, aircraft detection from satellite imageries with deep architectures and traditional methods was discussed. Different object detection algorithms based on deep learning approaches were trained and tested. For the evaluation, the images containing airport areas were manually labeled. A detection flow algorithm was developed for large scale satellite images for rapid detection and high accuracy. The effects of using different architectures and the effects of training methods on the performance were investigated.

BÜYÜK ÖLÇEKLİ UZAKTAN ALGILAMA GÖRÜNTÜLERİNDEN DERİN ÖĞRENME TEKNİKLERİ İLE UÇAK TESPİTİ

ÖZET

Bilgisayarlı görü ve yapay zeka konuları insan hayatında yeni bir alan değil. Yıllardır hayatlarımızdaki problemleri insan kaynağı gerektirmeyecek şekilde otomatize edebilmek adına probleme özel morfolojik yöntemler araştırılmakta ve denenmektedir. Son zamanlarda donanımlardaki yüksek hesaplama gücü, veri miktarı ve algoritmaların hızlı gelişimiyle birlikte birçok alanda bu morfolojik yaklaşımlar yerini derin öğrenme yöntemlerine bırakmaya başladı.

Her alanda olduğu gibi uydu görüntülerinin analizlerinde de bu yöntemler ilgi görmekte ve kullanımı yaygınlaşmaktadır. Uydu görüntülerinin analizi orman alanlarının ve yangınlarının belirlenmesi, tarım arazilerindeki ekili alanların takibi, şehir ve yol planlaması, güvenlik ve askeri gözetlemeler, afet ve kriz yönetimi gibi birçok konuda çok önemli roller oynamaktadır. Kilometrelerce karelik alanları içerebilen uydu görüntüleri düşünüldüğünde, bu analizlerin insan tarafından yapılabilmesi çok maliyetli ve zaman gerektiren işlemlerdir. Ayrıca doğru sonuçları elde edebilmek için analiz görevinde kullanılacak insanların, alanında uzman kişiler olması da gerekmektedir. Tüm bunlar düşünüldüğünde probleme özgü oluşturulacak bilgisayarlı görü sisteminin hem hızlı sonuç verebilmesi, hem de en az insanlar kadar yüksek doğruluk oranında çalışması beklenmektedir.

Uydu görüntülerinin analizinde bilgisayarlı görü çözümleri sınıflandırma, bölütleme ve nesne tespiti olarak üç başlık altında toplanır. Sınıflandırma ve bölütleme ise kendi içinde pixel tabanlı sınıflandırma, alan sınıflandırması, anlamsal bölütleme ve örnek bölütleme şeklinde alt başlıklarda incelenir. Tüm bu analizlerde görüntü üzerindeki konumsal ve spektral korelasyonlardan faydalanan derin öğrenme mimarisi olan evrişimli sinir ağları(CNN) kullanılabilmekte ve yüksek başarımlar elde edilmektedir.

Bu çalışmada uydu görüntülerinden uçak tespiti konusu ele alınmış, geleneksel yöntemler ile derin öğrenme tekniğine dayalı farklı sinir ağı mimarileri eğitilmiş ve test edilmiştir. Test için havalimanı bölgelerini içeren görüntülerde elle etikeletme yapılmıştır. Büyük ölçekli görüntülerde hızlı tespit ve yüksek başarım için bir algoritma geliştirilmiş, farklı mimarilerin kullanımı ve eğitim yöntemlerinin başarıma etkileri incelenmiştir. Çalışmada öncelikle literatür taranmış ve farklı yaklaşımlar incelenmiş, daha sonra makine öğrenmesi temelleri hakkında bilgi paylaşılmıştır. Makine öğrenmesinin alt başlığı olan derin öğrenme konusuna da değinilmiştir. Çalışmanın bel kemiğini oluşturan evrişimsel sinir ağları tanıtılmıştır ve temel kavramları üzerinde durulmuştur.

Derin öğrenme teknikleriyle çalışan nesne tespit modelleri evrişimsel sinir ağlarını öznitelik çıkarıcı olarak kullanmaktadırlar. Dolayısıyla çalışmanın metodoloji

kısmında CNN ile nesne tespit mimarilerinin kesiştiği kısımlara değinilmiş, son teknoloji tespit mimarileri incelenmiştir. Büyük ölçekli uydu görüntülerinde hızlı ve yüksek başarımla tespit gerçekleştirebilmek için kayan pencere yöntemi ve azami baskılama algoritmalarından yararlanılmıştır. Veri seti olarak “A Large-scale Dataset for Object Detection in Aerial Images (DOTA)” veriseti ve ayrıca test için hazırlanan 5 büyük havalimanı görüntüsünü içeren bir veriseti kullanılmıştır. Mimarilerin eğitimleri için farklı parametreler ve optimizasyon yöntemleri denenmiş ve sonuçlar COCO Metrik API kullanılarak 12 farklı metrik için çıkarılmıştır. Buna ek olarak modellerin F1 skorları da incelenmiş çalışmanın tespit sonuçları havalimanı bölgelerini içeren büyük ölçekli uydu görüntülerinden elde edilerek paylaşılmıştır.

Tespit mimarilerinde sınıflandırma işlemine ek olarak konumlandırma problemine de çözüm aranır. Sınıflandırma problemlerinde derin öğrenme mimarilerinin başarılarının artmasıyla birlikte nesne tespiti için de "Single Shot Multibox Detector (SSD), Faster Region-based Convolotional Neural Network (Faster R-CNN), Yolo Look Only Once (YOLO-v3)" gibi farklı mimariler ortaya çıkmıştır. Bu mimariler, nesne tespiti yapılabilmesi için gerekli olan sınıflandırma ve konumlandırma problemlerini tek bir sinir ağı ve yüksek başarımlar ile çözebilmektedirler. Bu son teknoloji mimariler günlük hayattaki nesnelerin video görüntüleri üzerinden tespitinin yapıldığı "Common Objects in Context (COCO) ve Pattern Analysis, Statistical Modeling and Computational Learning (Pascal VOC)" gibi yarışmalarda yüksek başarımlar elde ettiler ve hızlı sonuç sağlayabildikleri için de çokca kullanılmaktadırlar. Aynı şekilde son yıllarda uydu görüntülerinden nesne tespiti için de kullanılmaya başlanmış ve tatmin edici sonuçlar elde edilmiştir.

Derin öğrenme algoritmalarının eğitiminde mimarilerin yapısının yanında, uygun veriseti hazırlanması, parametre seçimi, optimizasyon yöntemleri ve eğitim sonuçlarını anlamlandırabilmek çok önemlidir. Bu amaçla uçak tespitini gerçekleştirebilmek için gayet kapsamlı ve çeşitliliği bol olan DOTA verisetinde bulunan uçak örnekleri kullanılmıştır. Veri sayısının fazla olması eğitilen modellerin her koşula uygun ve daha başarılı olmalarını sağlamaktadır. Dolayısıyla eğitimlerin her adımında tüm örneklere rastgele olacak şekilde kesme, döndürme uygulanıp, renk ve doygunluk değerleri değiştirilerek, veri çoklama işlemi uygulanmıştır. Parametreler eğitim aşamasında modellerin kayıp değerleri incelenerek öğrenme eğilimlerine göre belirlenmiştir. Yolo-v3 modelinin eğitiminde kullanılmak üzere bazı parametrelerin belirlenmesinde gözetimsiz bölütleme algoritması olan K-means algoritmasından yararlanılmıştır. Üç farklı nesne tespit mimarisi için de “Stochastic Gradient Descent (SGD), Root Mean Square Propagation (Rms-prop) ve Adaptive Moment Optimization (Adam)” optimizasyon yöntemlerinden yararlanılmıştır.

Eğitilmiş modellerle büyük ölçekli uzaktan algılama görüntülerinde uçak tespiti yapabilmek için kayan pencere yöntemi ile büyük görüntüler taranmaktadır. Derin öğrenme algoritmaları maliyetli çözümler oldukları için olabildiğince hızlı olabilmek adına ve tespit edilemeyen nesne kalmaması için pencere sayısı optimum olacak şekilde ve pencerelerin kesişim bölgelerinin alanı verisetlerinde bulunan ortalama uçak boyutlarında tutulmuştur. Tespit işlemi bu şekilde gerçekleştirildikten sonra kesişim olan bölgelerde aynı nesne için oluşacak birden fazla tespiti eleyebilmek adına azami baskılama algoritması uygulanmıştır.

Çalışmanın sonunda eğitilen modellerin ayrı ayrı hem DOTA verisetinden ayrılan test örnekleri, hem de bu tez çalışması için hazırlanmış 5 adet büyük ölçekli uydu görüntüsü üzerinde değerlendirilmesi yapılmıştır. Performans ölçümü için COCO değerlendirme formatı esas alınarak nesne boyutlarına ve görüntü başına yapılan tespit miktarına göre ortalama hassasiyet (AP) ve ortalama duyarlılık (AR) metrikleri hesaplanmıştır. Ayrıca yine nesne boyutlarına göre hassasiyet ve duyarlılık eğrileri çizdirilerek grafikler üzerinden konumlandırma hatası, arka plan karışıklığı, kaçan tespit oranı, farklı iou (intersection over union) değerleri için başarımları yorumlanmıştır. Ayrıca DOTA verisetinin eğitim ve test kısmı ile yine büyük ölçekli uydu görüntüleri için toplam hassasiyet, duyarlılık ve ikisinin harmonik ortalaması olan F1 metriği hesaplanarak modellerin eğitim verisetine ne kadar yakınsadığı ve öğrenme işleminin başarısı gözlenmiştir.

1. INTRODUCTION

The computer vision has been a subject that has been focused by researchers for years and they tried problem-specific morphological methods to overcome the issues in this field. As in other kinds of data, the rapid increase in visual data and the need for processing and getting information from them, exponentially increase the allocation of the resources for these studies day by day. The process of extracting information from visual data requires a large amount of manpower depending on the size of the data and that brings very high financial burdens with it. Therefore, the algorithms that are faster than human and at least as accurate as they are being studied. In addition, the desire of getting done of the daily works or heavy and dangerous works by autonomous systems is another factor that accelerates computer vision researches.

The computer vision is mainly divided into three main topics as segmentation, classification and object detection. Segmentation and classification are divided into sub-topics such as for instance segmentation, semantic segmentation, pixel-based classification, and scene classification. The semantic segmentation is usually a representation of the classes with shapeless boundaries according to the correlation of the neighbor pixels. The instance segmentation method is also able to separate objects boundaries belonging to the same categories which intersect between each other at an area. The scene classification technique simply treats the whole image patch and predicts which category it belongs to. The pixel-based classification progresses by classifying all pixels in the image one by one. It is usually effective to use this method in data such as hyperspectral remote sensing images which have much more band information, but it requires much processing power and long processing time. The problem of object detection is the process of finding the individual structures in the image separately and usually showing them with the bounding boxes.

Detecting the objects from satellite images has great importance in military applications, urbanization, agriculture, natural disaster, and crisis management. However, in comparison to natural images, this process needs much more expertise for satellite imageries with very low spatial resolution and also which contain very large areas. In addition, the results will depend entirely on the decisions of the people working on these tasks, it is possible to make the wrong conclusions. Generally, the objects that are tried to be detected from satellite images are man-made structures like water tanks, buildings, bridges, aircrafts, ships, vehicles and natural objects like islands, lakes et cetera. However, the complexity of the background and the variety of objects make this task quite challenging. We can think of object detection as a combination of two fundamental tasks, namely the classification of the objects and defining their location on the image. The studies to date, have focused on the improvement of these two tasks separately or together.

In the early studies, a large part of the target detection studies has been tried to be performed by unsupervised methods using different feature extraction methods. For example, synthetic aperture radar (SAR) images used the wavelet transform in ship detection [1]. Scale-invariant feature transform (SIFT) key points and graph theorem were used in the detection of buildings from panchromatic images [2]. However, such unsupervised methods often yield successful results for objects with simple structure types and very few variations.

Subsequently, they focused on supervised methods to detect objects with different structures from more complex scenes to achieve higher performance. The main reason for achieving more successful results in supervised methods is that the learning process is carried out by obtaining information from the samples that were previously labeled manually for the training phase. Before the use of convolutional neural network (CNN) [3] structures became widespread, different handcrafted features such as SIFT [4], the histogram of oriented gradients (HOG) [5], Gabor [6], etc. were used for classification step of the object detection task by methods such as support vector machine (SVM) [7], k-nearest neighbor (k-NN) [8]. The location of the objects on the image was determined by scanning the whole image by the trained classifier, usually with the method which is called sliding window. Because of the small number of parameters in the methods trained with these features, scanning the whole image by sliding window allows making detection with acceptable speeds.

In 2012, following the remarkable success of AlexNet's [9] at ImageNet Large Scale Visual Recognition Challenge [10], CNN architectures which are also known as deep learning methods, began to attract much interest. In the years following this improvement, which we can adopt it as a milestone for deep learning, visual geometry group networks(VGG) [11] with deeper architectures, GoogleNet [12] which consisting of inception modules and residual networks (ResNet) [13] have appeared and the error rate in the competition has decreased every year. Moreover, deep architectures have exceeded human performance. With these developments, CNN structures were used in the classification step of object detection tasks. Although the success of the CNN for the classification stage of object detection is promising, the sliding window method, which has a high cost of calculation due to the fact that having a large number of parameters, started to be abandoned. Instead, these architectures were used as the base network for feature extraction utilized at the classification process and the problem of localization was solved by producing object proposals on the image, resulting in deep architectures such as Region-based Convolutional Neural Network (R-CNN) [14], Spatial Pyramid Pooling Network (SPP-NET) [15], Fast R-CNN [16] and Faster R-CNN [17]. With these structures, accurate results and high speed have been achieved in the detection of objects that can be used in real time applications such as video. Due to the speed and performance provided, they were also used in large scale satellite imageries. Although producing the object proposals obtains successful results, there is a trade-off between the number of proposals with performance and speed. The number of your object proposals may indirectly affect the accuracy of your model or reduce the speed.

Afterward, You Look Only Once (YOLO) [18] and Single Shot Multibox Detector (SSD) [19] architectures were developed which turns the classification and localization steps of the object detection into a regression problem with utilization of a single neural network. These new structures overshadowed R-CNN architectures with achieving more accurate and fast results in major competitions like Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes(PASCAL VOC) [20] and Common Objects in Context (COCO) [21] challenges. Generally, a few of the researchers were able to directly experiment these deep learning techniques in the remote sensing area. The main reason is there are a lot of labeled natural images but very few with the satellite images. Therefore, these techniques are being tried and developed more in natural images than in remote sensing images (RSI).

In this thesis, the state-of-the-art object detection architectures examined and discussed the effects of the hyperparameters. For the detection tasks on large images, a framework developed and used a combination of the networks.

1.1 Purpose of Thesis

The main purpose of the thesis is obtaining very accurate and fast aircraft detector for large scale remote sensing images. For this purpose, the literature was scanned and the difficulties in this task were determined. After that, some of the state-of-art object detection architectures were trained and tested with different parameters with taking into account of the difficulties. A framework has developed for detecting objects more faster and accurate way.

The thesis is organized as follows : A detailed literature overview on geospatial object detection analysis with both traditional techniques and deep learning methods is presented in Chapter 2, theory of the learning and deep learning techniques are provided in Chapter 3, the methodologies of the thesis and architecture reviews are examined in Chapter 4 and the experiments and results conducted on airplane detection is discussed in Chapter 5. At last, the conclusion of this study, future works, and the opinions are shared in Chapter 6.

2. LITERATURE OVERVIEW ON GEOSPATIAL OBJECT ANALYSİS

Detection of objects from satellite imageries has been studied for decades. With the development of satellite sensors, spectral and spatial sensing capabilities, the quality of the analysis from the obtained images increases. Of course, the development of sensors has made a positive effect on the performance of the analyzes as well as the improvement of the applied methods. Developments and approaches used in remote sensing are examined in this section sequentially.

The beginning of earth surface observation and analysis is a research subject since the end of the 60s. Remote sensing images have been studied in many areas such as Land Use Land Cover (LULC) classification, vegetation indexing, environmental surveillance, geospatial object detection. Since the spatial resolutions of the remote sensing images at that time were too low, the focus was on pixel-level analysis.

In time, with the improvements of the spatial, spectral and radiometric resolutions in the remote sensing images, the performance of the pixel-level analyzes was not found sufficient. Instead, they focused on methods at the object level based on the spatial relations of each pixel. In this direction, researchers came up with the Object-Based Image Analysis (OBIA) [22] and Geographic Object-Based Image Analysis (GEOBIA) [23] to examine higher spatial resolution images. Although these methods catch some success, they were unable to extract semantic meanings. For example, they could not inform whether the image contained aircraft or vehicles. Or they could not distinguish the ships in the harbor or the ships on the high seas.

Because of the semantic information need from the images, the researchers conducted on different machine learning techniques to gain this ability. Before the utilization of the deep learning techniques in satellite images, handcrafted feature based supervised and unsupervised methods were widely used.

2.1 Handcrafted Features

Feature extraction is the process of revealing distinctive predominant features of the image according to the relationships of pixels in the raw image. The ability to describe the image with dominant characteristics is very useful and important for object detection performance. For this purpose, various feature extraction methods have been developed by the researchers which may be suitable for different object detection problems. In this section, the studies on the problem of object detection from satellite images which use handcrafted features were presented.

2.1.1 Texture features

Texture features aim to expose local density variances and patterns on the surface. In satellite imagery, it can give good results in problems where the sudden changes appear such as ship detection present in the high seas, airport detection, and vehicle detection.

Gabor wavelet can be given as a classic texture features. Gabor attributes are subtracted by filters calculated based on the determined spatial frequency and rotation. It provides a feature close to human visual perception. In the study of Polat E. and Yildiz C. [24], four different rotations and a frequency at which the sample images were most responsive were determined and according to these specifications, Gabor filters were applied to the airplane samples. The rate of detection was reached 91% and the false alarm rate (FAR) was 7.5%.

The local binary pattern (LBP) [25] which is another texture feature that is simply extracted by dividing the image into cells. Places, where the pixels are neighbors of the center pixel and greater than its value, are represented as one, and lower pixels are zero. The histograms of the cells are then calculated according to the frequency of these values and the LBP attribute is extracted by concatenating the calculated histograms. Grabner et al. use the LBP attributes, they attempted to detect vehicles from aerial images [26].

2.1.2 Scale invariant transform features (SIFT)

This approach transforms image data into scale invariant coordinate space in relation to the local features. The features are also invariant to rotation, clutter, lighting, and occlusion. Scale-invariant feature transform descriptor was firstly published by D. Lowe et al. and has become an important tool of computer vision tasks [4]. The extraction process is made by 4 steps; scale-space extrema detection, keypoint localization, assignment of the most relevant orientation and generating the keypoint descriptor from the oriented histograms.

For the remote sensing images, *Sirmacek. et al.* proposed a method with graph theoretical tool with SIFT to detect buildings and urban areas [27]. They suffered from low contrast between rooftops and the background with this method. Even so, the results were promising for such basic descriptor.

2.1.3 Histogram of oriented gradient features (HOG)

Histogram of oriented gradient feature just counts occurrences of gradient orientation in sub-regions of an image. It is similar to the scale invariant feature transform (SIFT) descriptors, although it is computed on dense grid cells and it utilizes the overlapping local contrast for better accuracy. It was first proposed by *Dalal N.* and *Trigs B.* in 2005 and was popular in many computer vision tasks [5].

The main idea behind the HOG descriptor is that the appearance and shape of local differences in an image can be described by the distribution of the intensity gradients or the directions of the edges. The image is divided into cells, and for the pixels in each cell, a histogram of gradient tendencies is compiled. The feature is the combining of them. To improve accuracy, local histograms can be normalized according to contrast while calculating the intensity over a larger area of the image, which is called as a block and then using this value to normalize all cells in the block. This normalization leads to a better invariance to changes in lighting and shading. In summary, HOG attributes are extracted at the end of a 4-stage process, such as making gradient calculations, orientating binning, creating descriptor blocks according to these bins and normalizing the blocks.

With its popularization, of course, it was also used with satellite images. *Chen et al.* did vehicle detection with HOG attributes after segmenting the roads with the line segment detector (LSD) algorithm [28]. *Kembhavi et al.* utilized from a multi-scale based classifier model with HOG features [29]. In this way, he was able to effectively identify vehicles of different sizes and scales. *Zhang et al.* introduced rotation invariant HOG descriptors which were used for object detection from satellite imagery [30]. Apart from these, an elliptical arc detector was used to detect oil tanks with the help of HOG features [31].

2.1.4 Bag of words features (BoW)

The main advantage of the Bag of Words (BoW) [32] features is its simplicity and invariance from viewpoints changes and background cluttering which was also adopted by the remote sensing researchers for good results on the image classification. For constructing the BoW model researchers generally, detect key points from the images as the first step with the key point detection methods such as Harris-Laplacian detector or Difference of Gaussian (DoG) detector [33]. After, there should be a second stage as local descriptor computation for the detected key points. The most popular descriptor is SIFT for this task. Then, for the computed descriptors, a visual vocabulary space should be constructed by a clustering technique like k-means. After that, vector quantization is applied to each key point into a visual word in the clusters. Lastly, a pooling steps that pools quantized local descriptors into a global histogram representation or feeding a classifier with them.

For solving the challenge of detecting geospatial objects with complex shapes from high-resolution images, sparse coding is used with BoW in Sun et al. study [34]. Cheng et al. utilized from probabilistic latent semantic analysis and k-nearest neighbor (k-NN) with Bow representations for the landslide detection problem [35]. Also, a spatial sparse coding bag of words (SSCBoW) model proposed for detection of objects which has much complex shape like aircraft [36]. The linear support vector machine (SVM) used as a classifier in that study and the proposed SSCBoW model overwhelmed the classical BoW model.

2.2 Unsupervised Learning Techniques

Excessive attention has been given to unsupervised techniques in object detection and classification studies from satellite imagery in the past decades. The process of creating labeled data is very costly and time-consuming. Therefore, unsupervised techniques have become a widely used method to avoid labeling data in remote sensing images and consequently come up with cost-effective solutions. In addition, in the early stages, the supervised techniques were not popular. This situation has also an important impact on preferring unsupervised methods.

The principal component analysis (PCA) method, which is one of the unsupervised techniques, is used as an aid in dimension reduction or in the selection of the features for the analysis of the remote sensing images (RSIs). As in the study of Liu et al., it was also used as a parametric shape extractor to recognize aircraft [37]. Chaitanya M. examined the k-means clustering method by using the SIFT features to the detection of car and aircraft objects [38]. Tang et al, along with the popularization of deep learning methods, tried to detect the ships from SPOT-5 optical satellite imageries by utilizing from a deep autoencoder structure [39]. Although good results have obtained compared to other methods, it has a disadvantage of high computational operations due to the fully connections between the autoencoder nodes.

2.3 Supervised Learning Techniques

The main advantage of supervised learning methods is that they are trained with data that has been previously labeled. Due to the fact that the data is labeled, the created model in comparison with the unsupervised methods can better generalize the sample space and thus, this gives more accurate results in the estimation of the new samples. In addition, although we can not obtain semantic information about objects in the unsupervised techniques, this comes up with the opposite for the supervised methods as another advantage.

In object detection from RSIs, support vector machine is the most popular and effective one for the supervised learning methods. Cheng et al. composed a mixture model by utilizing from SVM with the HOG features to detect airports and airplane [40]. Bi et al. generate the ship candidates from the binary saliency map of the training samples and the SIFT descriptors were extracted from them to feed the SVM model [41]. They used SPOT-5 panchromatic images and 3-stage detection process for computational efficiency.

Adaboost algorithm [42], as another supervised method, has played an important role in vehicle detection [43]. Aytekin et al. used 137 texture-based features in total to build a strong classifier with AdaBoost to detect airport runways [44]. Shi et al. generate a fake hyperspectral image from panchromatic satellite images with rearranging them into a vector to make it more observable for the relations of the adjacent pixels [45]. They produced ship candidates from these fake images and used them to train an AdaBoost classifier. Although they produced too many false detections near the land, they reached impressive results on the high seas.

K-nearest-neighbor (k-NN) is one of the simplest and conventional supervised technique that used in various RSI analyzes. Haapanen et al. used k-NN for determination of the forest, non-forest and water area from Landsat 7 ETM+ data [46].

In the cases that, the labeled data size is low, the researchers generally applied the weakly supervised methods. In the weakly supervised methods, data is also extracted from the unlabeled data with the help of labeled part. This is a noisy method of training, because it can be obtained false labeled data during the extraction. *Zhang et al.* proposed a weakly supervised method to detect airplanes, vehicles, and airports [47].

In the last few years, with the drastic improvements of the CNN architectures, they were utilized by remote sensing researchers for object detection studies [48] [49] [50] [51]. After the emergence of the architectures like SSD and YOLO which solve the detection problems with one network more accurate and fast, researchers have tended to use them. Radovic et al. worked on the detection of aircraft from the unmanned aerial vehicle(UAV) imageries with YOLO and achieved a 99.6 % precision rate [52]. Nie et al. used SSD to detect in various sizes of ships at inshore and offshore areas [53]. Wang et al. tried two sizes of the detector (SSD300 and SSD512) with SAR images for the same purpose [54]

3. FUNDAMENTALS OF MACHINE LEARNING

Machine learning is defined as the study area which gives the ability to learn without being programmed explicitly to computers. It is a computer program which orients itself to perform a task accurately by using the data. In other words, in order to accomplish a task in the background instead of the contiguous block of computer code, it learns information from the previous data and makes a decision automatically for the new given data by utilizing the power of the linear algebra, probability theory, and differential equations.

Machine learning is divided into two as supervised and unsupervised methods. We look at the learning principles of some of the methods before we get into the methodology section.

3.1 Unsupervised Learning Methods

There is no label information in unsupervised learning. It makes a distinction simply by grouping the samples in the data according to their similarity to each other. We will examine k-means clustering, hierarchical clustering, and autoencoders as the most important ones.

3.1.1 K-means clustering

One of the simplest tasks we can perform on an unlabeled data set is to find groups that are similar to each other which are called clusters. K-means is one of the most used clustering algorithms. It basically stores k centroids which are the center point of each cluster. If a sample is closer to the centroid of that cluster than any other centroid, it is considered to be in a particular cluster.

K-means algorithm tries to find the best k centroids by multiple iterations over the data and assign a class to each sample according to the distance between the centroids.

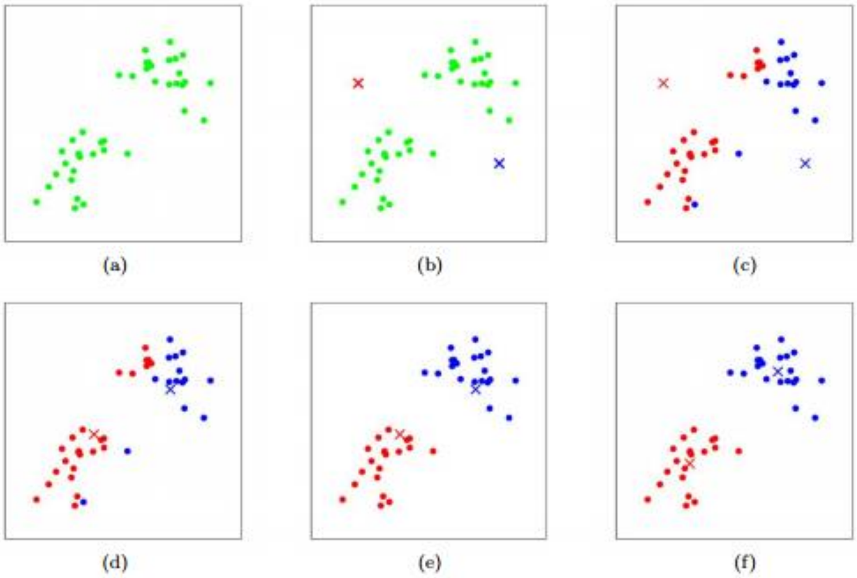


Figure 3.1 : K-means clustering algorithm. Data samples are represented as dots and cluster centroids as shown as crosses; (a) Original dataset, (b) Random initialization of cluster centroids, (c-f) Iterations to find clusters.

The algorithm simply does randomly defining k centroids at first. Then, it assigns each data samples to the cluster of the closest centroid. After that, the center of the clusters is recomputed and defined as the new centroids. Again, the closest samples are assigned to the new cluster centroids. These operations are done in each iteration until every sample converged and the centroids of the clusters do not change.

When we are given a training set $x^{(1)}, \dots, x^{(m)}$, where $x^{(i)} \in \mathbb{R}^{(n)}$ as usual and they could be feature vectors of each sample, our goal is to determine *k* centroids and a label $c^{(i)}$ for each data if we don't have label information for the given data. According to this knowledge, after the initialization of cluster centroids randomly as $\mu_1, \mu_2 \dots, \mu_k \in \mathbb{R}^{(n)}$, the k-means algorithm is as follows and repeats until convergence:

$$\text{For every i, set } \quad c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2 \tag{3.1}$$

$$\text{For every j, set } \quad \mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}} \tag{3.2}$$

3.1.2 Hierarchical clustering

Hierarchical clustering is done in two ways as agglomerative and divisive. As the name suggests, the agglomerative method (bottom-up) progress via merging the smallest groups until the top. Divisive method (top-down) splits the largest group and reaches each sample at the bottom. Therefore, hierarchical clustering is a greedy manner way to group unlabeled data. But it is useful to utilize from hierarchical clustering method in Natural Language Processing (NLP) tasks or image recognition tasks which have datasets with fine-grained classes.

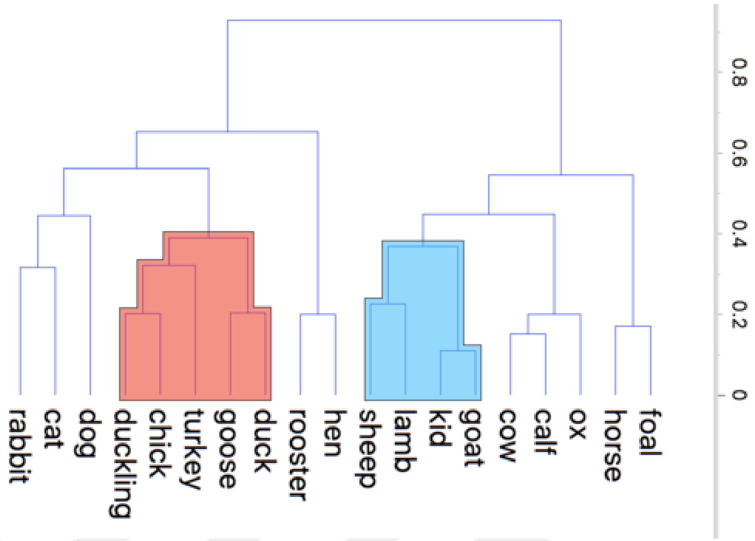


Figure 3.2 : Hierarchical agglomerative clustering dendrogram for the animal recognition. Axis y is the similarity metric.

For the agglomerative method, the merging operations are done by looking at the distance of two samples at the bottom to find the most similar ones. The distance metrics and merging methods vary for the tasks [53]. By merging them according to the similarity, a dendrogram of the data is created as shown in (Figure 3.2). As with the k-means method, it is also possible to determine how many sets of data should be separated. A similarity threshold is determined for this purpose and the part above this threshold is cut from the dendrogram.

3.1.3 Autoencoders

With the development of artificial neural network algorithms, autoencoders architectures are designed to be used in unlabeled data. When we assume we have unlabeled training examples as $x_{(1)}, \dots, x_{(m)}$, where $x_{(i)} \in \mathbb{R}^{(n)}$, an autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. It simply compresses the data in the encoding part of the network to hold the most useful features and tries to produce the same data as given at the input layer. By this way, it can discover the interesting structure about the data at the encoding phase and reconstruct it again at the decoding phase. But if the input data were completely random, this compression task would not be useful, because it can not find any correlation between the features. So, it is most effective with the structured data which have correlated features.

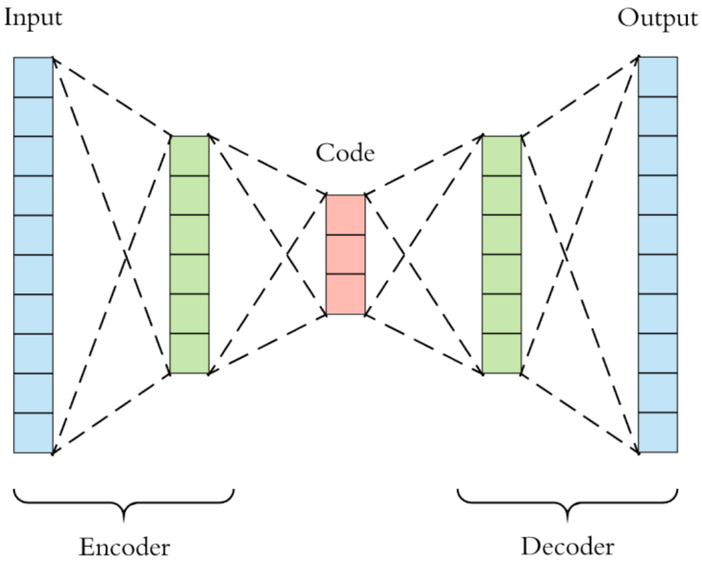


Figure 3.3 : An example of Autoencoder.

With the autoencoders, we can get the low-dimensional representation of the input data at the end of the encoding step which shows a similarity with the Principal Component Analysis (PCA). In a simple autoencoder structure, size of the input and output layers must be the same. Secondly, hidden layers must be symmetric about the center. The number of nodes for hidden layers must decrease from left to center and must increase from center to right as shown in (Figure 3.3).

In the encoding phase, the N-dimensional input vector is transformed to a K-dimensional feature vector by using non-linear function where $z_{(i)}$, is the K-dimensional encoded feature of N-dimensional sample $x_{(i)}$, b_1 is the bias vector, W_1 is KxN encoder weight matrix and $f(.)$ is an activation function which is mostly sigmoid function as given in equation 3.3.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

$$z_{(i)} = f\left(W_1 x_{(i)} + b_1\right) \tag{3.4}$$

In the decoding phase, a similar procedure is followed to reconstruct the input. As b_2 is the bias vector and W_2 is the weight matrix of decoding part, reconstructed data $x'_{(i)}$:

$$x'_{(i)} = f\left(W_2^T z_{(i)} + b_2\right) \tag{3.5}$$

Weight parameters of the autoencoder are learned by minimizing the calculation of loss function $J(X,\ X')$ between input and output data. Symbol λ is denoting the regularization parameter.

$$J(X,X') = \left(\sum_{i=1}^{M} \left\| x'_{(i)} - x_{(i)} \right\|^{2} - \lambda \|W\|^{2}\right) \qquad (3.6)$$

3.2 Supervised Learning Methods

In the supervised learning method, training dataset (x) also contains the label (y) information of each sample. Supervised methods try to learn a decision function so well, when it encounters a new sample that it can predict its label accurately. These methods can be used for 2 different problems as classification and regression. We will examine here more specifically the classification definitions of them which are used in remote sensing.

3.2.1 Support vector machine (SVM)

It is one of the most effective and simple methods used in supervised classification. For classification, it is possible to separate two groups by drawing a border between groups in the sample space. This border is called a hypothesis function or hyperplane and SVM tries to learn this function to separate samples according to categories which they belong to. The place where this function will be drawn should be the most distant from the members of both categories.

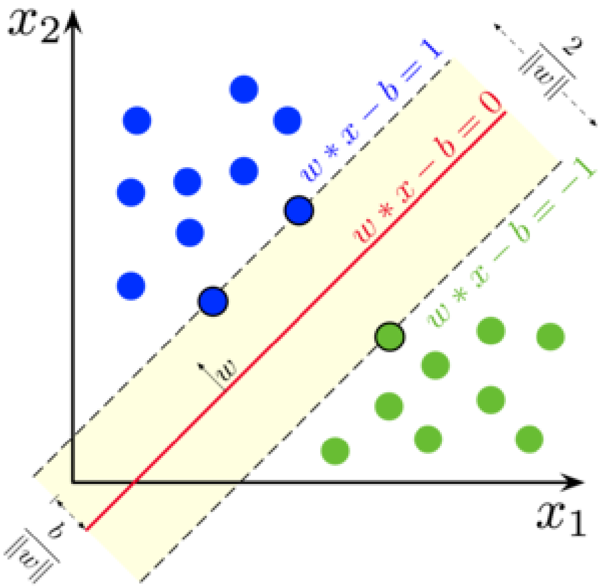


Figure 3.4 : An example of Linear SVM classifier for two classes in 2D feature space.

SVM classifiers are usually used for classes that can be separated linearly. Let's say that the images that contain two classes (e.g pool and building samples) will be used for the classification. For this, feature vectors are extracted from each of the sample images. These features can be designed and extracted with creating custom filters according to the tasks which are called as handcrafted features (HOG, LBP, Gabor, etc.). Either, by combining several feature vectors can be used for obtaining more representative features to use them with the SVM classifier.

In the SVM algorithm, the objective is to define such a hyperplane between the feature vectors extracted from the samples, so that you completely separate both classes. For this aim, the hyperplane between two class samples must be determined with a margin between the feature vectors of samples which is maximum. If y is a label vector, W is the weight matrix and x is the feature matrix of the samples, the hypothesis function can be defined as:

$$f(x) = W^T x + b = y \tag{3.7}$$

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

$$\text{Margin } m = \frac{2}{\|W\|} \tag{3.8}$$

By maximizing this margin the optimum hypothesis function could be determined. With the SVM, it classifies with one-to-all method when there is more than two class in the classification task. If the samples are not linearly separable, the polynomial hypothesis function should be used with the SVM algorithm.

3.2.2 Adaboost algorithm

The adaboost algorithm focuses on classification problems and aims to convert a set of weak classifiers into a strong one. Suppose we given a N-size training set $(X, Y) = \{(x_1, y_1), \ldots, (x_N, y_N) \mid x_i \in \mathbb{R}^{(n)}, y_i \in \{-1, 1\}, \mathrm{i} = 1, \ldots, \mathrm{N}\}$ where x_i is a feature vector of i-th sample and y_i is its label. The weights of all examples are initialized as $W_i = 1 / N$ and a set of weak classifier $f_m(x)$ are trained by weighted least-squares fitting on the labels Y. The weights are updated by $W_i \leftarrow W_i \,. \exp(-y_i f_m(x_i)) / Z$, with the normalization parameter Z which is defined by $Z = \sum_{i=1}^{N} W_i$. At each iteration, a new weak classifier is added and the process continues until a certain stopping condition obtained. We can specify this condition with M number of a weak classifier. Consequently, a strong classifier $F(x)$ is computed as a linear combination of all weak classifier.

$$F(x) = \sum_{m=1}^{M} f_m(x) \tag{3.9}$$

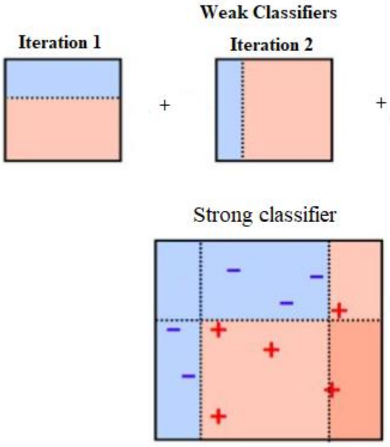


Figure 3.5 : Illustration of obtaining strong classifier with Adaboost algorithm.

3.2.3 K-nearest neighbor (k-NN)

The k-NN is just making a decision by looking at the new coming sample at the sample space of the training data and determine which class is most closest to this sample. The k represents the number of the closest data point, so it can be thought as a parameter which the new data fit this condition to belong to a class.

Algorithmically, suppose we have a N-size training set as $(X, Y) = \{(x_1, y_1), \dots, (x_N, y_N) \mid x_i \in \mathbb{R}^{(n)}, y_i \in \{1,2, \dots, \mathrm{C}\}, \mathrm{i} = 1, \dots, \mathrm{N}\}$ where x_i is a feature vector of i-th sample and y_i is its label when we have C classes. Given a test sample $x' \in \mathbb{R}^{(n)}$, utilizing by a distance function $f_{x'}$, which can be an Euclidian distance as $f_{x'} = \|x - x'\|$, a set of training samples X can be ordered to obtain k nearest neighbors of the test sample as $X' = \{x'_1, x'_2, \dots, x'_k\}$ with their corresponding labels $Y' = \{y'_1, y'_2, \dots, y'_k\}$. After that , the kNN classifies the test sample x' by applying a majority voting rule to these closest training samples :

$$\underset{j=1,\dots,C}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(y'_i, j) \tag{3.10}$$

where δ defined as :

$$\delta(i,j) = \begin{cases} 0 \ if \ i \neq j \\ 1 \ if \ i = j \end{cases} \tag{3.11}$$

Although, the k-NN algorithm is the simple supervised learning algorithm for classification by just having one parameter *k* to tune the system, choosing it large number, can decrease the performance and can be a time-consuming task. Also, for classifying new coming sample, you have to keep your whole training data in the memory or apply some sufficient statistics methods to imitate them.

3.2.4 Artificial neural networks (ANN)

Artificial neural network algorithms have been developed by mimicking biological neuron cells. When the signal transmission from one cell to the other in a biological neural network is over a certain threshold level, the impulse passes through the axons of the previous cell to the dendrites of the other neuron by chemical activation. Thus, the signal is transmitted from its axons to the next neuron at a certain intensity. Artificial neural networks have a similar situation. The neurons in the entrance form the input layer, the neurons in the middle create the hidden layer(s) and the neurons, in the end, form the output layer. There are links between the previous layer and each neuron with the next layer. These links represent the weight matrices. The numbers of hidden layers and the number of cells in layers vary according to the complexity of the classification process.

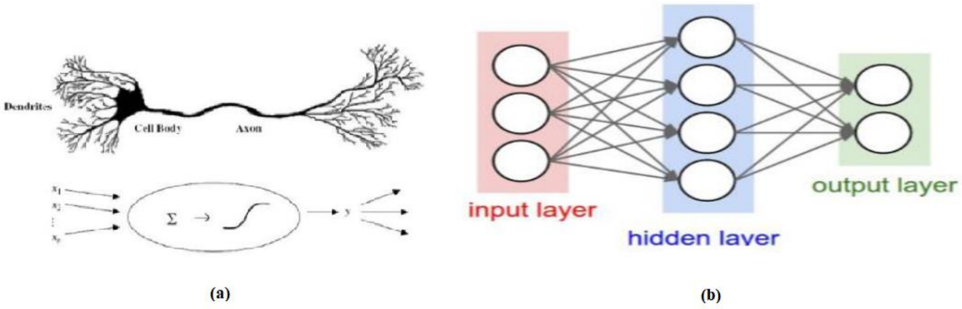


Figure 3.6 : Illustration of (a) biological and artificial neuron (b) artificial neural networks.

The input layer usually contains the cells up to the input data size of a sample in the training dataset. The number of cells in the output layer depends on the number of classes.

A data set consisting of previously labeled data is required to train the network. During the training phase, a loss function (in other words objective function) is calculated by the help of feed forward and back propagation algorithms and by using the weight matrices in the layers, by comparing the predicted result with the ground truth label of the input data. According to this calculated loss, the weight matrices are updated again in each iteration with the backpropagation algorithm.

3.2.4.1 Convolutional neural network (CNN)

Convolutional neural networks are very similar to ordinary artificial neural networks They are made up with neurons which form the filters that have learnable weights and biases. The main difference of CNN is that the architecture is designed to receive multidimensional data as input instead of 1D data.

For the computer vision tasks, during the training phase with CNN structures, each feature maps is extracted from the images with a certain numbers of filters. In the last layers, we use the fully connected layers to classify or score the input data. These structures basically learn the filters that can extract best features which describe the image categories well and by this way it can predict which classes they belong to, and in doing so, utilize feed-forward and back-propagation algorithms.

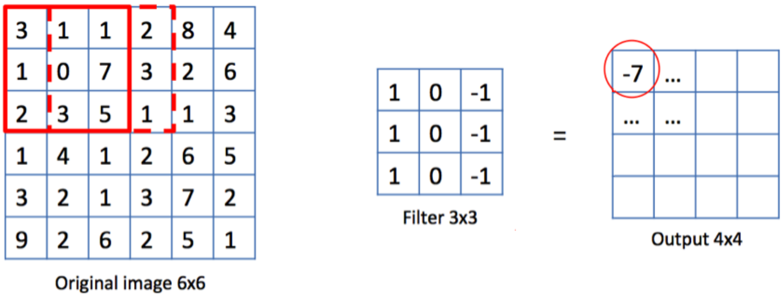
The CNN architectures are generally comprised of 4 structures as convolutional layers, pooling, and fully connected layers. Some of the most commonly used terms will also be examined here.

Convolutional layers

The convolution is a point-wise integral operation between two functions in calculus. For the CNN architectures convolution layers takes the role of feature map extraction from the input data by doing point-wise product operations. Parameter of these layers consists of a set of learnable filters which they also called as kernels. These kernels can gather structured information from the data by obtaining the feature maps. The convolutional layers have basically size of $c\ x\ c\ x\ d$, c is the size of the kernels and d is the number of kernels. When we pass an input map X, with a size of $m\ x\ m\ x\ k$ through the convolution kernels, m is the width and height of the input and k is the channel number, we obtain Y as a feature map at the end of the 2D convolution operations with a size of $\left(\frac{m-c+2p}{s}+1\right)\ x\ (\frac{m-c+2p}{s}+1)$. Here, p is the padding size of the input and s is the stride size of the filter.

If we consider W is the parameter matrix of convolution layers (suppose parameters of filters as weight), * is 2D convolution operation, $f(.)$ is the non-linear activation function and b is the bias parameter. We can define the feature map of the j-th convolution layer as in equation 3.12.

$$Y^j = f\left(W^j * X + b^j\right) \tag{3.12}$$



3	1	1	2	8	4
1	0	7	3	2	6
2	3	5	1	1	3
1	4	1	2	6	5
3	2	1	3	7	2
9	2	6	2	5	1

Original image 6x6

1	0	-1
1	0	-1
1	0	-1

Filter 3x3

=

-7	...		
...	...		

Output 4x4

Figure 3.7 : 2D convolution opertion with zero padding, one stride and 3x3 filter.

After the convolutional operations, a non-linear activation function is applied to the feature map to prevent the non-linearity. Rectified linear unit (ReLU) is one of the most used activation function for this aim. It has a mathematical formula as $f(x) = max(0, x)$, which simply makes the negative values zero. It can be seen in (Figure 3.7) the first element of the feature map will be zero by applying this function.

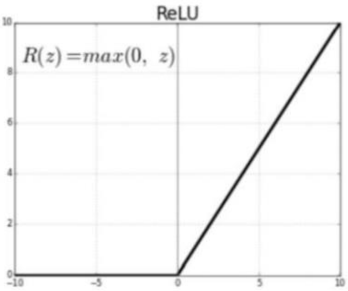
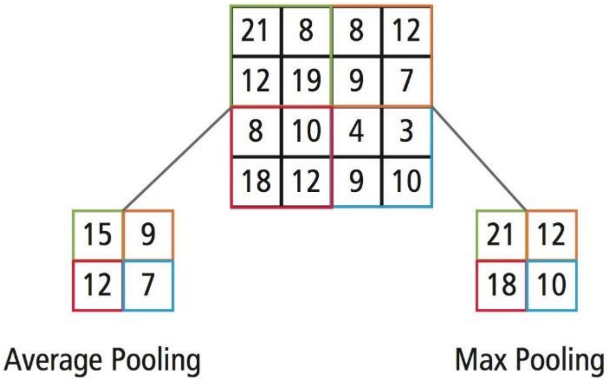


Figure 3.8 : Rectified linear unit function.

Pooling filters

Pooling operation is used for down-sampling between the convolution layers. They basically do spatial dimension reduction of the feature maps by pooling by keeping the most of the spatial information although these operations are lossy. There are two types of pooling filters that are mostly used by deep learning communities. One of them is average pooling which just simply does the average of the feature map underlying the mapping frame. The other one is the max pooling which is similarly getting the maximum value of the feature map underlying the frame. As in convolutional layers, the pooling layers have a structure similar to that of $c\ x\ c\ x\ d$, where c is the size of the kernels, d is the depth (number of kernels) and they applied with a stride number s.



21	8	8	12
12	19	9	7
8	10	4	3
18	12	9	10

15	9
12	7

21	12
18	10

Figure 3.9 : 2x2x1 average and maximum pooling filters with 2 stride number.

Fully connected layers

The output of the convolutional layers produces high-level features of the data. After the feature extraction, we need to classify the data into various categories. This can be done by fully connected (FC) layers. As the name suggests, all of the neurons in the previous layer are completely linked to neurons of the fully connected layer. Fully connected layers could make the output of the last convolutional layer flattened and connected to the output layer of the network architecture. This is the easiest way of learning a non-linear combination of the features.

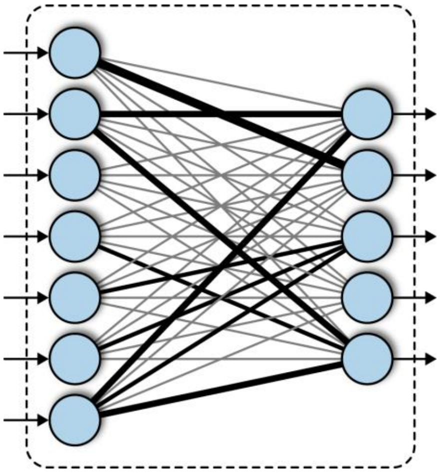


Figure 3.10 : Illustration of fully connected layers, last layer as the classifier layer.

It can be seen the illustration of the FC layers in (Figure 3.10), the last layer as a classifier and the links represents the weights. Some of the weights are shown thicker that can be imagined as the relation between those neurons are much strong. At the last fully connected layer, the input data will be classified by firing most related neuron with considering the weights.

4. METHODOLOGY AND ARCHITECTURE REVIEW

In this thesis, we worked with the state-of-the-art object detection models to increase the speed and accuracy for the detection of aircraft objects by taking into account differences of satellite images from natural images. Unlike natural scenes, some of the airplanes, that we detected in satellite images can be very small compared to the field of view. In addition, even if they have fewer perspective differences as being just collected from above, a dataset has to be created taking into account of nadir-angle of the satellite. Also, atmospheric conditions and sun angles are considered. In our detection framework, we used Faster R-CNN, SSD and YOLO object detection networks. Although the accuracy is very important, it must be taken into account that the framework needs to process very large scale satellite images quickly. For this aim, a solution proposed in this chapter and the architectures are examined. The training process and the results of the processes of each object detection model are presented in the next chapter.

4.1 Region Proposal Network : Faster R-CNN

Faster R-CNN is one of the most used object detection networks which achieved accurate and quick results with CNN structures. It is started to use for nearly real-time applications such as video indexing tasks because of these capabilities.

Faster R-CNN is progressively developed over time. The first version of it, the R-CNN, basically uses a selective search algorithm, that utilizes a hierarchical grouping method to produce object proposals. It produces 2000 proposed object as the rectangular boxes and they are passed to a pre-trained CNN model. Then, the feature maps of them are extracted from the CNN model to pass them to an SVM for classification.

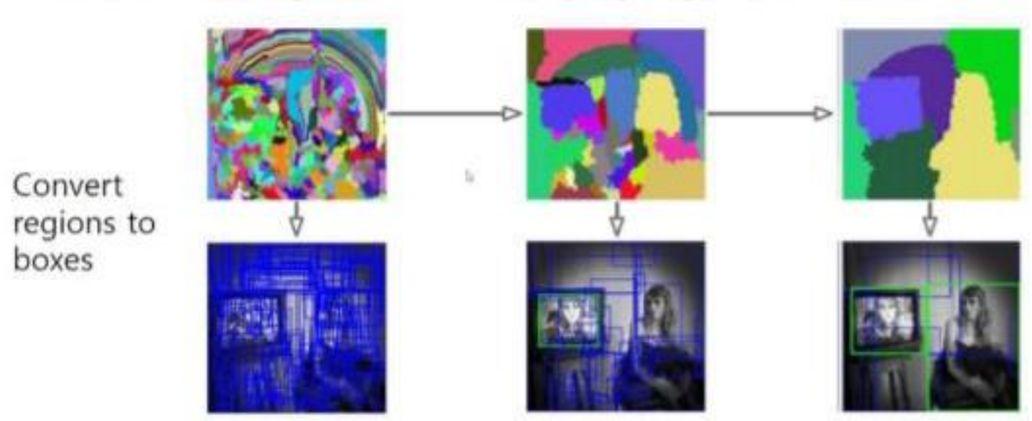


Figure 4.1 : Selective search algorithm, bottom-up segmentation, merging regions at multiple scales.

In 2015, *Girshick R. et al.* came up again with the Fast R-CNN which moves the solution one step forward. The main difference of the Fast R-CNN is just producing the object proposals from the feature map of the CNN, instead of getting them from the whole input image. By this way, there is no need to apply CNN process for 2000 times to extract feature maps. Then, the region of interest (ROI) pooling is applied to ensure to get standard and pre-defined output size. Finally, they are classified with a softmax classifier and made bounding box localizations with linear regression.

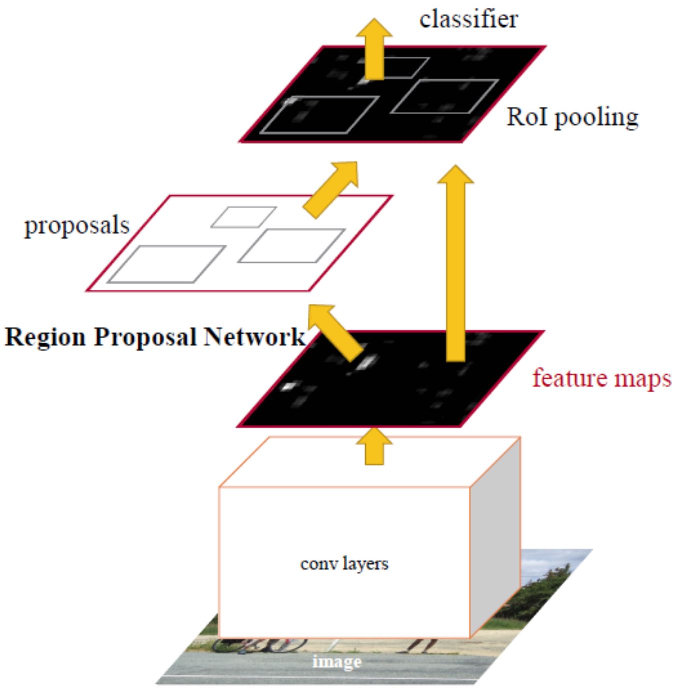


Figure 4.2 : Processes of the Faster R-CNN.

In the Faster R-CNN, selective search method is replaced by a region proposal network (RPN), which is aiming to learn to propose an object from the feature maps. The RPN is the first stage of this object detection method. As shown in (Figure 4.2), feature maps extracted from a CNN is passed to the RPN for proposing the regions. For each location of the feature maps k anchor boxes used for generating region proposals. The anchor box number k is defined as 9 considering the 3 different scales and 3 aspect ratios in the original paper [19]. With a size of $W\ x\ H$ feature map, there are $W\ x\ H\ x\ k$ anchor boxes in total that comprised of the negative (not object) and positive (object) samples. This means there are many negative anchor boxes for an image and to prevent the bias occurrences because of this imbalance, the negative and positive samples are chosen randomly by 1:1 ratio (128 negative and 128 positives) as a mini batch. The RPN learns to generate the region proposals at the training phase by utilizing these anchor boxes by comparing the ground truth boxes of the objects. A bounding box classification layer (cls) of the RPN, outputs $2\ x\ k$ scores whether there is an object or not object for k boxes. A regression layer is used to predict $4\ x\ k$ coordinates (center coordinates of box, width, and height) of k boxes. After generation of the region proposals, the ROI pooling operation is done as in the Faster R-CNN at the second stage of the network. Again as in Fast R-CNN, an ROI feature vector is obtained by fully connected layers and this vector classified by softmax to determine which category it belongs and a box regressor is applied on it to adapt the bounding box of that object.

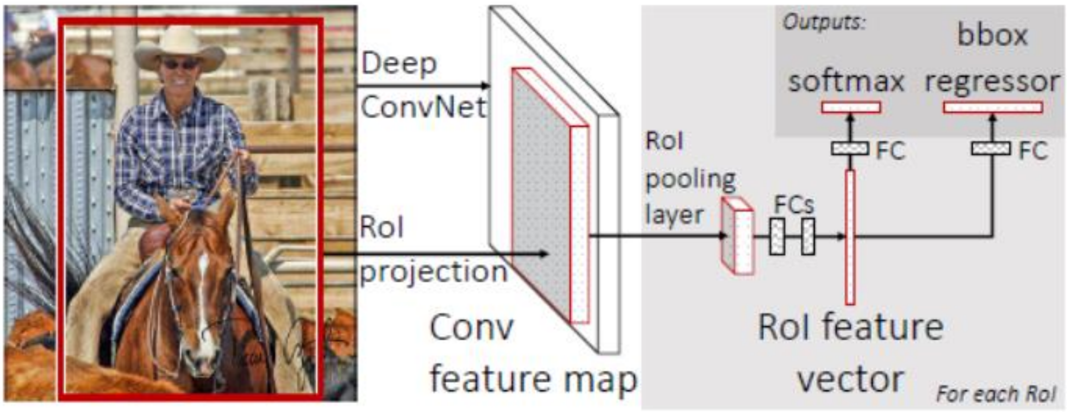


Figure 4.3 : Faster R-CNN detection network.

In our study, the Faster R-CNN is used with a residual neural network (ResNet) that comprised of 101 residual layers. Because they won the COCO 2015 challenge by utilizing the ResNet-101, instead of VGG-16 in Faster R-CNN. Also, we add one more additional scale parameter for generating the anchor boxes to detect smaller airplanes (4 scales, 3 aspect ratios, $k = 12$).

4.1.1 Loss function

The loss function of the RPN network for an image is defined as :

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.1}$$

Here, i is the index of an anchor, p_i is the prediction probability of anchor i being an object, p_i^* is the ground truth label and it is 1 if the anchor is an object, is 0 if the opposite. L_{cls} and L_{reg} represent respectively the classification loss which is a log loss over two classes (object or not object) and the regression loss is smooth L_1 function used for t_i and t_i^* parameters. t_i is a vector representation of predicted bounding box, t_i^* is ground truth bounding box associated with a positive anchor. Lastly, the parameter λ is used for balancing of the loss function terms, N_{cls} and N_{reg} are the normalization parameter of the classification and regression losses according to the mini batch size and anchor locations.

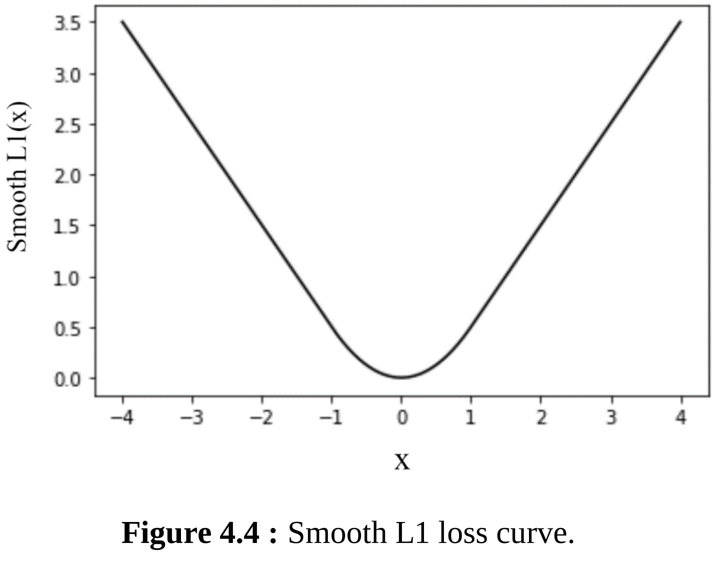


Figure 4.4 : Smooth L1 loss curve.

4.1.2 Residual blocks

When the CNN networks are getting a deeper structure degradation problems can occur. As the architecture deepens, the layers of the higher level can just act as an identity function. The output of them which are the feature maps are more similar to the input data. This causes the accuracy gets saturated and then degrades rapidly. For solving this problem the residual blocks come to help. The idea is, instead of learning from a direct mapping of $x \rightarrow y$ with a function $H(x)$, the residual blocks just changed it as $H(x) = F(x) + x$, where $F(x)$ and x represent the stacked non-linear layers and identity function respectively.

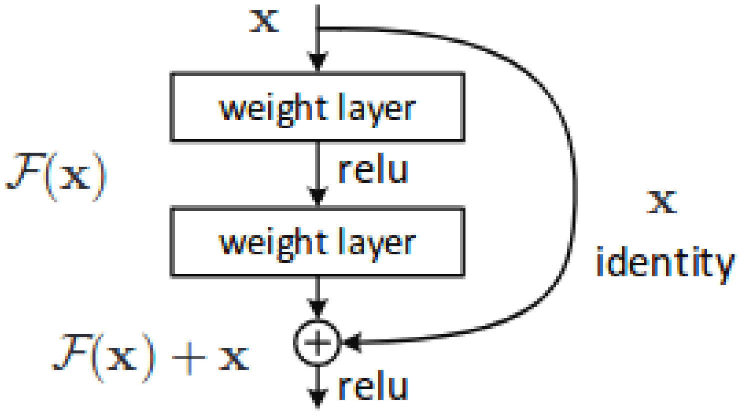


Figure 4.5 : Residual blocks.

The ResNet-101 is built by these residual blocks to achieve more accurate results.

4.2 Single Shot Multibox Detector (SSD)

We used SSD, which is a form of a single convolutional neural network. It is working with the corporation of extracted feature maps and generated bounding boxes which are called as default bounding boxes. The network simply does the loss calculation, by comparison, the offsets of the default bounding boxes and predicted classes between the ground truth values of the training samples at every iteration with trying different filters. After that, it updates all the weight parameters according to that calculated loss value with a back propagation algorithm. By this way, it tries to learn best filter structures to be able to catch the features of the objects well and generalize all the training samples for reducing the loss value and attaining high accuracy at the evaluation phase.

In the SSD method, a state-of-the-art CNN architecture used as a base network for feature extraction with the additional convolution layers which produce a various scale of feature maps to not miss detection of objects with different scales. Also, SSD allows different aspect ratios for generating default bounding boxes. By this way, the predicted boxes can wrap around the objects in a tighter and more accurate fashion.

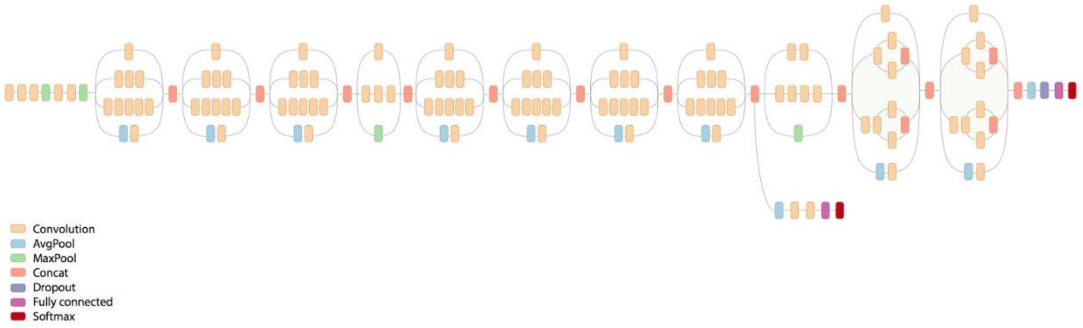


Figure 4.6 : InceptionV2 architecture.

At the original SSD paper, they used VGG-16 as a base network, but we ran the Inception-v2 model to have higher precision and faster detection speed. This is because the Inception-v2 has a deeper structure than VGG models but also has fewer parameters with thanks to the inception modules which comprised concatenating of multiple convolution layers. For example, GoogleNet, which is the first network that we encountered with the inception modules, employed only 5 million parameters which represented a 12x reduction with respect to AlexNet and it gives slightly more accurate results than VGG. Furthermore, VGGNet has 3x more parameters than AlexNet [6].

The structure of the inception modules in the middle of the network provides real success. As it can be seen in (Figure 4.7), instead of applying sequential convolutional layers as in the traditional CNN architectures, first of all, the features are extracted from the previous layer by combining a 1x1 convolution which is aiming to make dimension reduction and two different convolution operations as the size of 5x5 and 3x3 are derived from it. All of them are put together and the next inception module is passed with this concatenated form.

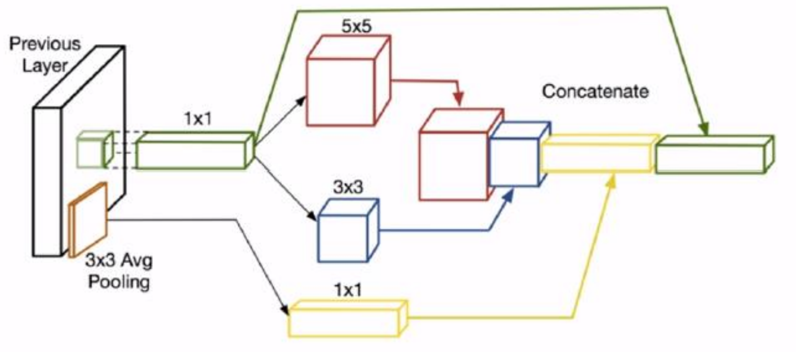


Figure 4.7 : Illustration of the inception modules.

As shown in (Figure 4.8), last inception modules of Inception V2 network, used as the feature generator with six different scales for our detection network. Each extracted feature maps can produce a fixed set of detection predictions as it is indicated at the end of the architecture. It is followed by a non-maximum suppression (NMS) algorithm to yield final detections. For the feature map of size m x n with p channels and k pieces of default bounding boxes, there would be m x n x k numbers of prediction calculated for class scores and predicted bounding box offsets. At the training step, this number also multiplied with batch size, denoted by BS in (Figure 4.8) for every iteration.

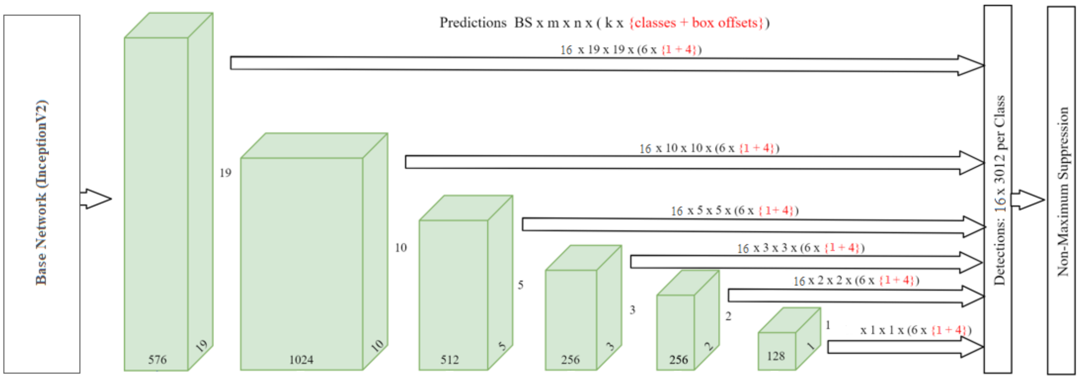

Base Network (InceptionV2)
Predictions BS x m x n x (k x {classes + box offsets})
16 x 19 x 19 x (6 x {1 + 4})
16 x 10 x 10 x (6 x {1 + 4})
16 x 5 x 5 x (6 x { 1+ 4})
16 x 3 x 3 x (6 x {1 + 4})
16 x 2 x 2 x (6 x {1 + 4})
x 1 x 1 x (6 x {1 + 4})
576
19
19
1024
10
10
512
5
5
256
3
3
256
2
2
128
1
1
Detections: 16 x 3012 per Class
Non-Maximum Suppression

Figure 4.8 : SSD architecture uses Inception V2 as a base network with 16 of batch size for training.

4.2.1 Default bounding boxes and negative sample generation

During the training, there should be determined the default bounding boxes correspond to which ground truth sample. The network is trained according to this simple rule. For each ground truth box, it is selected from default boxes that vary over the location, aspect ratio, and scale. The main purpose is to match each ground truth box to default bounding box with the best jaccard overlapping through higher than 0.5 thresholds. This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping default boxes rather than requiring it to pick only the one with maximum overlap.

To handle different object scales, SSD utilizes feature maps extracted from several different layers in a single network. For this aim, a fixed number of default bounding boxes must be produced at different scales and aspect ratios in each region at each extracted feature maps. We set six levels of aspect ratios with supposing $a_r \in \{1, 2, 3, 1/2, 1/3\}$ and s_k is the scale of the k-th square feature map for generating default boxes.

The sixth one is generated for the aspect ratio of 1 with a scale of s'_k=$\sqrt{s_k s_k+1}$. So, the width (w_k^a=$s_k\sqrt{a_r}$) and height (h_k^a=$s_k\sqrt{a_r}$) can be computed for each default box. As illustrated in (Figure 4.9), we can see how generated default bounding boxes on 5 x 5 feature map can be represented on the input image and overlap with the possible objects.

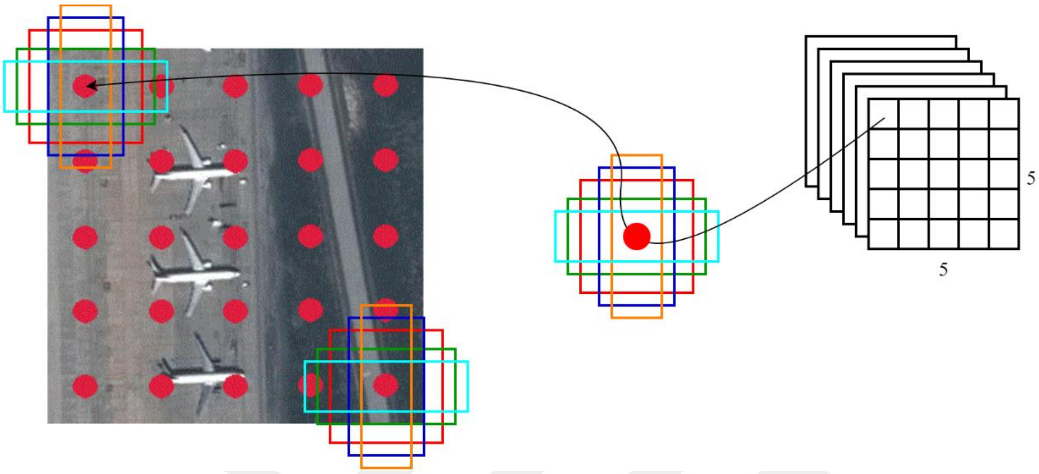


Figure 4.9 : Illustration of 5x5 feature map and generated default bounding box with 6 aspect ratio.

After the matching step, which is done at the beginning of the training, most default boxes are determined as negatives, especially when the number of possible default boxes is large. Instead of using all the negative examples, for protecting the balance with the positive examples, they sorted using the highest confidence loss for each default box and picked the top ones so that the ratio between the negatives and positives is at most 3:1. This ratio was found as it provides faster optimization and more accurate training [13].

4.2.2 Loss function

The loss (objective) value calculated as combining the confidence of the predicted class scores with the accuracy of the location. The total loss value (localization loss + confidence loss) is calculated as follows, which is an indication of the pairing of the *i*-th default box with j-th ground truth box of class p such that $x_{ij}^p = \{1,0\}$:

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right) \tag{4.2}$$

When we suppose N is as the number of matching default boxes, if there is no match (*N = 0*), the total loss is determined as zero directly. The α value is the balance of two types of losses and it is equal to 1 during the cross-validation phase. The localization loss is calculated as Smooth L1 loss between the offsets of the predicted box (*l*) and the ground truth box (*g*). If the center location of the boxes denoted as *cx, cy*, the default boxes *d*, width *w* and height *as h*:

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,\ cy,\ w,\ h\}} x_{ij}^{k} smooth_{L1}(l_i^m - ĝ_j^m) \tag{4.3}$$

$$ĝ_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^{w} \quad ĝ_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^{h} \tag{4.4}$$

$$ĝ_j^{w} = \log\frac{g_j^w}{d_i^w} \quad ĝ_j^{h} = \log\frac{g_j^h}{d_i^h} \tag{4.5}$$

And the confidence loss (*c*) is calculated as softmax loss of the predicted class relative to other classes:

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log \hat{c}_i^p - \sum_{i \in Neg}^{N} \log(\hat{c}_i^0) \tag{4.6}$$

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{4.7}$$

4.3 You Look Only Once v3 (YOLO)

Yolo-v3 is grounded upon the custom CNN architecture which is called as DarkNet-53 [55]. In the first version of it, the Yolo-v1 architecture is shown in (Figure 4.10) which is inspired by the GoogleNet, does basically downsampling the image and at the end, it produces final predictions from a tensor. This tensor is obtained in a similar way as in the ROI pooling layer of the Faster R-CNN network.

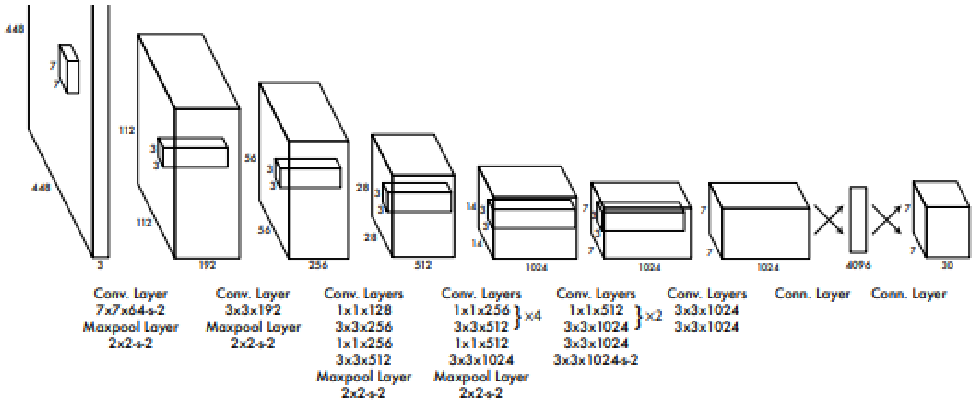


Figure 4.10 : Yolo-v1, architecture has 24 convolutional layers followed by 2 fully connected layers.

Yolo-v2 used a 30 layer architecture which is made up from Darknet-19 and additional 11 layers for object detection adaptation. This new structure made it more accurate and faster but it often struggles with the small objects in the field of view. Also, it does not utilize from the advantages of the residual blocks or up-sampling operations while Yolo-v3 does.

Yolo-v3 has a fully convolutional architecture which uses a variant of Darknet that it has 53 layers trained on the ImageNet classification dataset. For the detection tasks, they add 53 more layers onto it and train it with Pascal VOC dataset. With this way, they beat most of the detection algorithm while it is still fast for real-time applications. By the help of the residual connections and upsampling, it can do detections at 3 different scales from the specific layers of the structure. This makes it much better at the detection of smaller objects but slower than the previous versions because of the complexity.

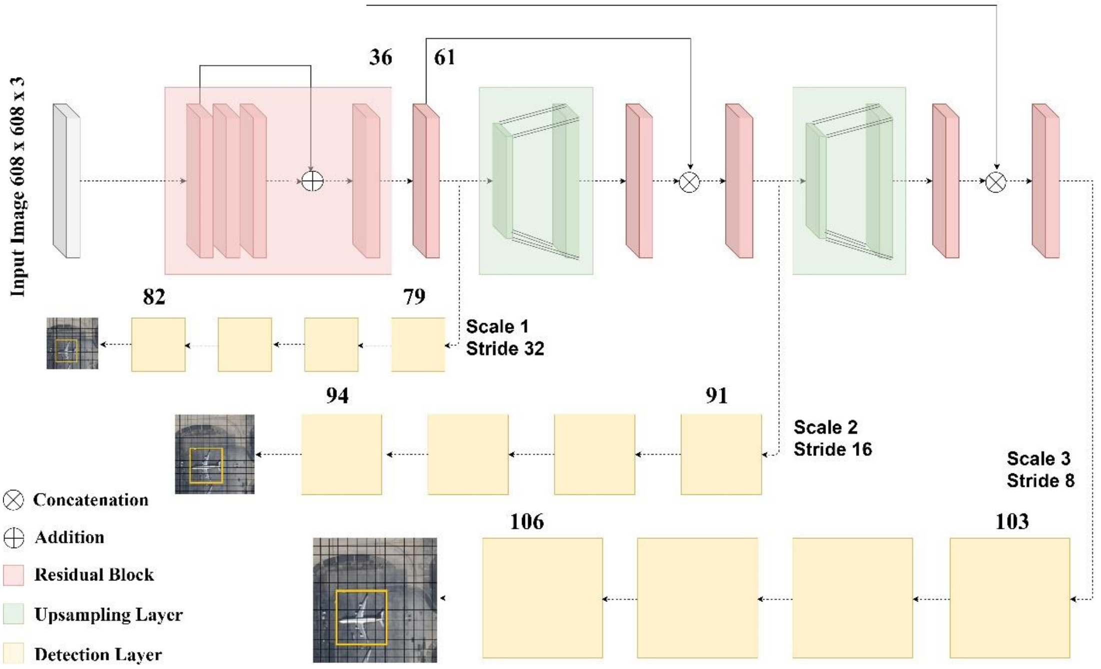


Figure 4.11 : Yolo v3 architecture.

The shape of the detection kernel is $1\ x\ 1\ x\ (B\ x\ (5\ +\ C))$. In v3 network, 9 pieces of the anchor are used for detection, 3 for each scale. Here B is the number of the anchors on the feature map, 5 is for the 4 bounding box offsets and one for object confidence. C is the number of categories. In our study, we use the Yolo-v3 network and the class is the only airplane, so the detection kernel shape will be $1\ x\ 1\ x\ (3\ x\ (5\ +\ 1))$ for each scale. The first detection process is made from the 82nd layer after the first 81 layer down-sampled the input image by size of 32 strides. If we have an input image of $608\ x\ 608$, that will be output as a feature map of $18\ x\ 18$ at that layer. This means we have $18\ x\ 18\ x\ 18$ detection features from this layer. After the first detection operation, the feature map is up-sampled by 2. This up-sampled feature map is concatenated with the feature map coming from the 61st layer. Then, a few $1 x 1$ convolution opeartions are performed to fuse features and reduct the depth dimension. After that, the second detection is made from the 94th layer which returns the detection feature map of $36 x 36 x 18$. The same procedure is done for the third scale at the 106th layer which yields a feature map of size $72 x 72 x 18$. This means, it produces 20,412 predicted boxes for each image. As in the SSD network the final predictions are proposed after NMS algorithm applied.

4.4 Detection Flow

While the usual sliding window technique slides the whole image at a fixed sliding step, it cannot ensure that the windows cover the objects exactly. Also, small sliding steps cause huge computation cost and making it too large, decreases the accuracy.

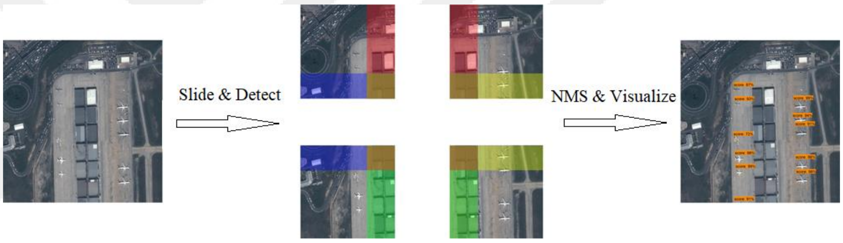
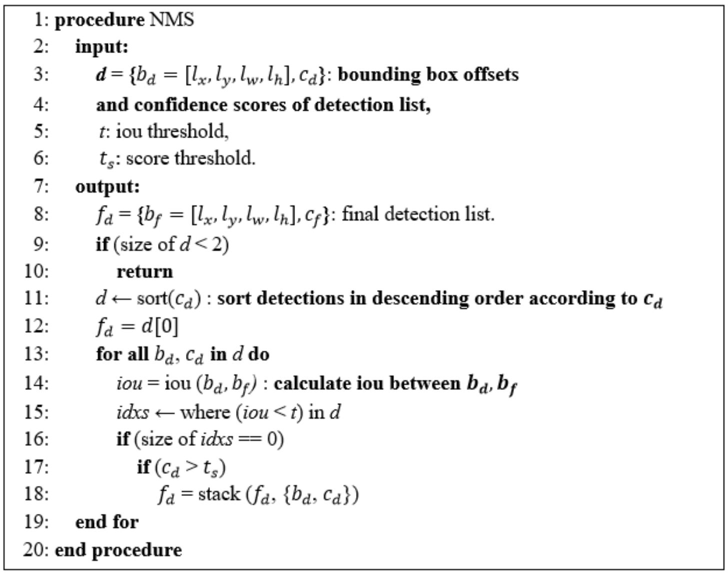


Figure 4.12 : Illustration of sliding detection flow.

As shown in (Figure 4.12), we create a detection flow with a sliding window approach with optimized sliding step for better accuracy and faster detection. When we just slide with input size of the detection networks, the objects at the edges of the window can not be detected or the bounding box offsets of them will be not correct. To tackle this problem, we determine an overlapping area between two windows according to the average size of the bounding boxes of the aircrafts in the prepared dataset as 144 pixels which fits well for good detection accuracy and speed. In the sliding process for an image with a certain overlap, *k x l* windows obtained to detect by SSD for horizontal and vertical directions respectively.

$$k = \left[\frac{height - overlap}{ssd\ height - overlap}\right] \tag{4.8}$$

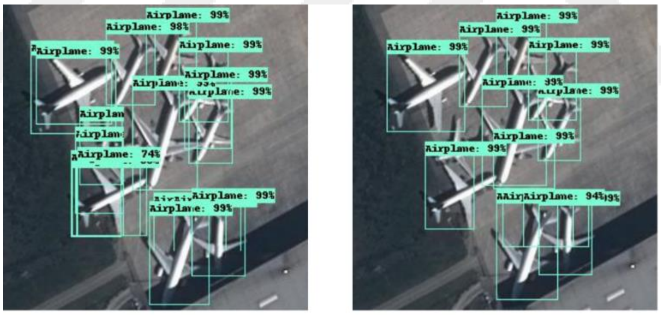
$$l = \left[\frac{width - overlap}{ssd\ width - overlap}\right] \tag{4.9}$$



```
procedure NMS
  input:
    $d = \{b_d = [l_x, l_y, l_w, l_h], c_d\}$: bounding box offsets
    and confidence scores of detection list,
    $t$: iou threshold,
    $t_s$: score threshold.
  output:
    $f_d = \{b_f = [l_x, l_y, l_w, l_h], c_f\}$: final detection list.
    if (size of $d < 2$)
      return
    $d \leftarrow \text{sort}(c_d)$ : sort detections in descending order according to $c_d$
    $f_d = d[0]$
    for all $b_d, c_d$ in $d$ do
      $iou = \text{iou}(b_d, b_f)$ : calculate iou between $b_d, b_f$
      $idxs \leftarrow$ where $(iou < t)$ in $d$
      if (size of $idxs == 0$)
        if $(c_d > t_s)$
          $f_d = \text{stack}(f_d, \{b_d, c_d\})$
  end for
end procedure
```

Figure 4.13 : Non maximum suppression algorithm (NMS).

After the sliding and detection step, we used the NMS algorithm to eliminate multiple detection occurrence over an object in the overlapping regions and we also applied score thresholding to decrease the number of the false detections.



(a) (b)

Figure 4.14 : Detection results of occluded objects (a) without NMS algorithm, (b) with NMS algorithm.

5. EXPERIMENTS AND RESULTS

The experiments and results are demonstrated in this section. After the training processes were done, we did the evaluation for training and test set which were prepared from DOTA dataset and for 5 large scale airport images with all trained networks. We prepared the ground truth annotations and evaluation results as COCO format to use COCO performance metric API to get more insights about the networks. The results section provides information about the API.

5.1 Dataset

We used DOTA (A Large-scale Dataset for Object Detection in Aerial Images) dataset for training and testing purpose. It is an open source object detection dataset for remote sensing and collected by Google Earth and China Centre for Resources Satellite Data and Application by satellite JL-1 and satellite GF-2. It contains 15 object categories as airplane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field and swimming pool. The image sizes are in the range of 800 x 800 to 4000 x 4000. In this study, airplane detection is aimed, therefore we just select the 1631 images which contain mostly commercial airplane objects and the number of them is 5209. We divided the images as the size of 1024 x 1024 patches to train Faster R-CNN and 608 x 608 for training SSD and YOLO-v3 detectors. The spatial resolution of the images varies in range 0.11 to 2 meter and they contain various orientation, aspect ratios and pixel size of the objects. Also, the images vary according to the altitude, nadir-angles of the satellites and the illumination conditions. The selected images also split as 90 percent of object samples to use at training and the rest of them for testing.

As can be seen in (Figure 5.1), DOTA training and test sets also include different samples in terms of airplane dimensions, background complexity, and illuminance conditions. Some image patches have some cropped objects, and some examples are black and white panchromatic images. Thanks to all these variations in the Dota dataset, the trained object detection architectures will be able to achieve similar performance in different image conditions.

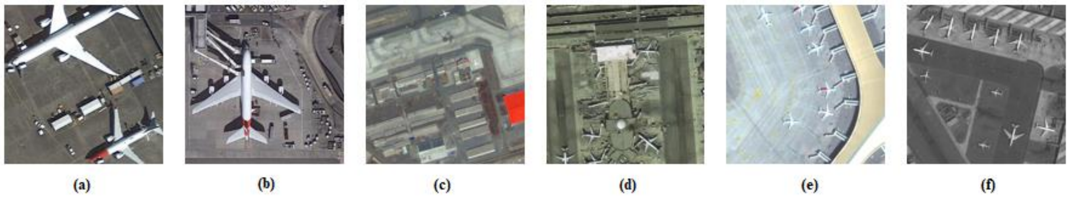


Figure 5.1 : Patches from DOTA test set; (a) cropped, (b) very big, (c) very small, (d) complex background, (e) illuminance effect, (f) panchromatic samples.

We used also 5 large scale images for testing our proposed detection flow. They are collected from Plèiades 1A and Plèiades 1B satellites which are the pan-sharpened images that have 0.5 meter spatial resolution. The images contain Istanbul Ataturk, Istanbul Sabiha Gokcen, Izmir Adnan Menderes, Ankara Esenboga and Antalya airport districts. They cover about 53 km² areas and contain 280 commercial airplanes. They also differ by the environmental conditions, altitude, and nadir-angles of the satellites.

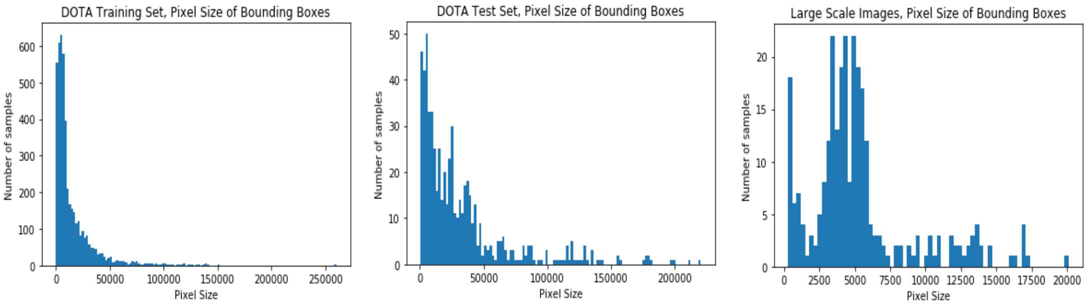


Figure 5.2 : Number of the pixel size of airplane bounding boxes in the dataset.

As shown in (Figure 5.2), the bounding box area distributions of aircraft samples are between 0 and 15000 pixels for the DOTA train set. Although this shows almost the same distribution as the Dota test set, it differs slightly from the examples in the large scale data set that we created. There is no object sample over 20000 pixels in the large scale test set and the areas of the samples are mostly in between 3000 and 6000 pixels.

5.2 Training of the Object Detection Networks

In this study, all the training process and experiments are done with Tensorflow and Keras open source deep learning libraries, which were developed by the Google research team [57]. The hardware used for these purposes is the Nvidia Geforce GTX

1080 graphic card. For each architecture, we used different training configurations according to the network and our hardware specifications. We also used the transfer learning technique by their pre-trained parameters learned from COCO dataset.

With the transfer learning approach, we start the training with the pre-trained parameters to utilize useful information gathered from the previously trained network with different data used for another problem. Although COCO dataset does contain natural images, the pre-trained model of the networks, which utilized from COCO, can be used in our problem as well, because we can learn some basic features directly from them such as edge, corner, shape, and color which form the basis of all of the vision tasks. After starting our network with the parameters of the pre-trained model, we fed it with our training examples of DOTA which we prepared before.

We used 1024 x 1024 image patches to train Faster R-CNN. For the RPN stage of it, the bounding box scales are defined as 0.25, 0.5, 1.0 and 2.0 with 0.5, 1.0 and 2.0 aspect ratios which means that the network generates 12 anchor boxes for each lacation of the feature maps. Batch size was defined as 2 to prevent the memory allocation errors. For the first attempt, the training process is continued 400k iterations and it tooks 3 days. The learning rate was started as 0.003 and was continued by reducing to half of it for each further 75k steps. The training loss did not decrease anymore and we start new training with the learning rate tenth of the old one and perform the process for 900k iterations by reducing quarter of the learning rate for each 50k steps after 150k-th iterations.

For the SSD network, with the size of 608 x 608 image patches used for training. The sizes and aspect ratios of the default bounding boxes of each feature map layers remained as same as in the original SSD article [19]. We used RMSProp optimization method for gradient calculations with 0.001 learning rate, 0.9 decay factor for each 25k iterations. The batch size was defined as 16 and the training process is done for 200k step that tooks 2 and half day. With the first attempt of the SSD, we could not get good results, too. Therefore we started a new training process with 0.0004 learning rate value and the same decay factor for each 50k-th iterations along with 450k iterations.

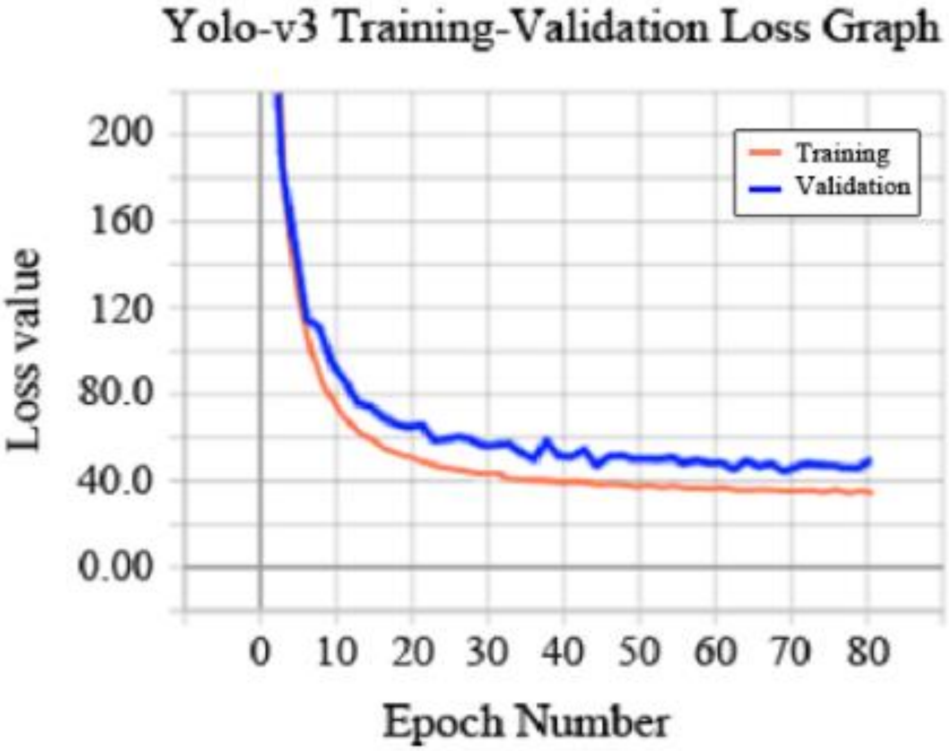
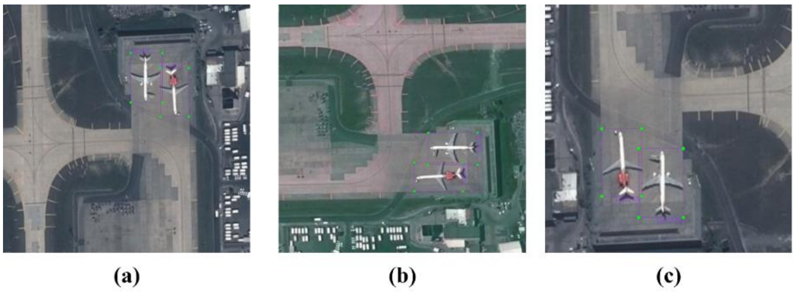


Figure 5.3 : Yolo v3 training and validation loss graphic.

The Yolo-v3 architecture was trained with Adam optimizer by learning rate $5 x 10^{-5}$ with decay factor 0.1 for every 3 epochs which the validation loss does not decrease. We used 9 anchor boxes with different size, 3 for each stage of the network as in the original paper. Before the training, we clustered the bounding boxes of our whole data according to their sizes with the k-means clustering algorithm to find 9 optimum anchor box sizes. Then we sorted them from small to large and we produced each three of them from the first stage to third. For the validation purpose, 10 percent of the training data was splitted for monitoring the validation loss during the training process. The batch size was defined as 8 and the whole training was continued for 80 epochs. One epoch means the feed forward and back propagation processes are done for the whole training dataset for one time. Training of the Yolo-v3 tooks about one and a half day.

For the training of all networks, we applied horizontal and vertical flip and cropping as augmentation techniques randomly. Besides, we scaled the image patches in HSV (hue-saturation-value) to imitate atmospheric and lightning conditions as shown in (Figure 5.4).



(a) (b) (c)

Figure 5.4 : Illustration of data augmentations; (a) Original image patch, (b) Rotated and scaled in HSV, (c) Rotated, cropped and scaled in HSV.

We observed that the Faster R-CNN and SSD can not converge well the training dataset in a short time period. At the first attempts, the training losses of them have tendencies to decrease and the results could be better with the smaller learning rates. Additionally, more variety in the aircraft samples could affect the generalization of them in a short time period. Therefore, we also softened the data augmentation process (e.g squeeze the HSV scale in a low range).

5.3 Evaluation Metrics

5.3.1 Intersection over union (IOU)

The ground-truth labels for the object detection tasks mean the bounding boxes which were drawn by people or a machine learning system that specify the exact and true location and categories of objects in an image. Intersection over union metric is calculated by finding the ratio of the overlapping area between the union area of the ground truth bounding boxes and predicted boxes.

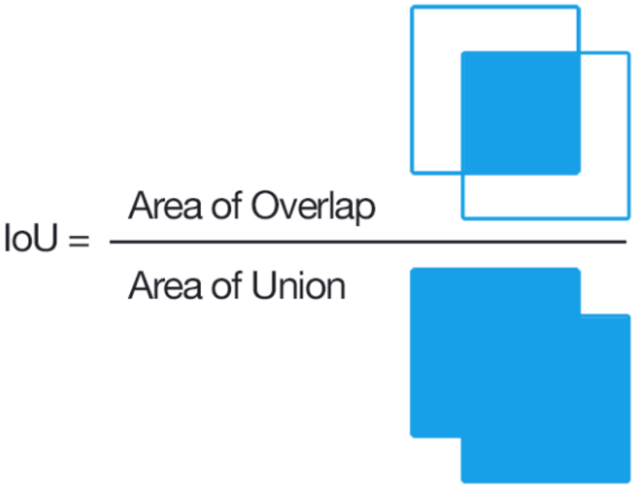


Figure 5.5 : Intersection over Union.

5.3.2 Performance metrics

In the object detection tasks, there is two widely used performance metric as mean average precision (mAP) and F1 score. At the training process, a detector compares the predicted bounding boxes with ground truth bounding boxes according to IOU at each iteration to update its parameters. Generally, it is aimed to have 0.5 iou ratio for each prediction of an object at the training stage. This means if the network predicts an object with a bounding box which its %50 area is overlapping with ground truth box, it is considered as a true prediction. But when the localization is a matter for a computer vision task, this ratio could be set higher. In our training process, we left it as 0.5, and we expected to detect the objects at least with this ratio at the evaluation phase. Therefore, this ratio is used for calculating the performance metrics.

Mean average precision is computed by the overall precision value of all predictions for recall value over 0 to 1. It is the calculation of the average precision value over all predictions. The precision means the fraction of the actual matches of all objects we detected as matches and the recall is the ratio of a number of objects that we can detect correctly to a number of all ground truth samples. So, the only recall rate or precision rate is not enough to measure the performance of our framework and the harmonic mean of them, which is F1 score, helps us in this way. Suppose true positive (TP) as truly detected objects, false negative (FN) as non-detected objects and false positive (FP) as falsely detected objects:

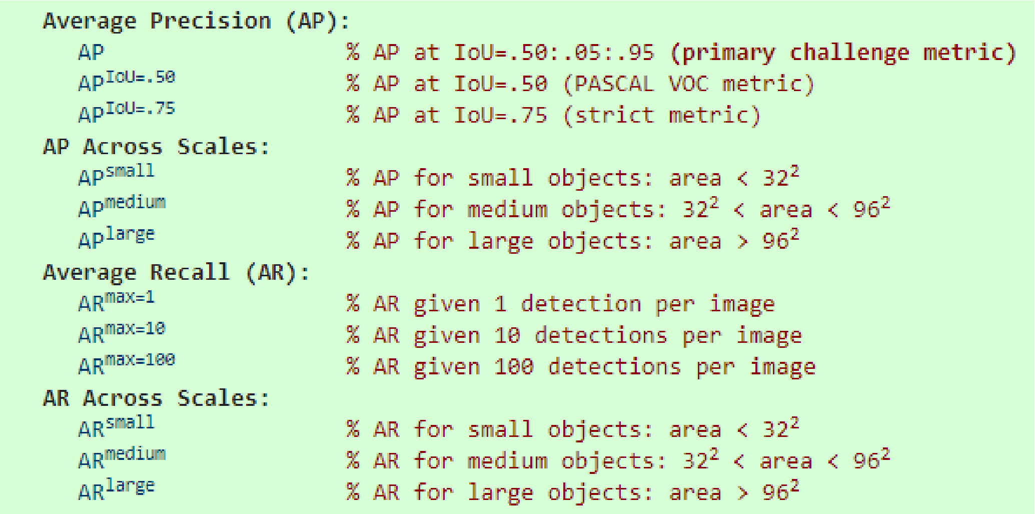
$$Precision = \frac{TP}{TP+FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.2}$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.3}$$

5.4 Results

With the COCO metric API, 12 different metrics are calculated to measure the characteristics and performance of object detection algorithms. Unless otherwise defined, average precision and average recall are calculated by averaging over 10 different intersection over union values (from 0.5 to 0.95 at 0.05 intervals). In addition, the values where iou is 0.5 and 0.75 are calculated for AP. AP is the average precision calculation according to all categories and iou values (In our problem, there is 1 category as aircraft). AR is the maximum number of detections per image, averaged over categories and IoUs. These calculations are also examined by looking at the bounding box areas. According to COCO, objects with a size of fewer than 32² pixels are defined as small, between 32² and 96² as medium and larger than 96² pixels. When performing performance metrics, calculations are made according to all three dimensions and for all dimensions separately. When calculating the metrics, we defined the aircraft scales as in the COCO [56].



Average Precision (AP):

AP % AP at IoU=.50:.05:.95 (primary challenge metric)
$AP^{IoU=.50}$ % AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$ % AP at IoU=.75 (strict metric)

AP Across Scales:

AP^{small} % AP for small objects: area < 32^2
AP^{medium} % AP for medium objects: 32^2 < area < 96^2
AP^{large} % AP for large objects: area > 96^2

Average Recall (AR):

$AR^{max=1}$ % AR given 1 detection per image
$AR^{max=10}$ % AR given 10 detections per image
$AR^{max=100}$ % AR given 100 detections per image

AR Across Scales:

AR^{small} % AR for small objects: area < 32^2
AR^{medium} % AR for medium objects: 32^2 < area < 96^2
AR^{large} % AR for large objects: area > 96^2

Figure 5.6 : 12 performance metrics of COCO challenge.

We named these metrics as in the (Table 5.1).

Table 5.1 : Performance metrics with calculation rules.

Calculated For	Metric Name
AP for [IoU=0.50:0.95 | area=all | maxDets=100]	1. Metric
AP for [IoU=0.50 | area=all | maxDets=100]	2. Metric
AP for [IoU=0.75 | area=all | maxDets=100]	3. Metric
AP for [IoU=0.50:0.95 | area=small | maxDets=100]	4. Metric
AP for [IoU=0.50:0.95 | area=medium | maxDets=100]	5. Metric
AP for [IoU=0.50:0.95 | area=large | maxDets=100]	6. Metric
AR for [IoU=0.50:0.95 | area=all | maxDets= 1]	7. Metric
AR for [IoU=0.50:0.95 | area=all | maxDets= 10]	8. Metric
AR for [IoU=0.50:0.95 | area=all | maxDets=100]	9. Metric
AR for [IoU=0.50:0.95 | area=small | maxDets=100]	10. Metric
AR for [IoU=0.50:0.95 | area=medium | maxDets=100]	11. Metric
AR for [IoU=0.50:0.95 | area=large | maxDets=100]	12. Metric

The Dota dataset was randomly divided into two as test and training. However, there is a difference in the distribution according to medium and large object scales. As we can see in (Table 5.2), for the large scale test dataset that we created, most of the objects are in the medium range.

Table 5.2 : The ratios(%) of data sets according to object scale.

	Small	Medium	Large
Dota Training Set	0.06	0.52	0.42
Dota Test Set	0.03	0.28	0.69
Large Scale Image Set	0.1	0.76	0.14

In order to see how well the models converge the training data, we also calculated the performance metrics for the DOTA training set in addition to the test data. As it is understood from the metrics, Faster R-CNN gave the best results when looking at the mean of precision for different iou values. Yolo-v3 is good for 0.5 iou and above, while Faster R-CNN is better if 0.75 iou and above is desired. When we look at Metric 4,5 and 6, Faster R-CNN gives the best AP result for different iou in small, medium and large objects for the DOTA test set. However, in the large scale image set, we can see that Yolo-v3 are better for small and medium objects. The reason for this is that the architectures have different structures to learn different attributes from training data. The 7,8 and 9 metrics give us information about the recall rates for all object sizes according to the detection number per image. Here again, the Faster R-CNN looks better. When we look at the AR results according to metrics 10, 11 and 12, we can see that the recall rates of Yolo-v3 is worse than the SSD for large scale image set. In addition to this, the SSD is also ahead of the Faster R-CNN for small and medium aircrafts.

The performances of all trained models were examined with COCO metric API, except for the first training attempts of SSD and Faster R-CNN. Because they gave bad results. The fact that DOTA training and test performances are close to each for the three architectures shows that the models can actually learn the object examples in DOTA well. However, when compared to the large scale image set that we prepared, there is a big performance gap. The main reasons for this may be that the dimensions of the aircraft are distributed different than DOTA and they contain different types of aircraft.

Table 5.3 : Performance of DOTA training, validation and large scale image set according to COCO metrics.

	Dota Training Set			Dota Test Set			Large Scale Image Set		
	Yolo-v3	2. SSD	2. Faster R-CNN	Yolo-v3	2. SSD	2. Faster R-CNN	Yolo-v3	2. SSD	2. Faster R-CNN
Metric 1	0,42	0,41	0,48	0,39	0,37	0,45	0,14	0,15	0,17
Metric 2	0,80	0,71	0,75	0,76	0,61	0,71	0,43	0,43	0,36
Metric 3	0,39	0,44	0,57	0,34	0,41	0,51	0,07	0,07	0,13
Metric 4	0,04	0,02	0,11	0,07	0,04	0,08	0,05	0,02	0,04
Metric 5	0,41	0,39	0,46	0,35	0,29	0,39	0,17	0,18	0,16
Metric 6	0,49	0,47	0,54	0,41	0,42	0,48	0,06	0,14	0,3
Metric 7	0,18	0,18	0,21	0,25	0,27	0,30	0	0,00	0
Metric 8	0,48	0,45	0,51	0,45	0,43	0,50	0,06	0,05	0,07
Metric 9	0,5	0,47	0,53	0,45	0,43	0,50	0,24	0,27	0,26
Metric 10	0,06	0,06	0,16	0,07	0,08	0,13	0,05	0,07	0,07
Metric 11	0,48	0,46	0,52	0,40	0,35	0,42	0,27	0,29	0,26
Metric 12	0,56	0,53	0,59	0,48	0,47	0,53	0,19	0,30	0,40

With the COCO metric API, precision-recall curves can be plotted according to the object size and differences between the curves can give some insights about the models in terms of the capabilities for aircraft detection problem.

5.4.1 SSD evaluation results

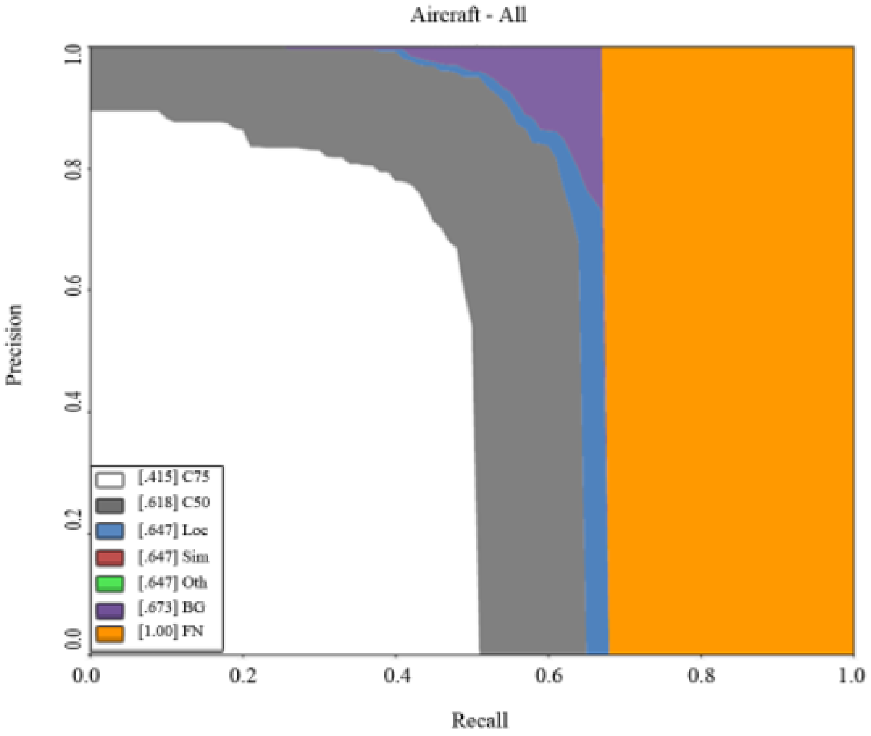


Figure 5.7 : SSD precision-recall curve of DOTA test set for all objects size.

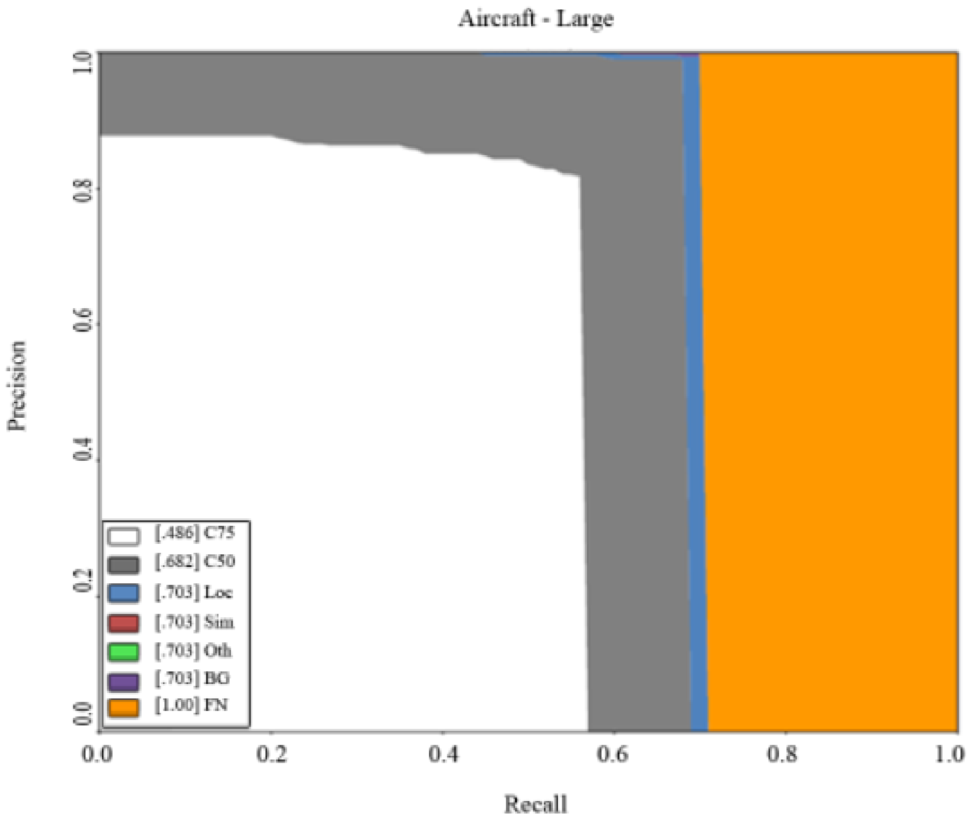


Figure 5.8 : SSD precision-recall curve of DOTA test set for large objects size.

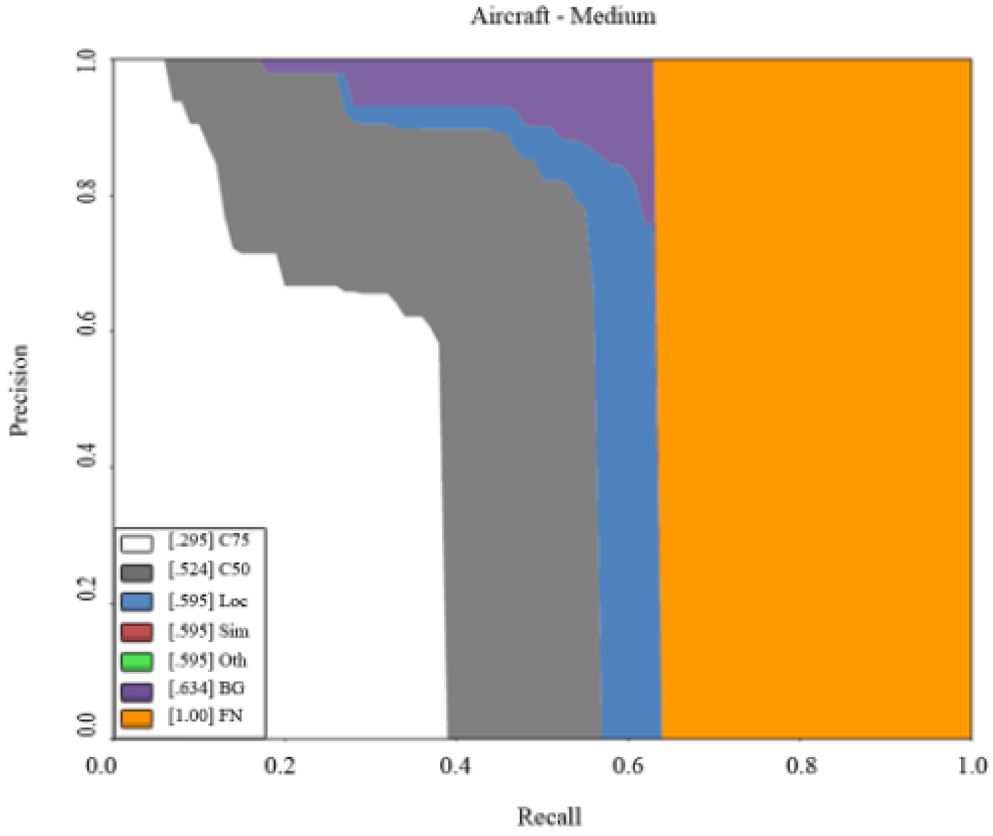


Figure 5.9 : SSD precision-recall curve of DOTA test set for medium objects size.

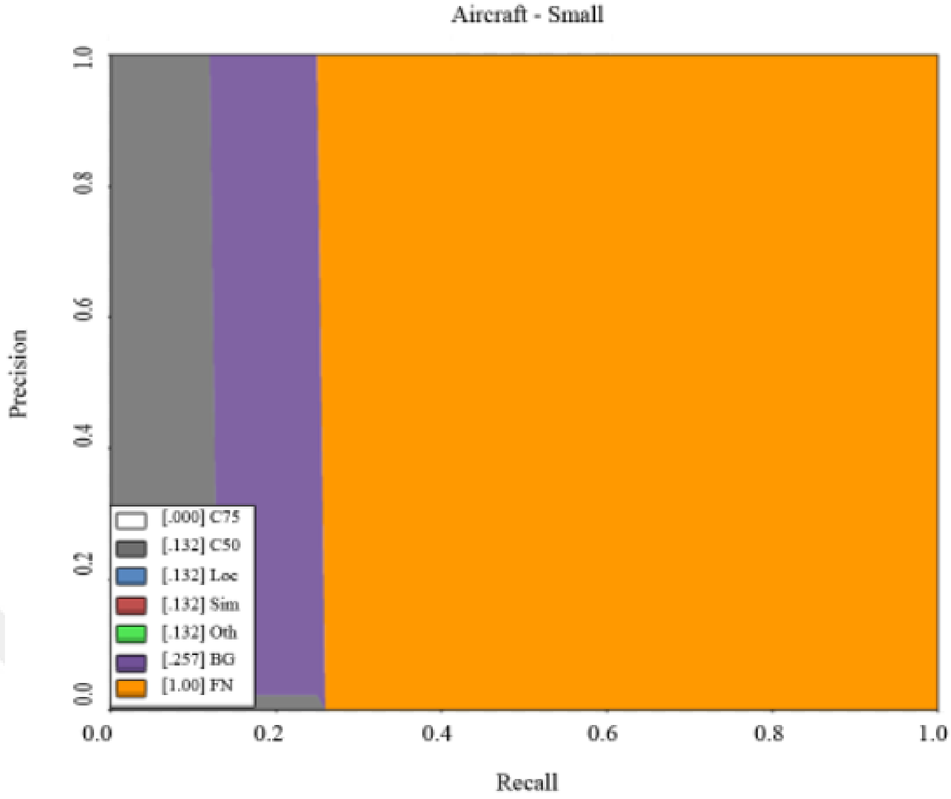


Figure 5.10 : SSD precision-recall curve of DOTA test set for small objects size.

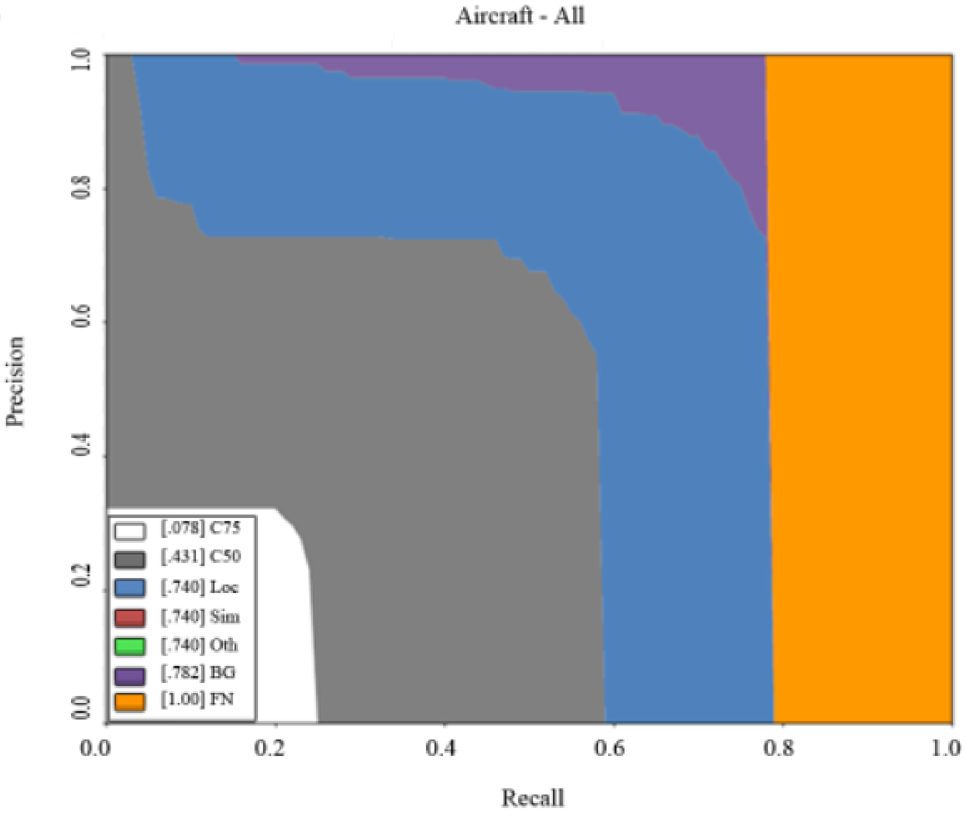


Figure 5.11 : SSD precision-recall curve of large scale test set for all objects size.

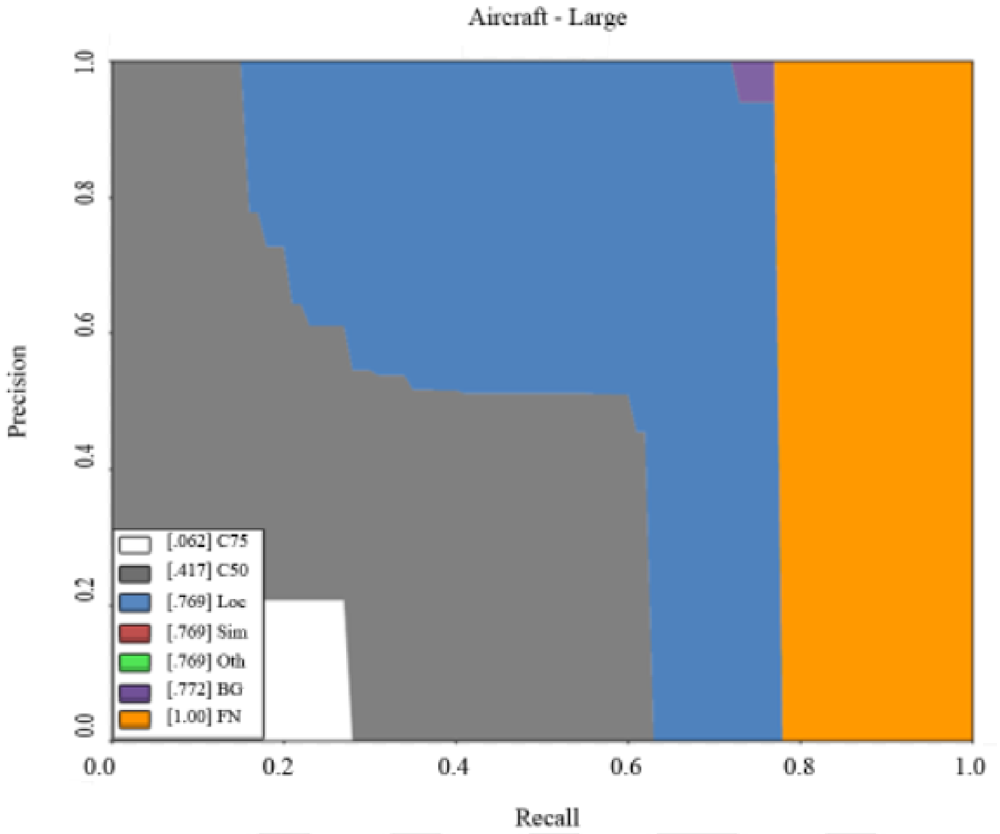


Figure 5.12 : SSD precision-recall curve of large scale test set for large objects size.

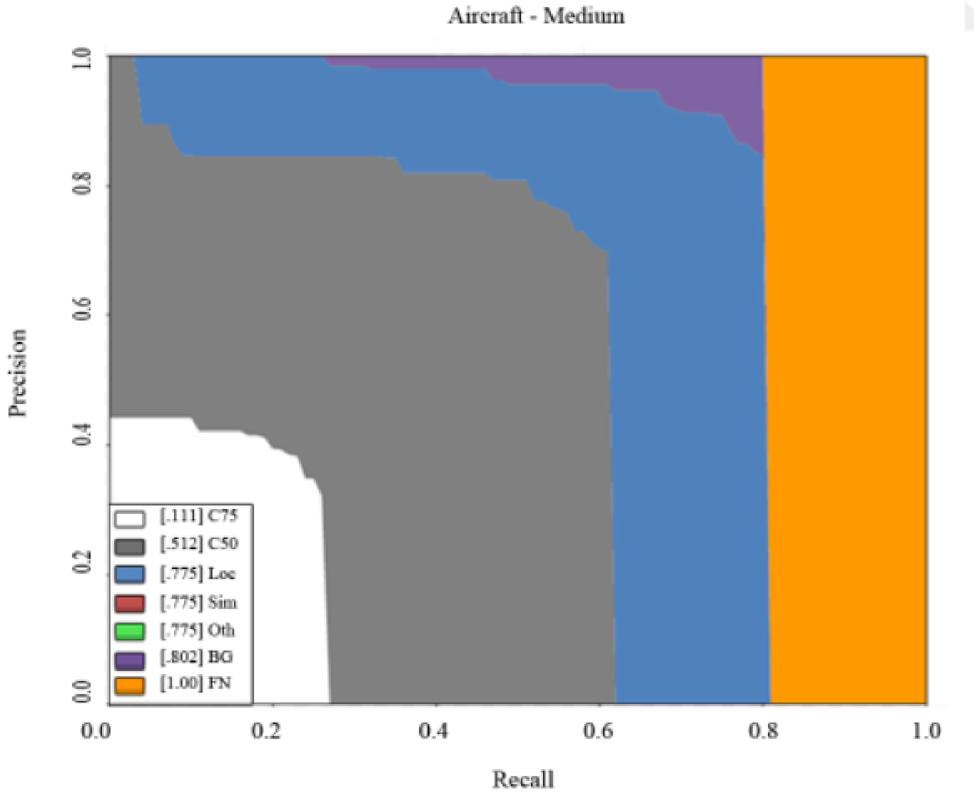


Figure 5.13 : SSD precision-recall curve of large scale test set for medium objects size.

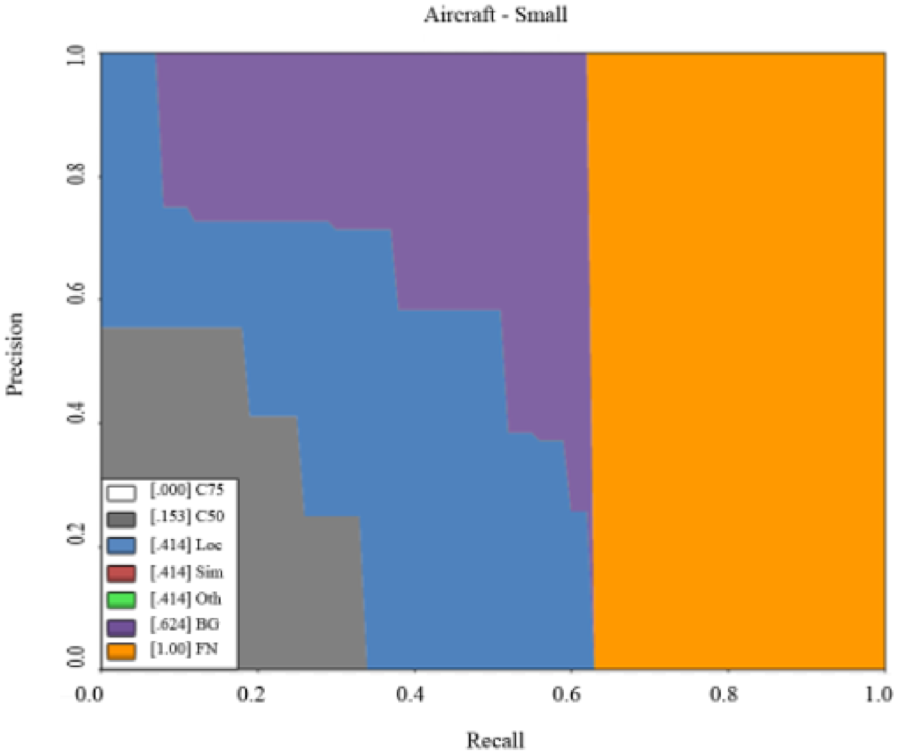


Figure 5.14 : SSD precision-recall curve of large scale test set for small objects size.

5.4.2 Faster R-CNN evaluation results

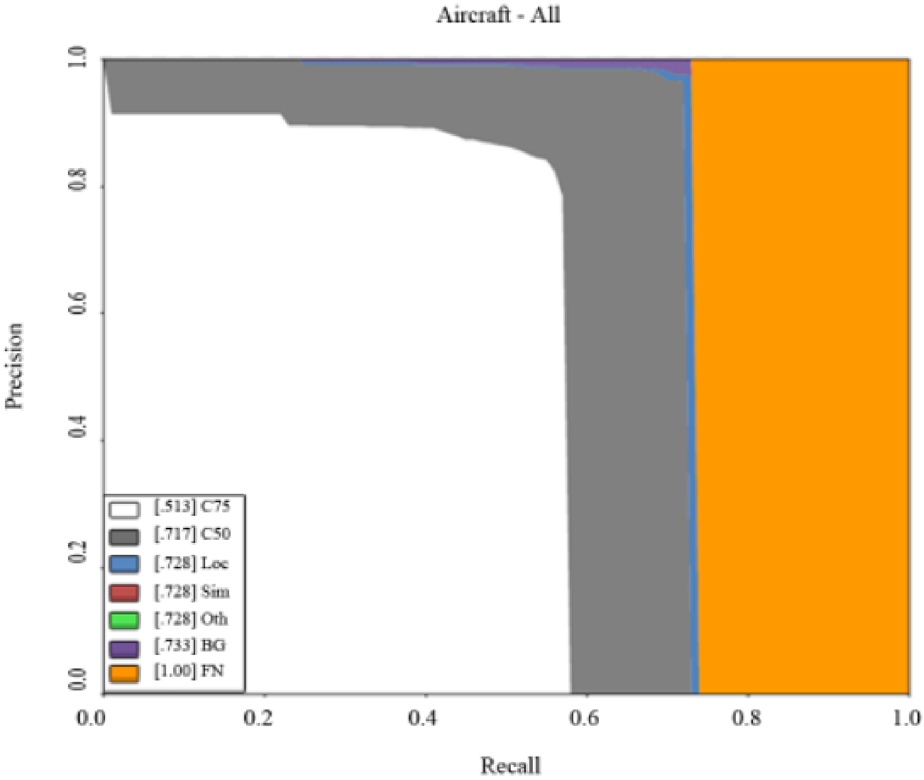


Figure 5.15 : Faster R-CNN precision-recall curve of DOTA test set for all objects size.

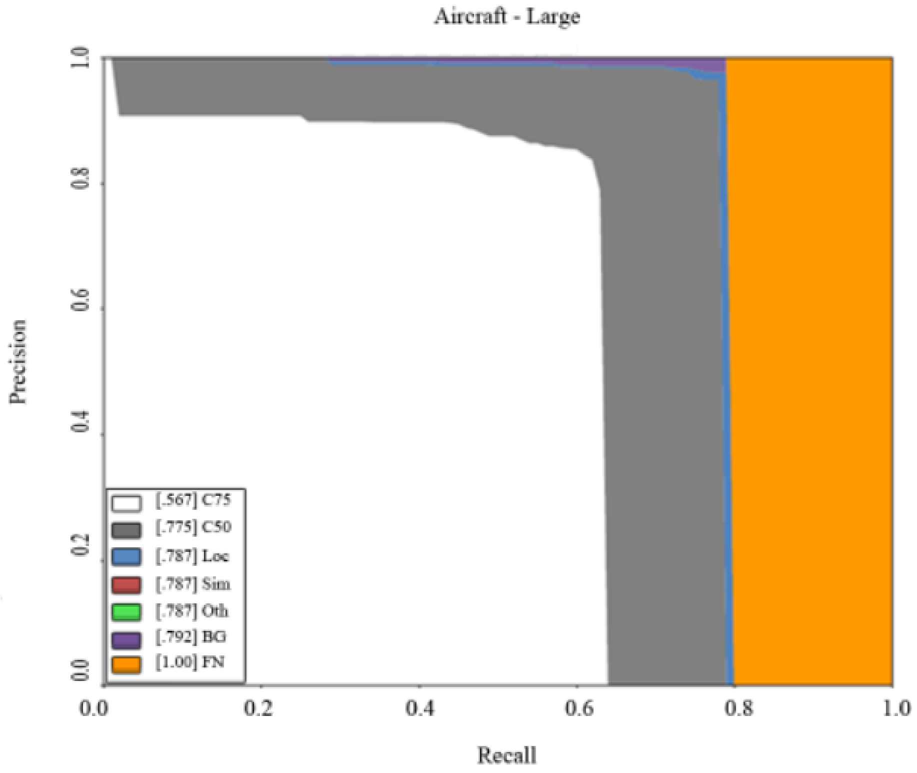


Figure 5.16 : Faster R-CNN precision-recall curve of DOTA test set for large objects size.

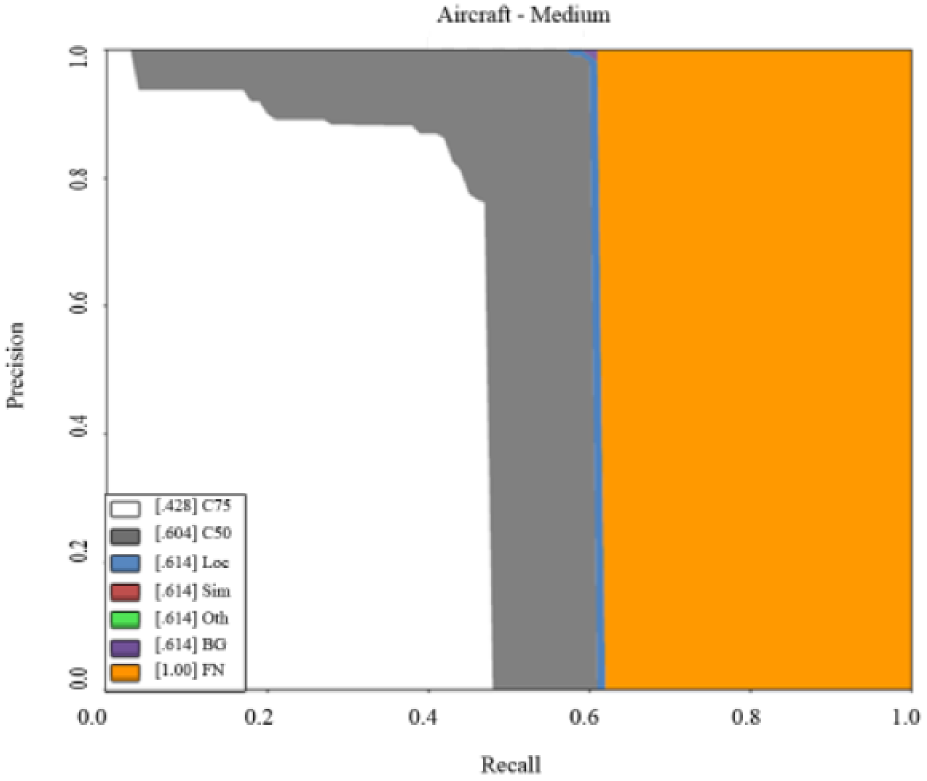


Figure 5.17 : Faster R-CNN precision-recall curve of DOTA test set for medium objects size.

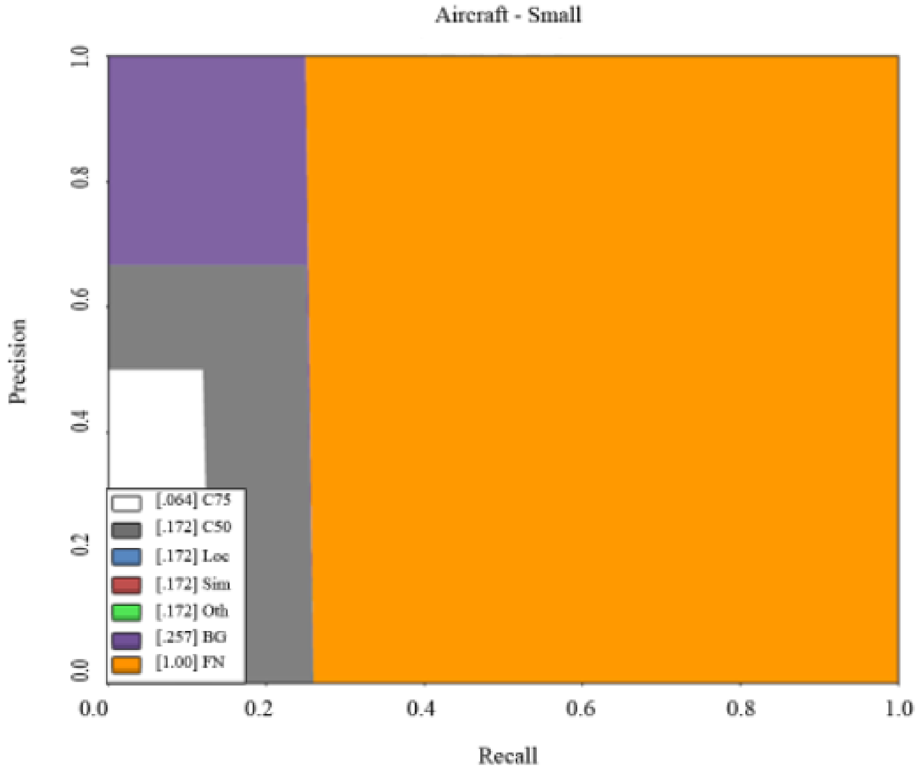


Figure 5.18 : Faster R-CNN precision-recall curve of DOTA test set for small objects size.

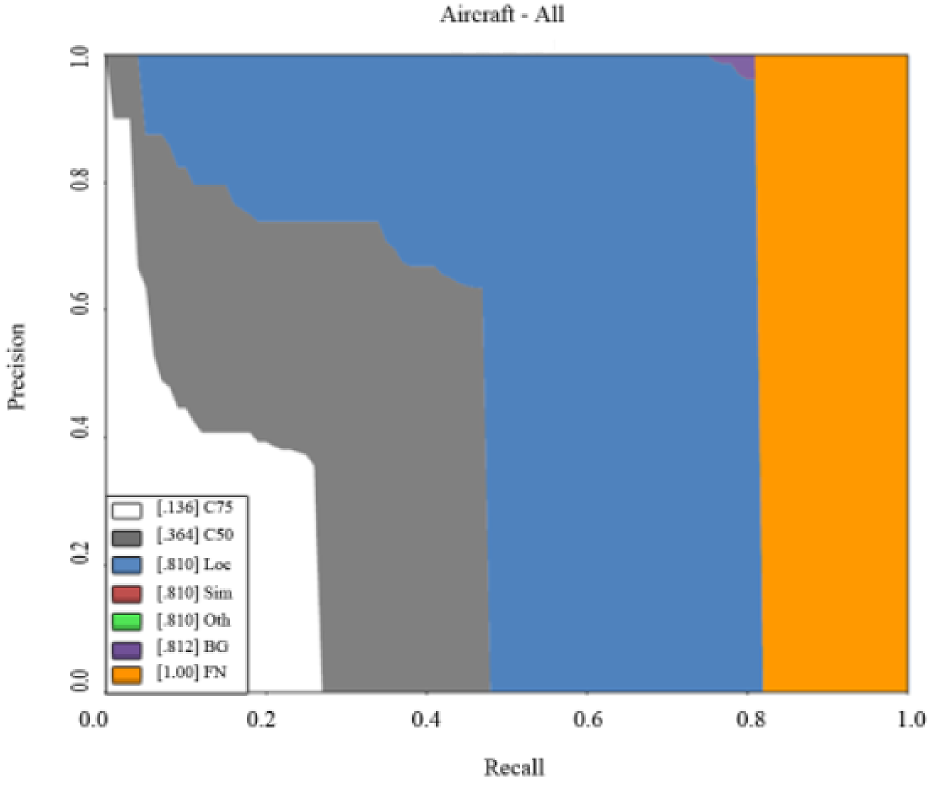


Figure 5.19 : Faster R-CNN precision-recall curve of large scale test set for all objects size.

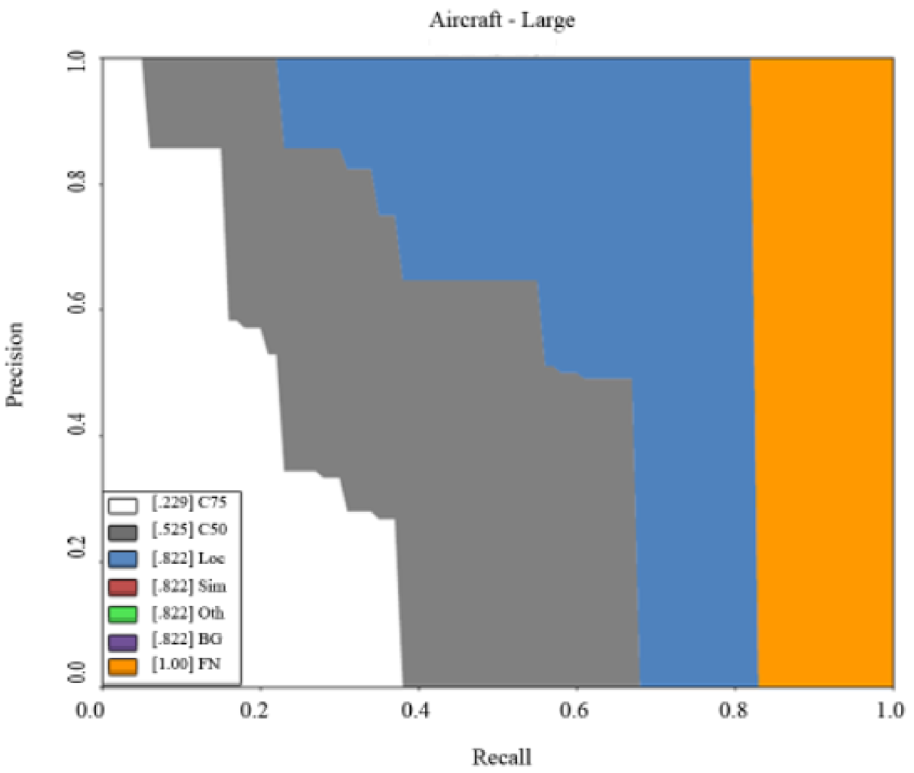


Figure 5.20 : Faster R-CNN precision-recall curve of large scale test set for large objects size.

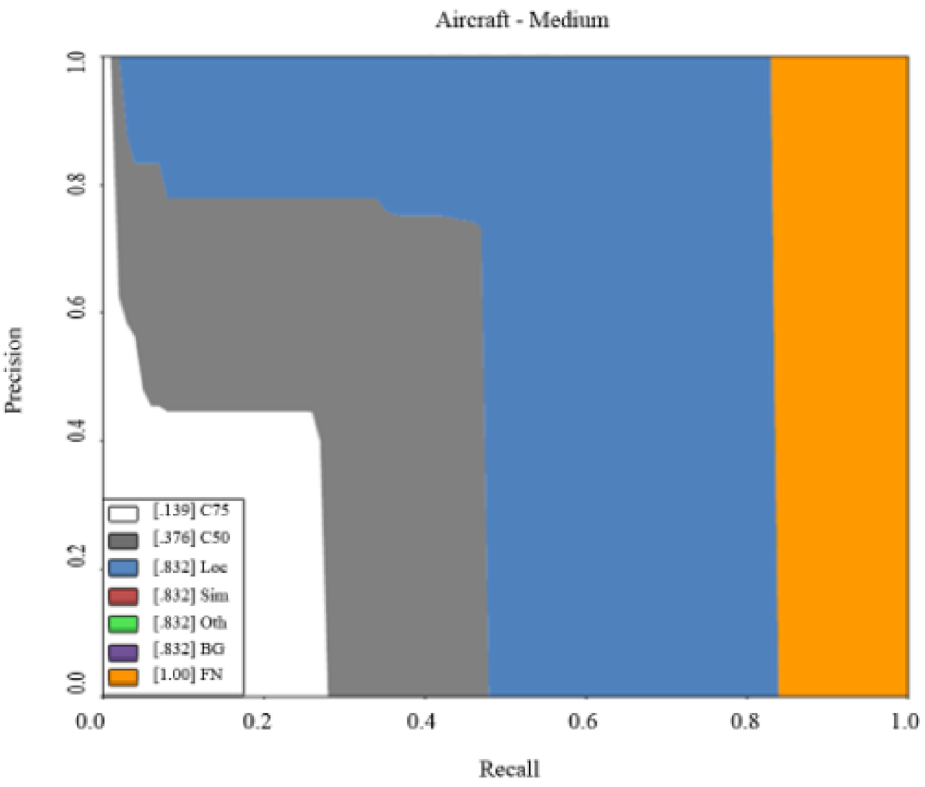


Figure 5.21 : Faster R-CNN precision-recall curve of large scale test set for medium objects size.

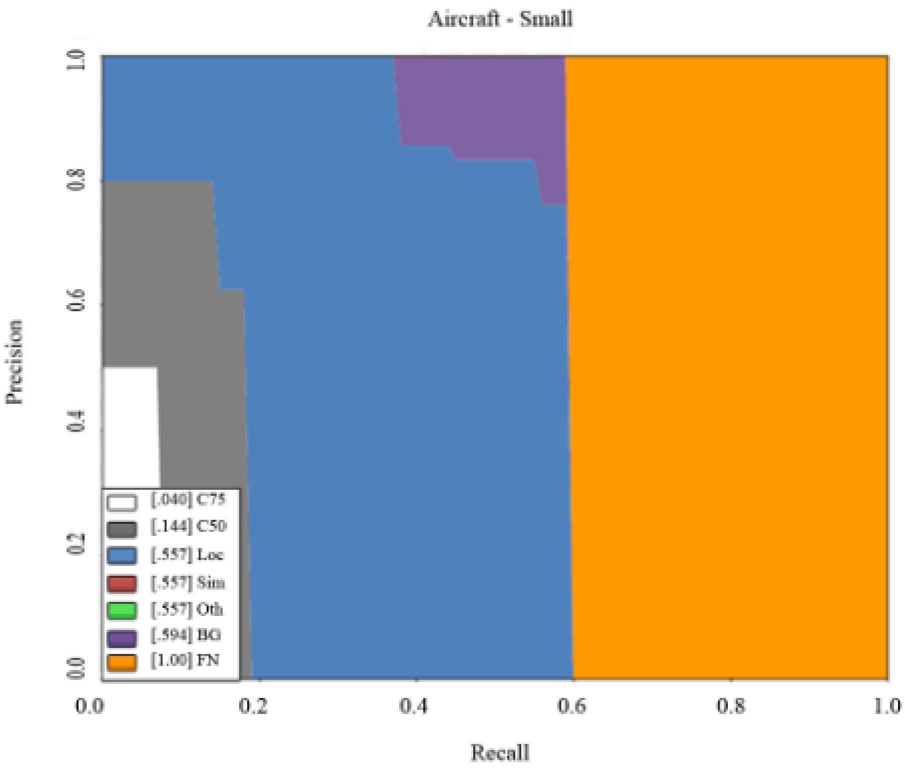


Figure 5.22 : Faster R-CNN precision-recall curve of large scale test set for small objects size.

5.4.3 Yolo-v3 evaluation results

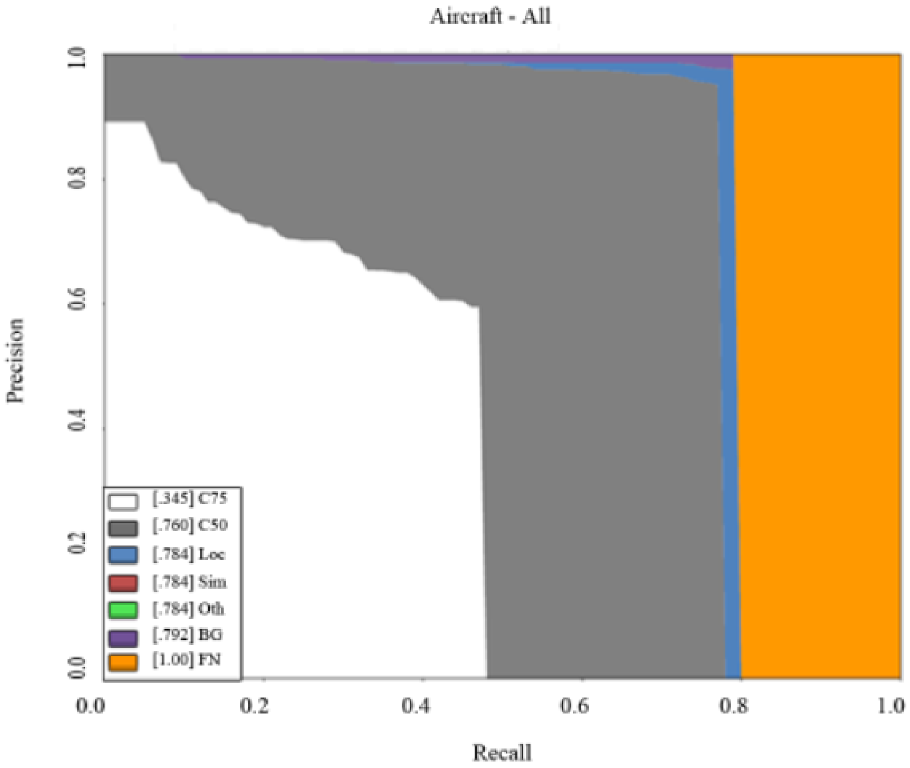


Figure 5.23 : Yolo-v3 precision-recall curve of DOTA test set for all objects size.

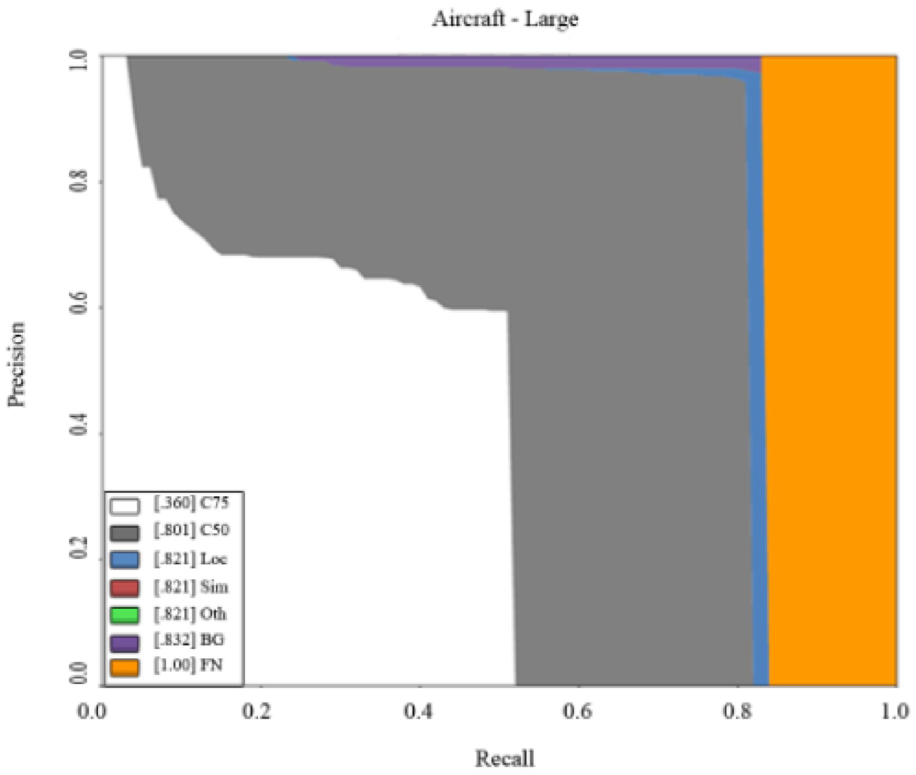


Figure 5.24 : Yolo-v3 precision-recall curve of DOTA test set for large objects size.

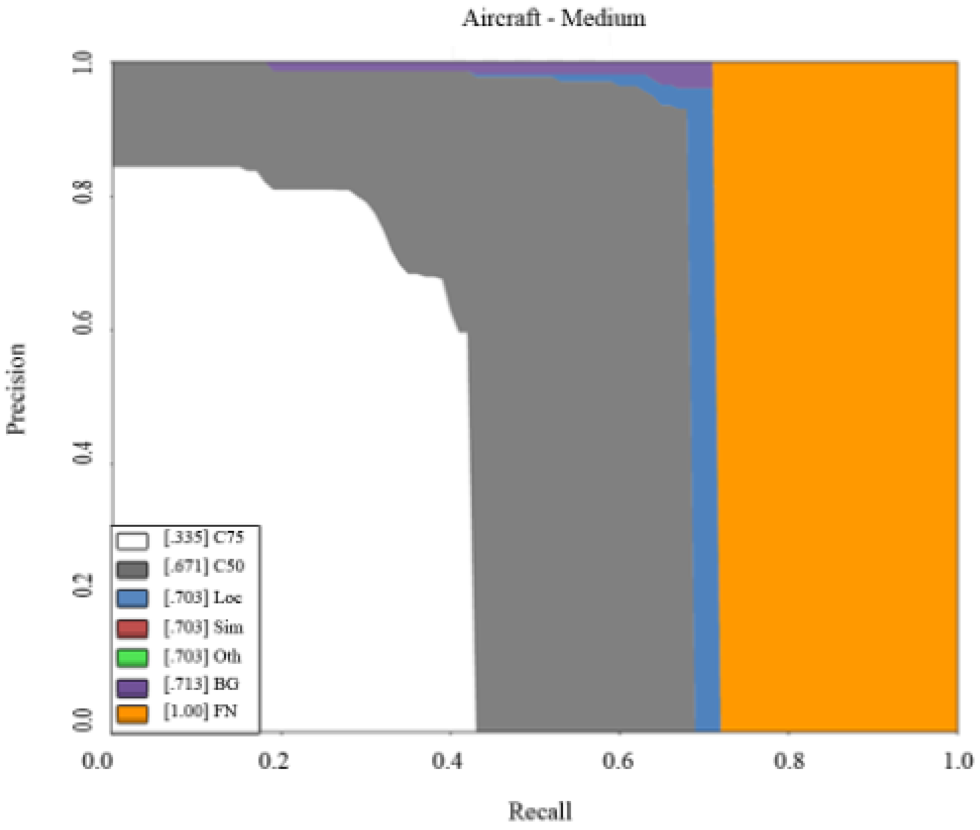


Figure 5.25 : Yolo-v3 precision-recall curve of DOTA test set for medium objects size.

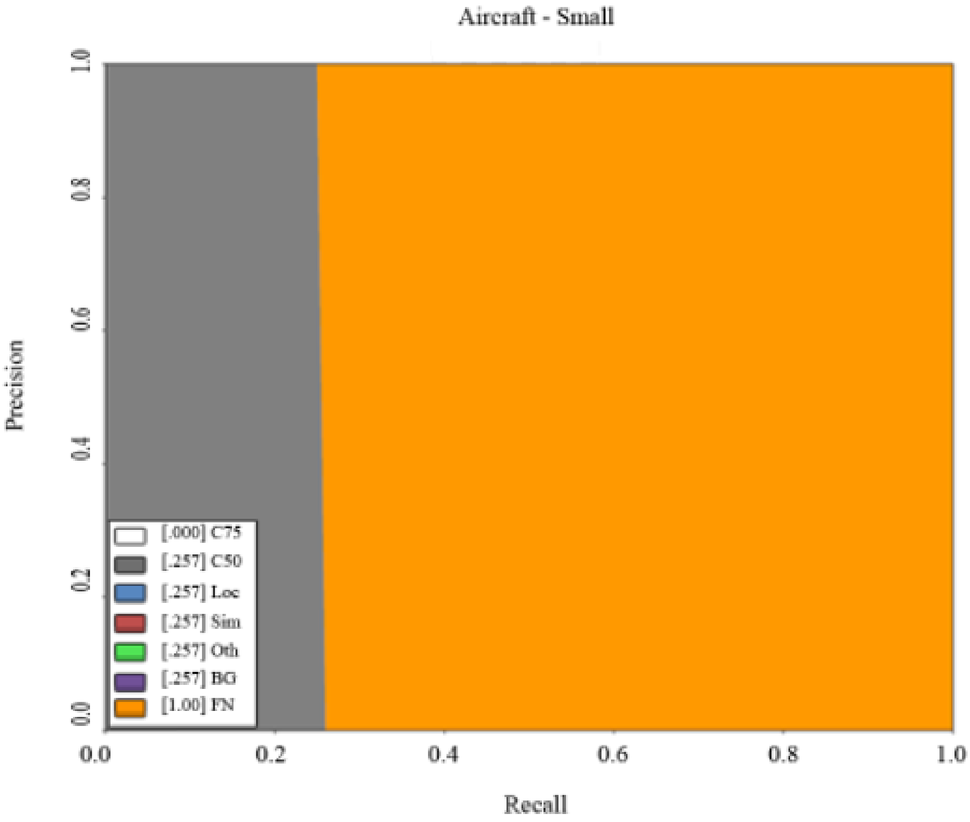


Figure 5.26 : Yolo-v3 precision-recall curve of DOTA test set for small objects size.

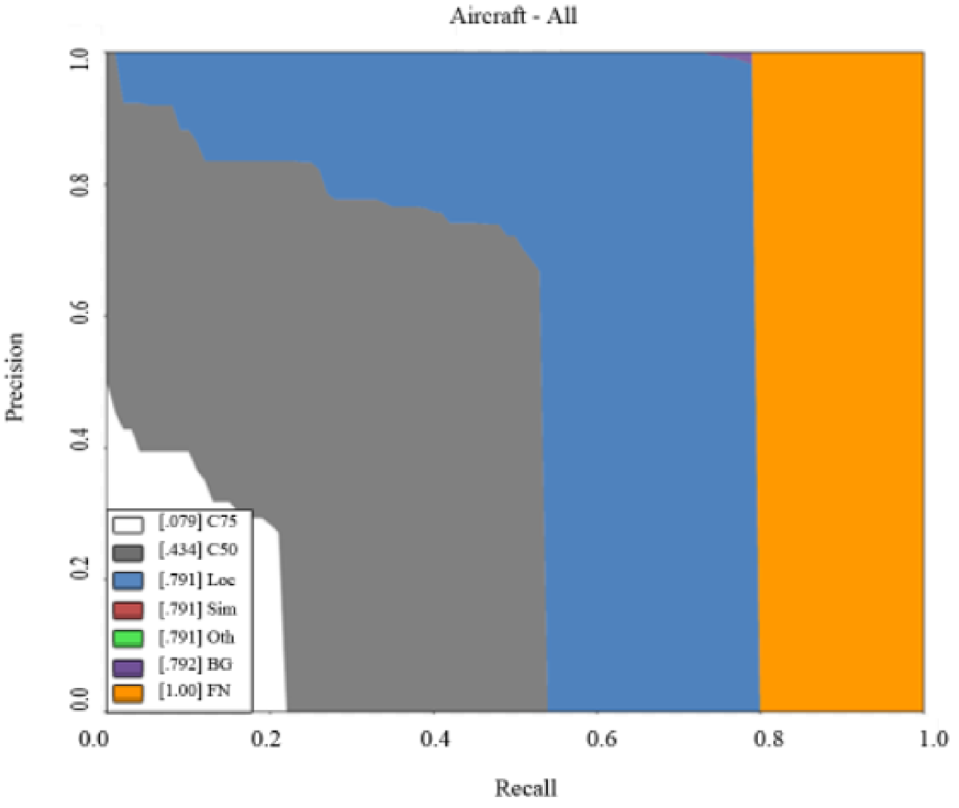


Figure 5.27 : Yolo-v3 precision-recall curve of large scale test set for all objects size.

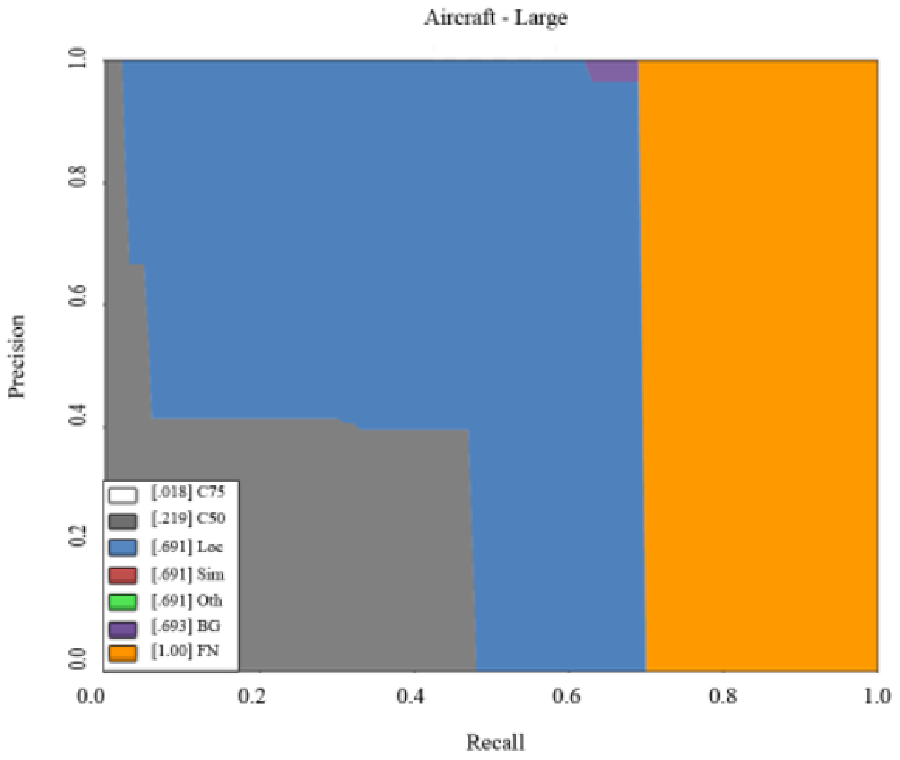


Figure 5.28 : Yolo-v3 precision-recall curve of large scale test set for large objects size.

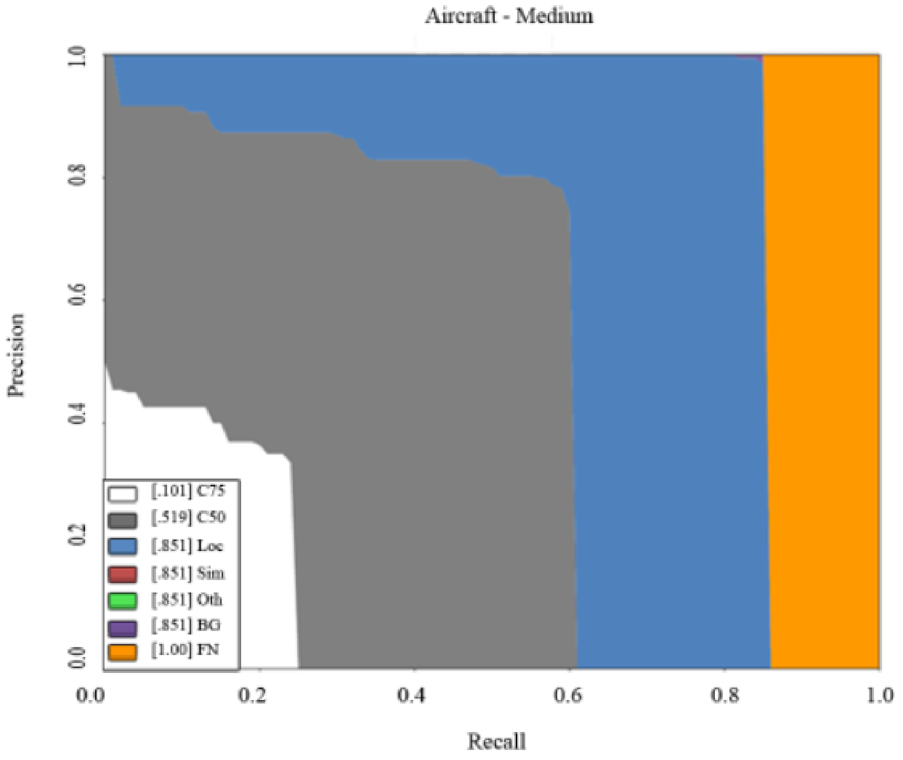


Figure 5.29 : Yolo-v3 precision-recall curve of large scale test set for medium objects size.

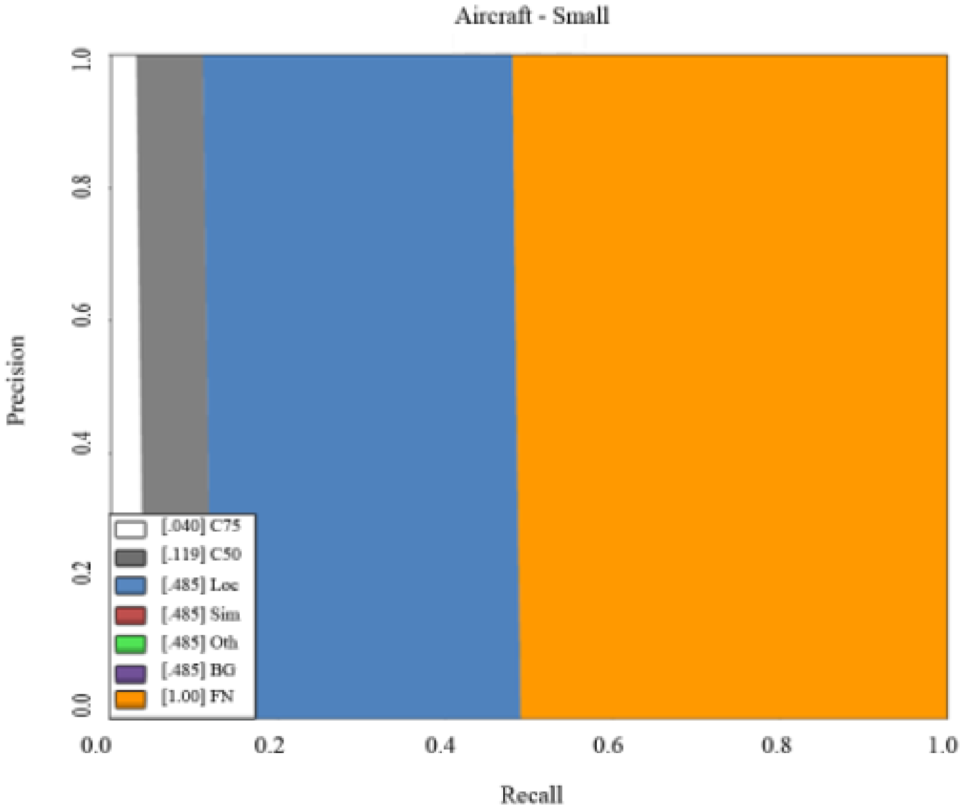


Figure 5.30 : Yolo-v3 precision-recall curve of large scale test set for small objects size.

As seen in the (Figure 5.7) to (Figure 5.30) we get the precision-recall curves according to small, medium, large scale of objects and all of them. The evaluations were done for the DOTA test set and large scale image set separately. The orange area out of the curves represents the false negative portion of the evaluated data set. In other words, it is the PR after all errors are removed. The purple area means the falsely detected objects which are the backgrounds in our work. The blue area shows the localization errors of the predicted boxes. It is indicated that the PR curve at 0.1 iou value. The white area shows the area under the precision-recall curve which comprised by the prediction with iou above 0.75. Lastly, the grey area is for the detections with the iou ratio above 0.5.

Brown area (Sim) is the PR curve after the supercategory false positives are removed. Green area (Oth) is PR after all class confusions are removed. Because of we do not have any supercategory (e.g cat and dog are two categories and animal is super category of them) or any other category, we do not have these curves in our plots.

Table 5.4 : mAP metrics of all test sets for all networks according to COCO Metric API.

	Dota Test Set				Large Scale Image Set			
mAP	C75	C50	Loc	Bg	C75	C50	Loc	Bg
Yolo-v3 All	0.34	0.76	0.78	0.79	0.08	0.43	0.79	0.79
Yolo-v3 Large	0.36	0.80	0.82	0.83	0.02	0.22	0.69	0.69
Yolo-v3 Medium	0.33	0.67	0.70	0.71	0.10	0.51	0.85	0.85
Yolo-v3 Small	0	0.25	0.25	0.25	0.04	0.11	0.48	0.48
2. SSD All	0.41	0.61	0.64	0.67	0.07	0.43	0.74	0.78
2. SSD Large	0.48	0.68	0.70	0.70	0.06	0.41	0.76	0.77
2. SSD Medium	0.29	0.52	0.59	0.63	0.11	0.51	0.77	0.80
2. SSD Small	0	0.13	0.13	0.25	0	0.15	0.41	0.62
2. Faster R-CNN All	0.51	0.71	0.72	0.73	0.13	0.36	0.81	0.81
2. Faster R-CNN Large	0.56	0.77	0.78	0.79	0.22	0.52	0.82	0.82
2. Faster R-CNN Medium	0.42	0.60	0.61	0.61	0.13	0.37	0.83	0.83
2. Faster R-CNN Small	0.06	0.17	0.17	0.25	0.04	0.14	0.55	0.59

We can see easily the large aircrafts detected better for the dota test and large scale image set from the figures. Also, it can bee seen that the networks detect well with the iou above of 0.5 when we compare the margin area between iou with 0.75. Although, the localization error for the DOTA test set is less, it appears to be much greater in large scale images. For the Yolo-v3 network, we chosed 9 optimum anchor sizes by clustering the whole DOTA training samples according to the object sizes of them, but we can see in the large scale image set, the pixel sizes of objects are much smaller. Besides, the object number of DOTA samples in the training set is much more than the large scale image set that we prepared. This could occur the imbalance between the object sizes of two dataset. Namely, the sizes of the anchor boxes we choose with the k-means algorithm could not that optimum size for the large scale image dataset. This condition could be the explanation of the bigger localization errors of the large scale image set.

Although, SSD network obtains worse performance for the test sets, it is much sensitive to localize the objects well by comparing with the other networks. We can also see the Yolo-v3 network is better at the detection of the small objects with 0.5 iou. Additionally, Faster R-CNN can detect the small objects of DOTA test set with %6 mAP while the other networks can not and for the small objects of the large scale image set, it has similar performance with the Yolo-v3.

We also did the calculation for the best precision, recall and F1 score with the iou thresholds 0.5 for all networks and all dataset. In order to observe how the models can generalize the training data, we have calculated these scores for the training data as well.

Table 5.5 : Precision, Recall and F1 score of all datasets.

	Dota Training Set			Dota Test Set			Large Scale Image Set		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Yolo-v3	0.99	0.95	0.97	0.96	0.89	0.92	0.97	0.87	**0.91**
1. SSD	0.99	0.44	0.61	0.96	0.43	0.59	0.65	0.36	**0.46**
2. SSD	0.89	0.73	0.80	0.86	0.68	0.76	0.87	0.65	**0.74**
1. Faster R-CNN	0.99	0.51	0.67	0.99	0.47	0.63	0.74	0.41	**0.52**
2. Faster R-CNN	0.97	0.92	0.95	0.98	0.89	0.93	0.98	0.92	**0.94**

The detection processes took about 37s by SSD, 97s by Yolo v3, 102s by the Faster R-CNN with our proposed detection flow approach for all of the large scale image set which cover 53 km^2 area in total.

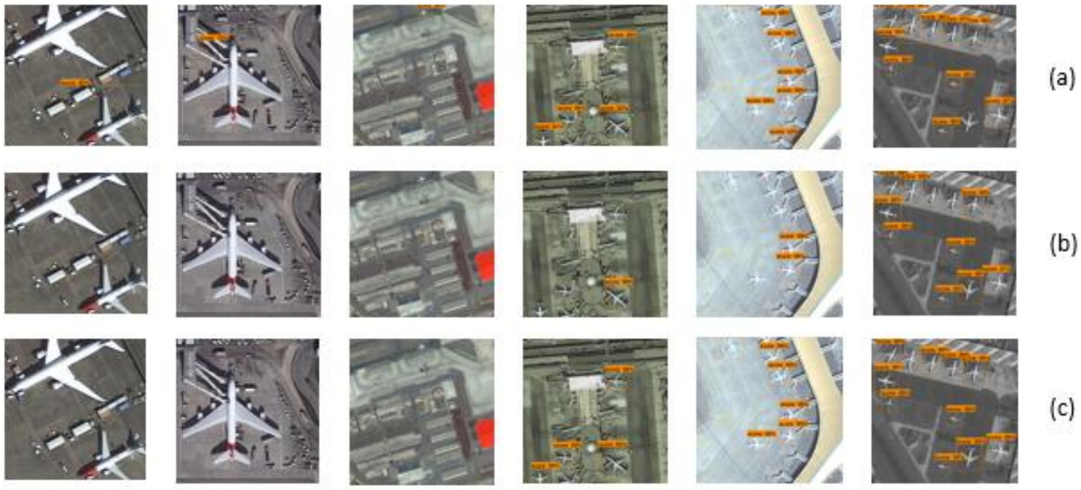


Figure 5.31 : Detection results of some of the DOTA test set patches (a) Yolo-v3, (b) SSD, (c) Faster R-CNN.

(Figure 5.31) shows the detections made on some challenging sample images in the DOTA test set. As we can see in these examples, Yolo-v3 is more successful than the others. Although, the selected samples have different aircraft sizes, illuminance effects, background complexities and different band information, the Yolo-v3 has a small amount of missing objects, while SSD shows the worst results. But with the overall test samples we can see from the tables, Faster R-CNN is best model and SSD is worst. We did the evaluation process with Yolo-v3, second trained SSD and Faster R-CNN. The precision recall curves, mAP result tables and the resulting images are extracted by them. Additionally, we included in the (Table 5.5), the first trained model of SSD and Faster R-CNN evaluation results according to precision, recall and F1 scores to see the improvements.

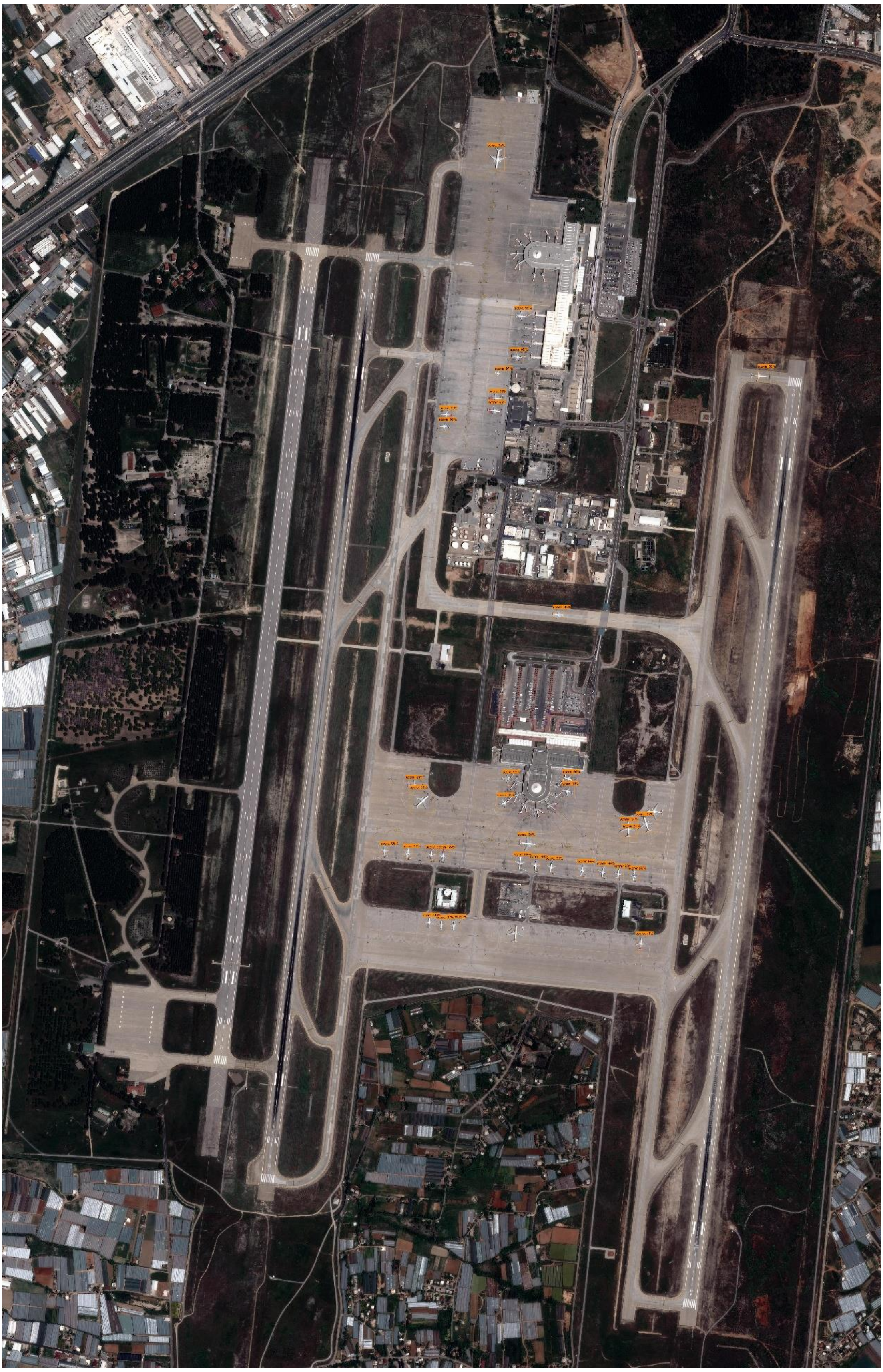


Figure 5.32 : Yolo-v3 evaluation for the Antalya Airport.

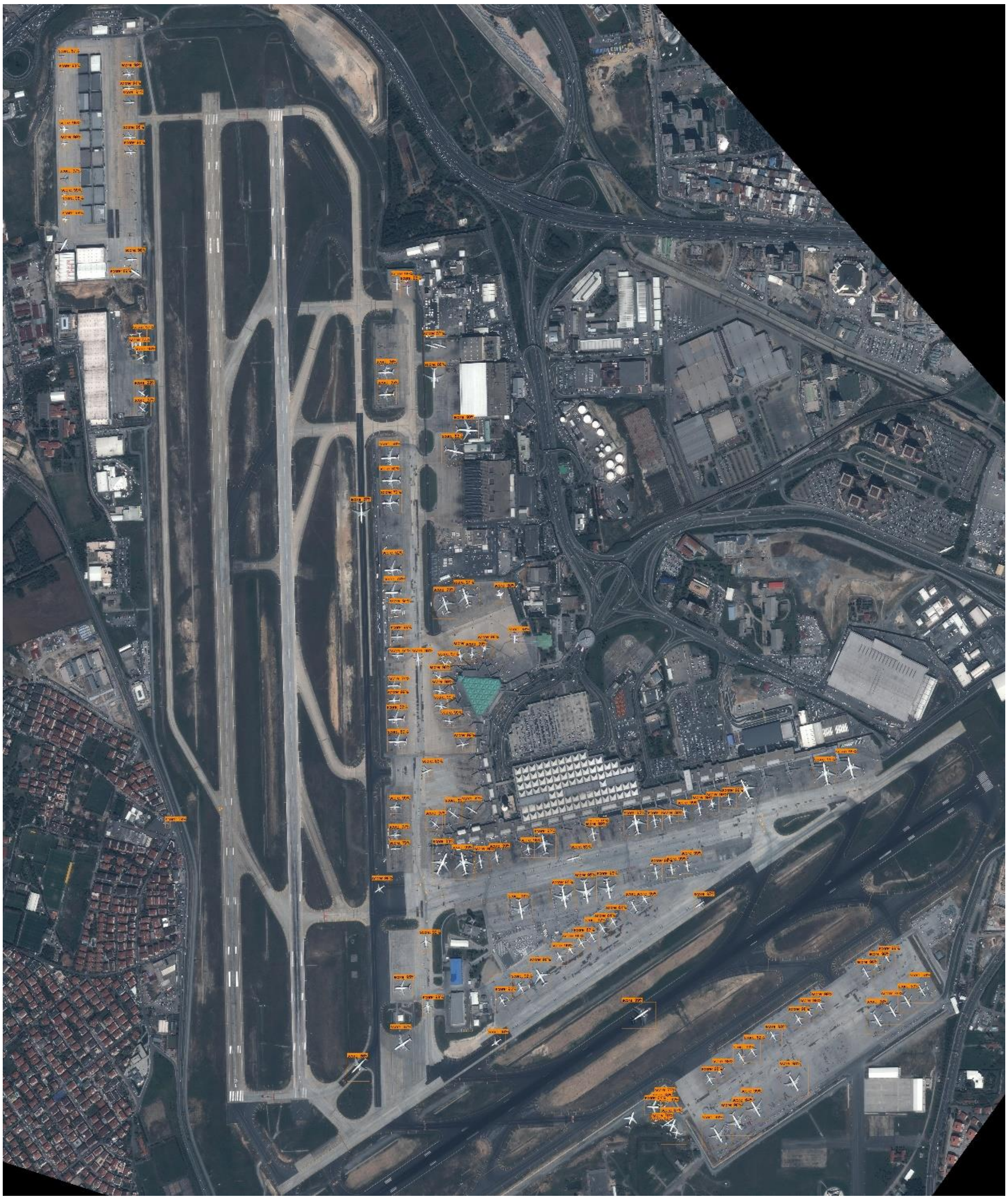


Figure 5.33 : Yolo-v3 evaluation for the Istanbul Ataturk Airport.

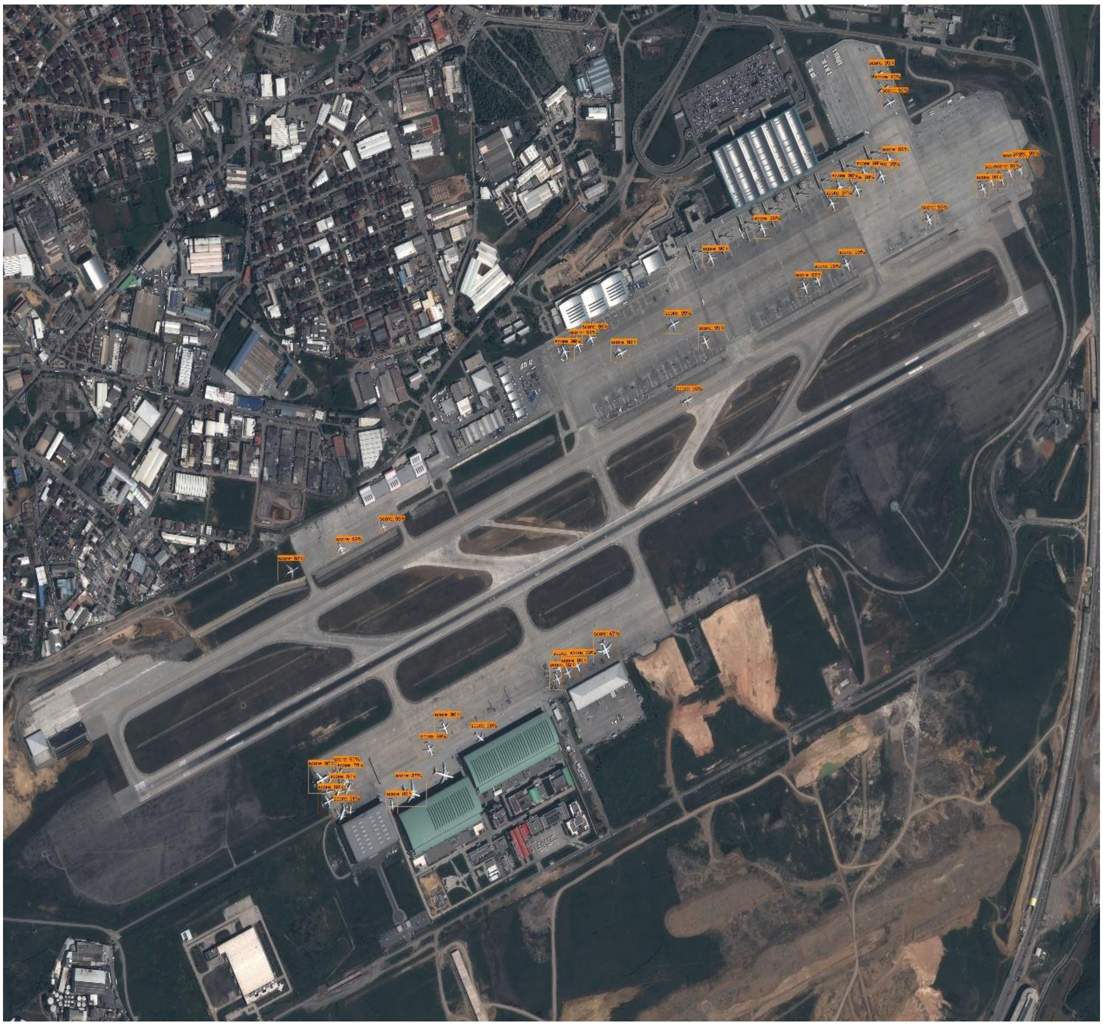


Figure 5.34 : Yolo-v3 evaluation for the Istanbul Sabiha Gökçen Airport.

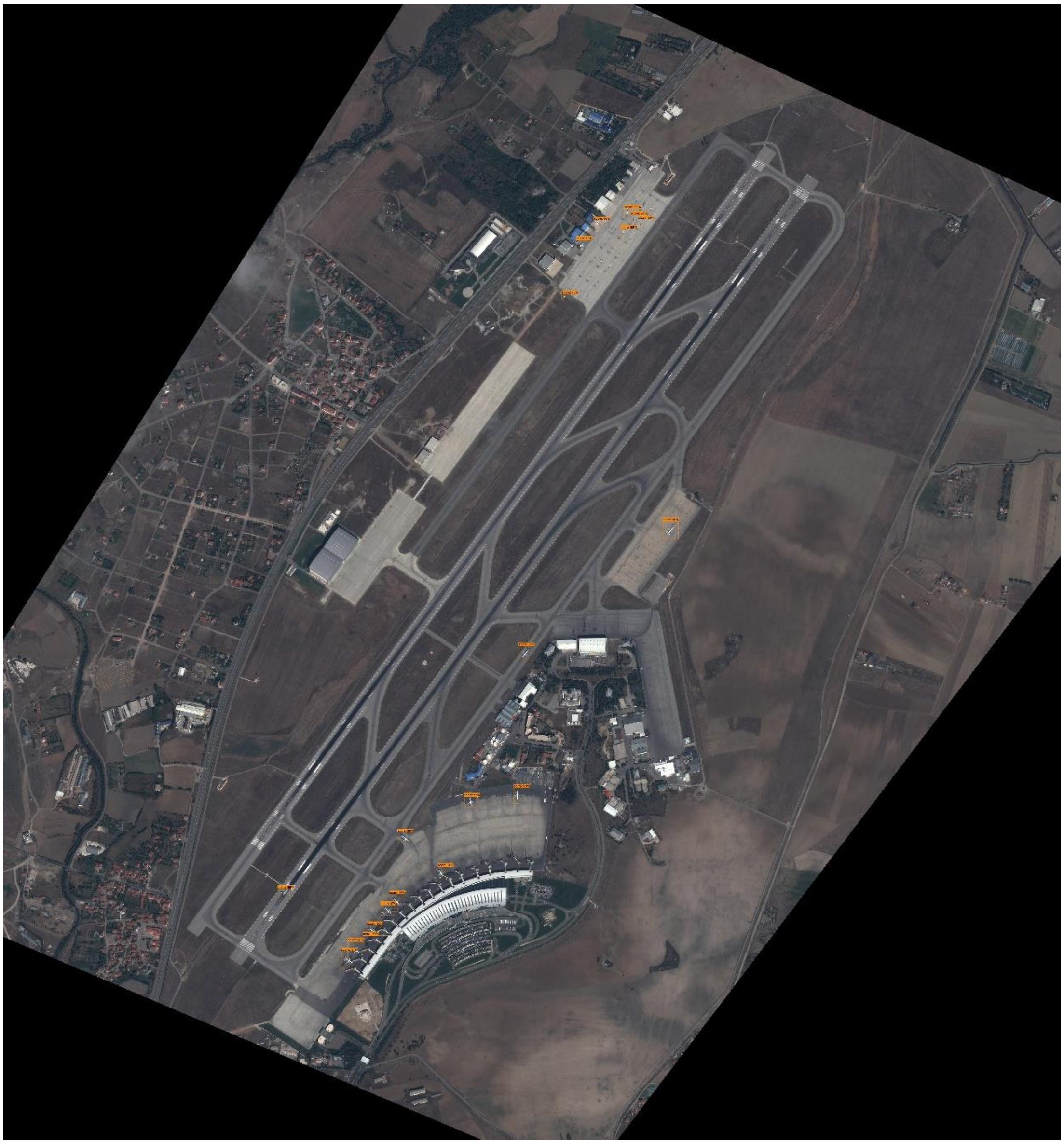


Figure 5.35 : Yolo-v3 evaluation for the Esenboga Airport.

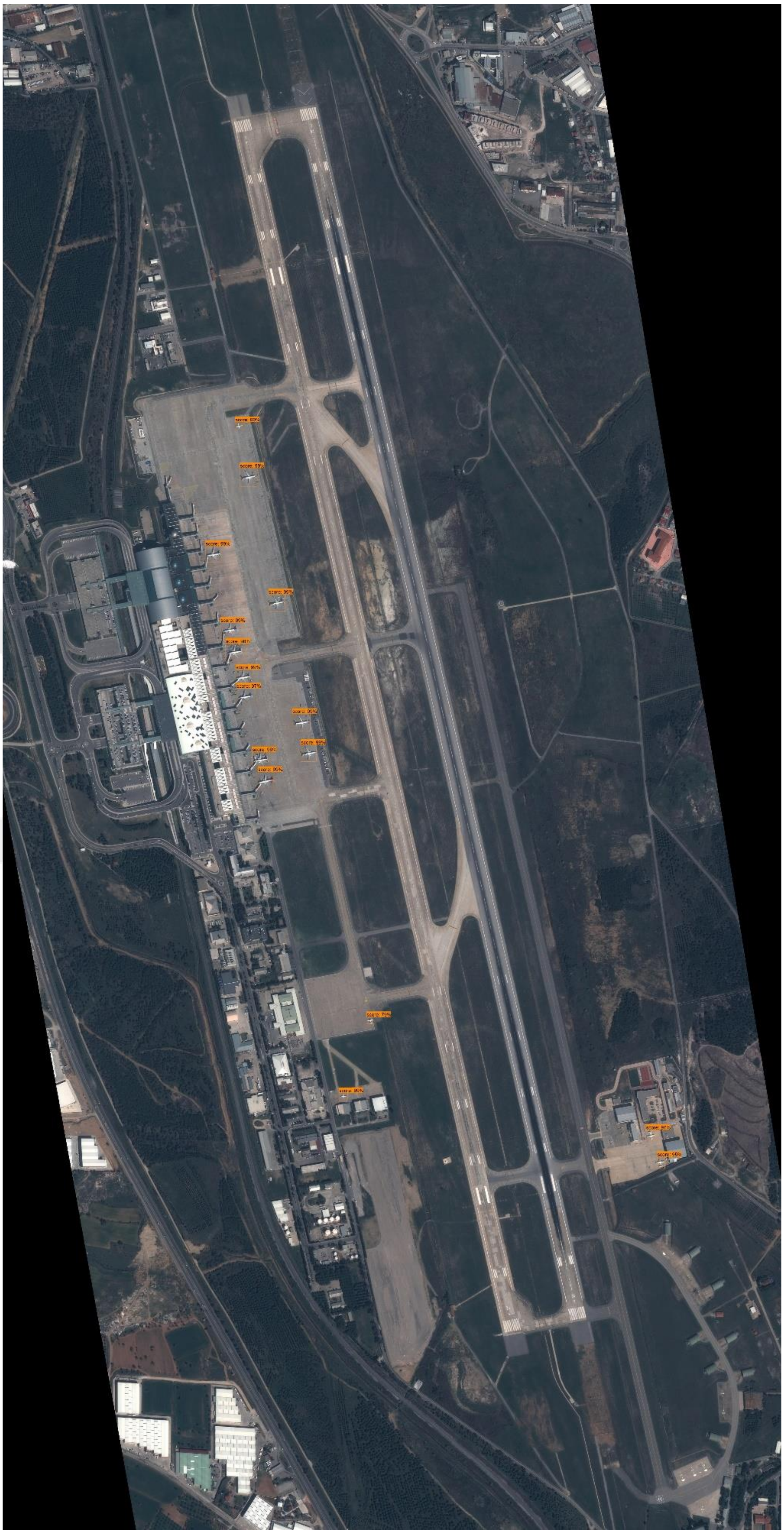


Figure 5.36 : Yolo-v3 evaluation for the Izmir Adnan Menderes Airport.

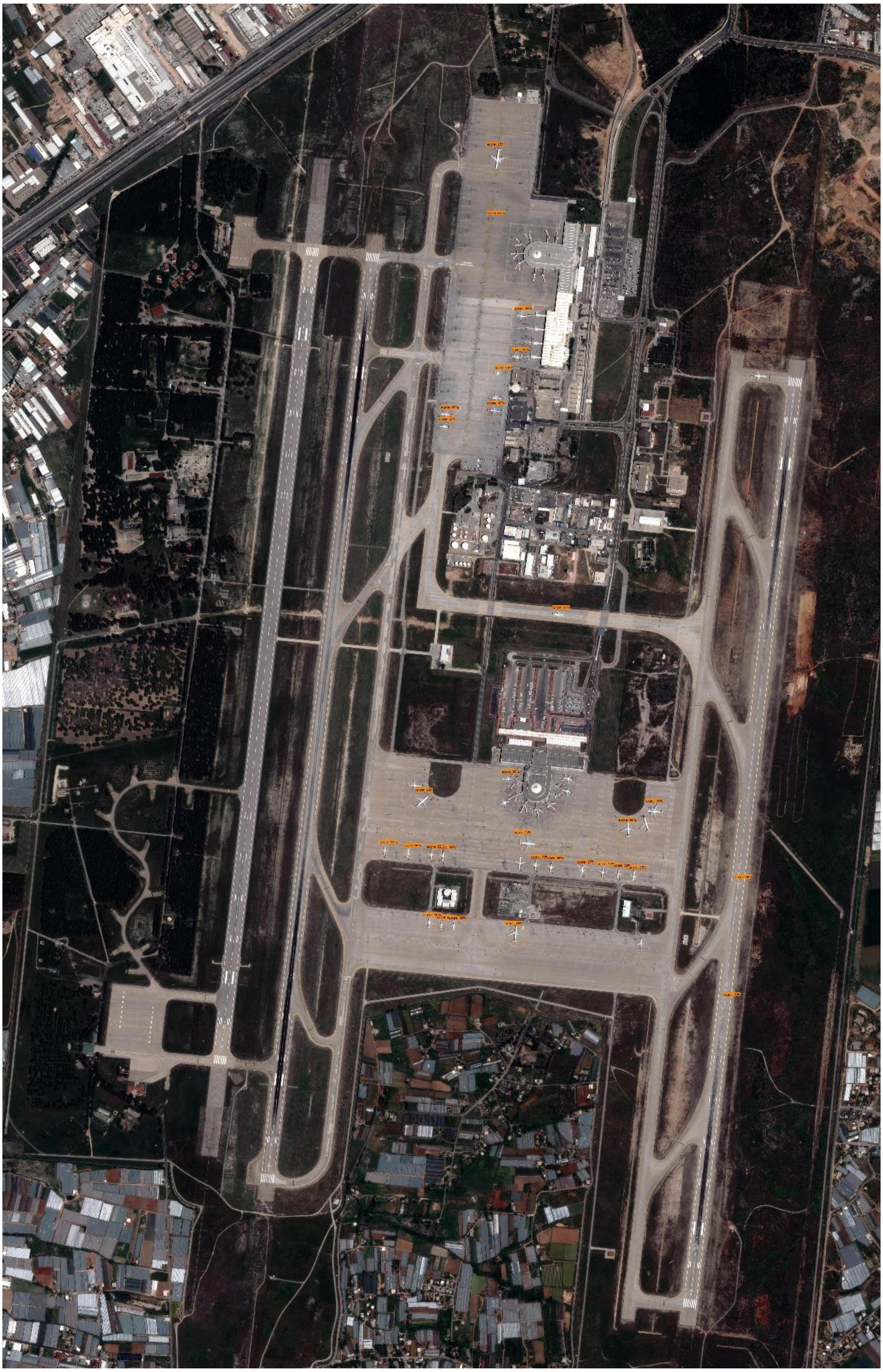


Figure 5.37 : SSD evaluation for the Antalya Airport.

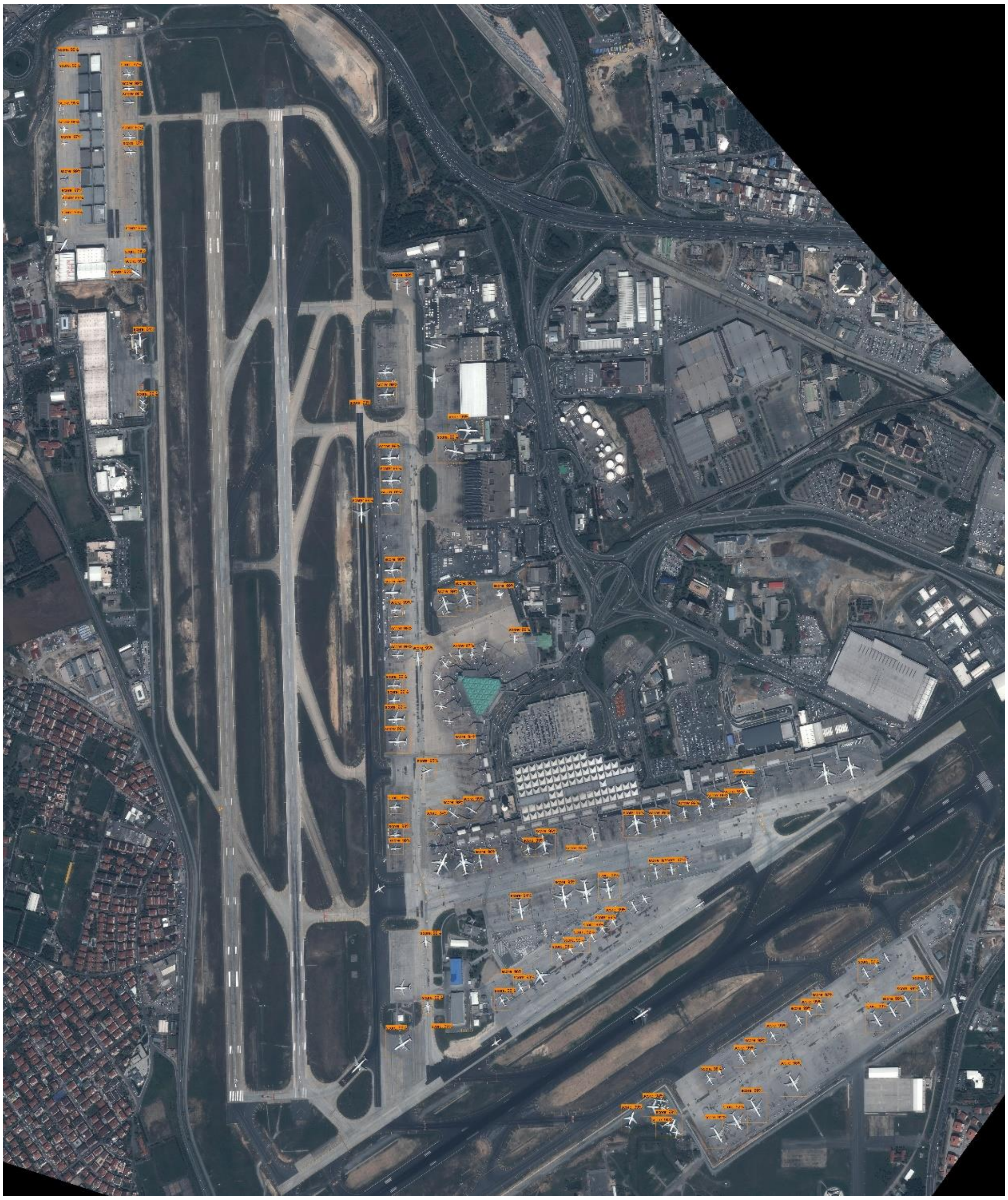


Figure 5.38 : SSD evaluation for the Istanbul Ataturk Airport.

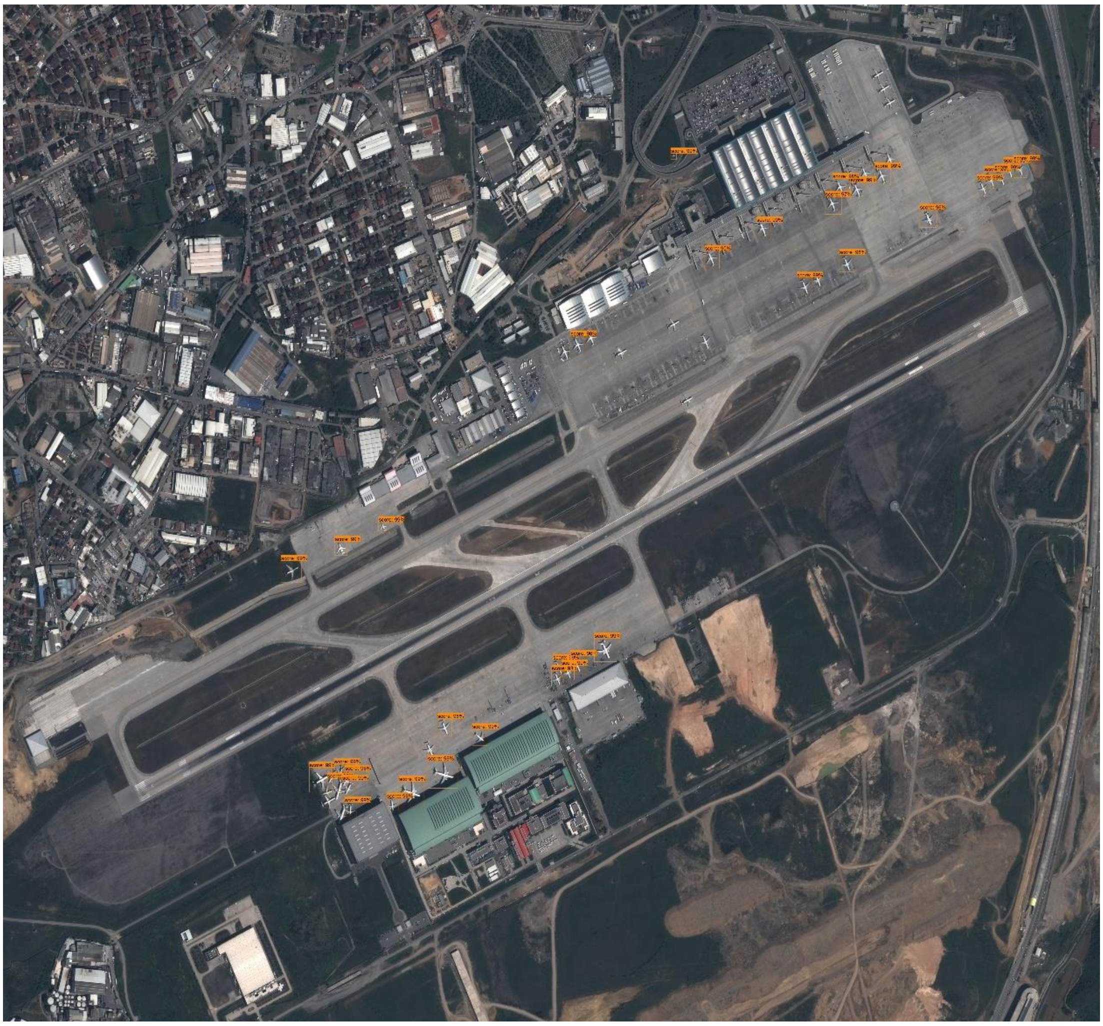


Figure 5.39 : SSD evaluation for the Istanbul Sabiha Gökçen Airport.

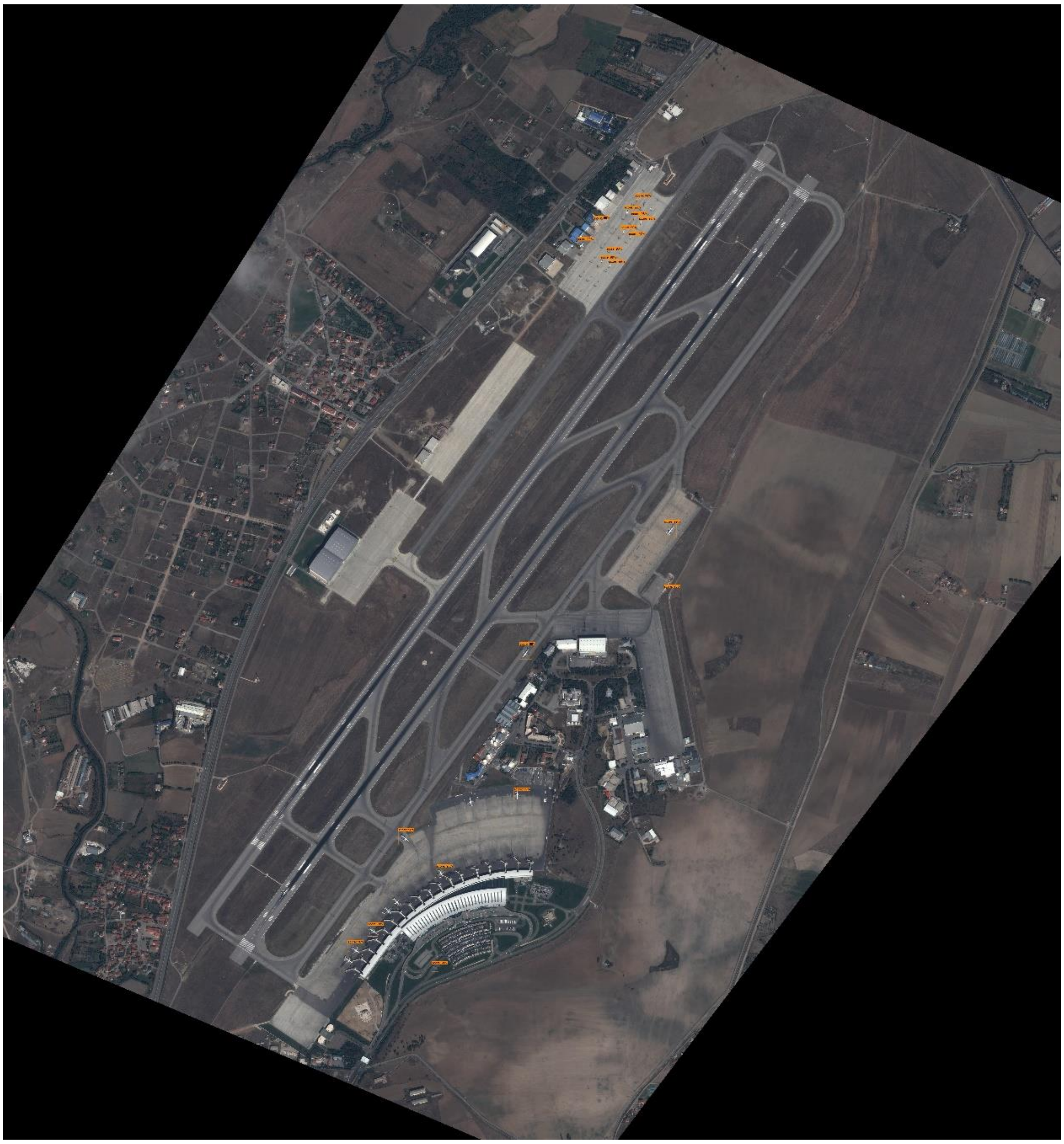


Figure 5.40 : SSD evaluation for the Esenboga Airport.

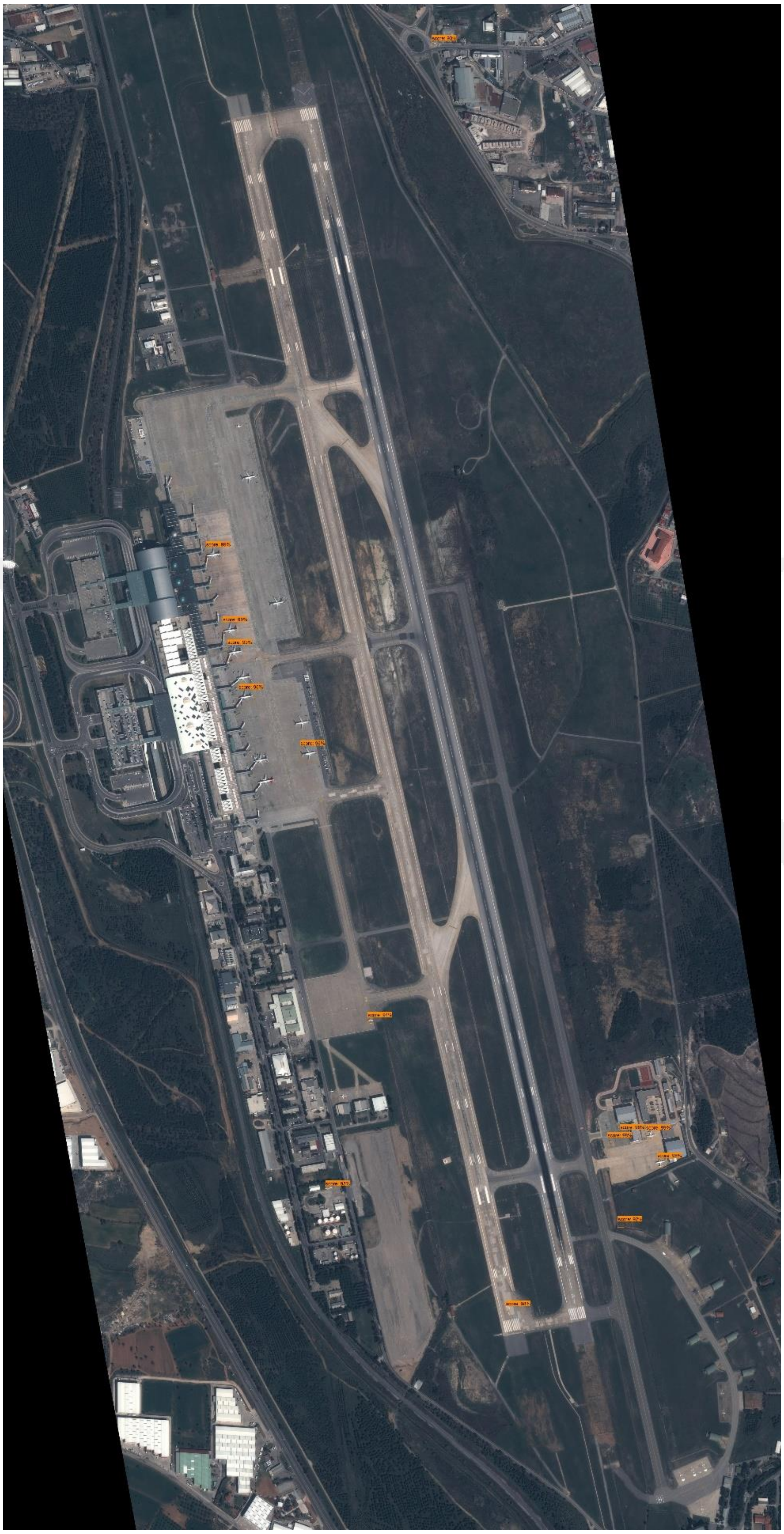


Figure 5.41 : SSD evaluation for the Izmir Adnan Menderes Airport.

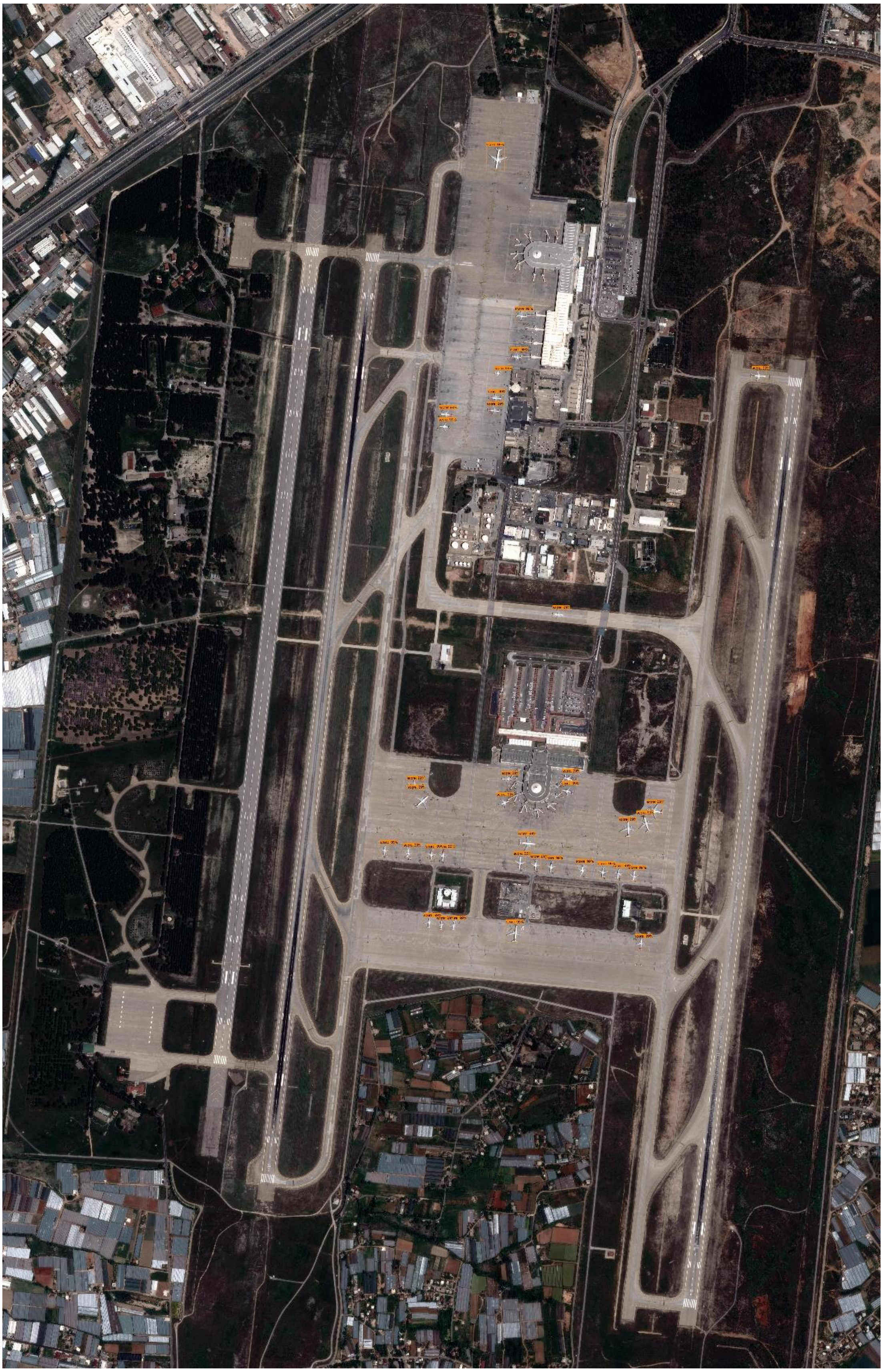


Figure 5.42 : Faster R-CNN evaluation for the Antalya Airport.

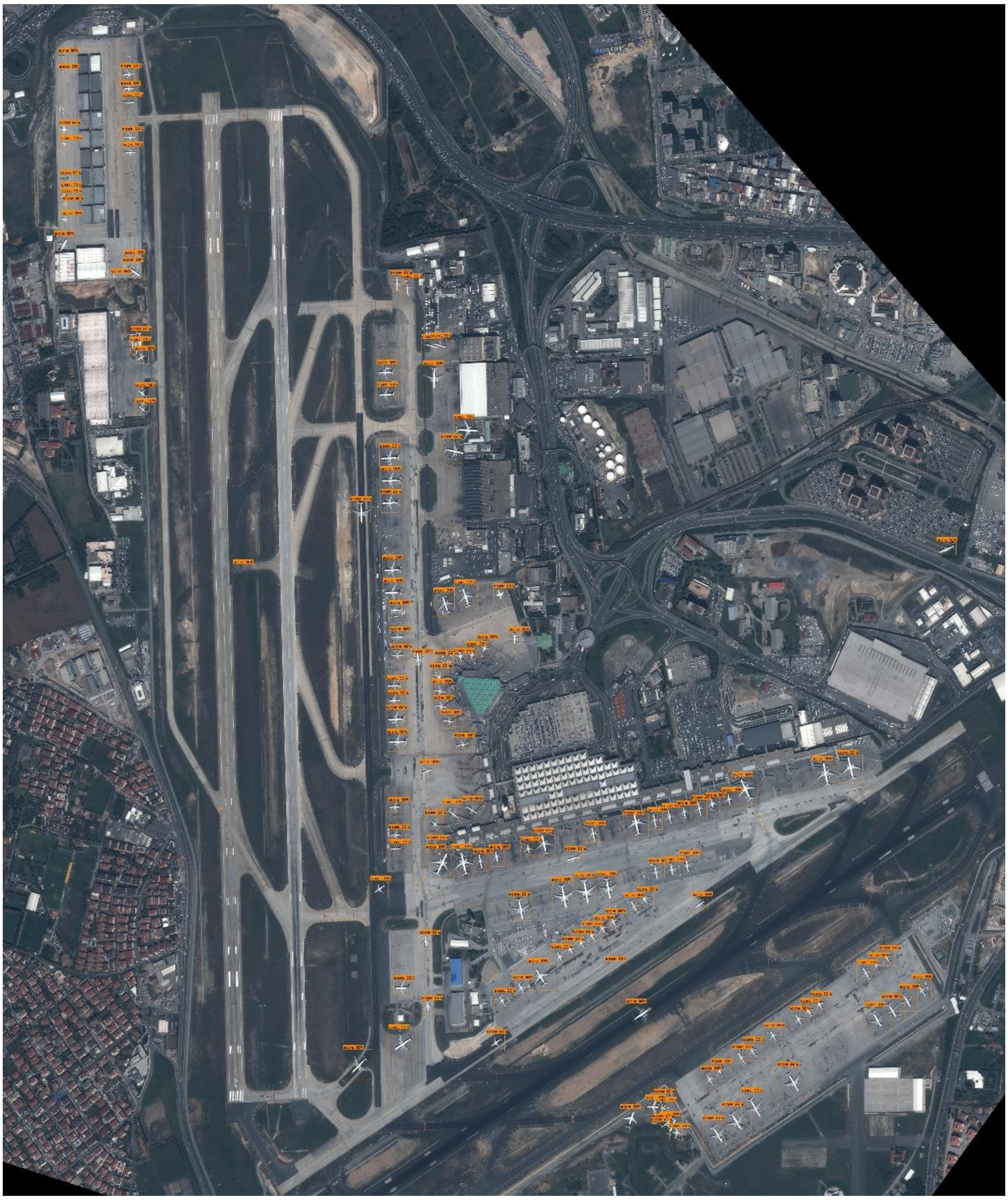


Figure 5.43 : SSD evaluation for the Istanbul Ataturk Airport.

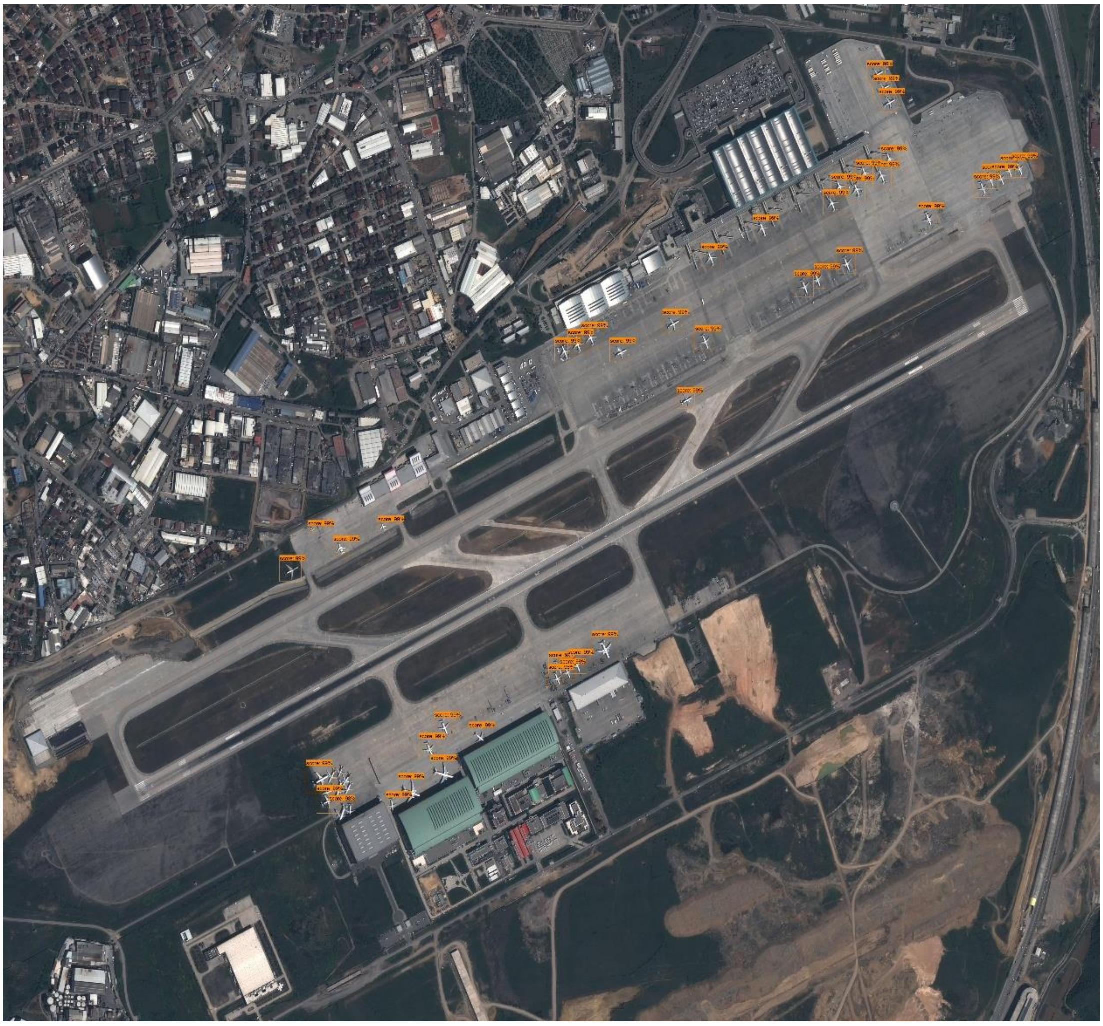


Figure 5.44 : SSD evaluation for the Istanbul Sabiha Gökçen Airport.

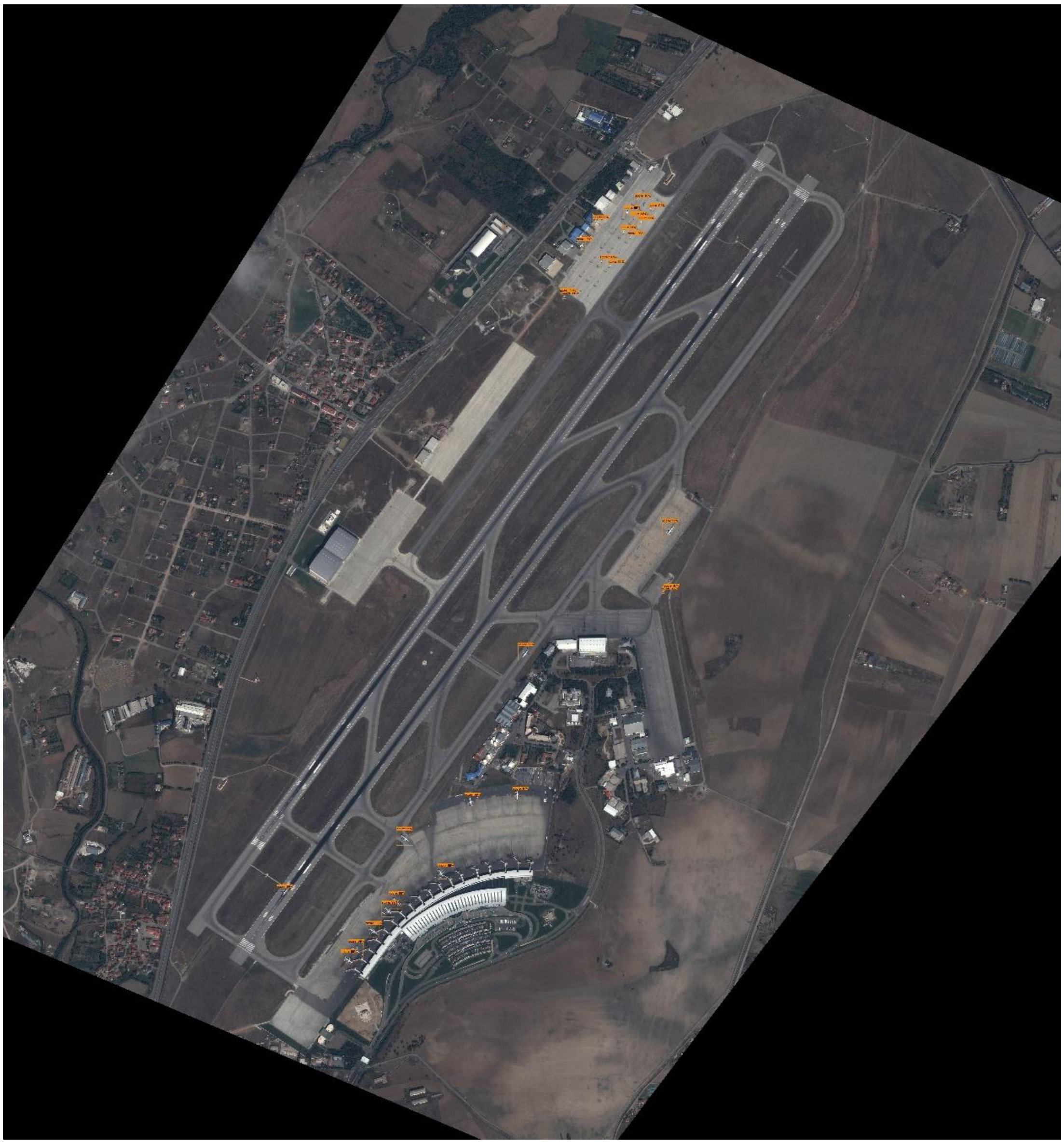


Figure 5.45 : SSD evaluation for the Esenboga Airport.

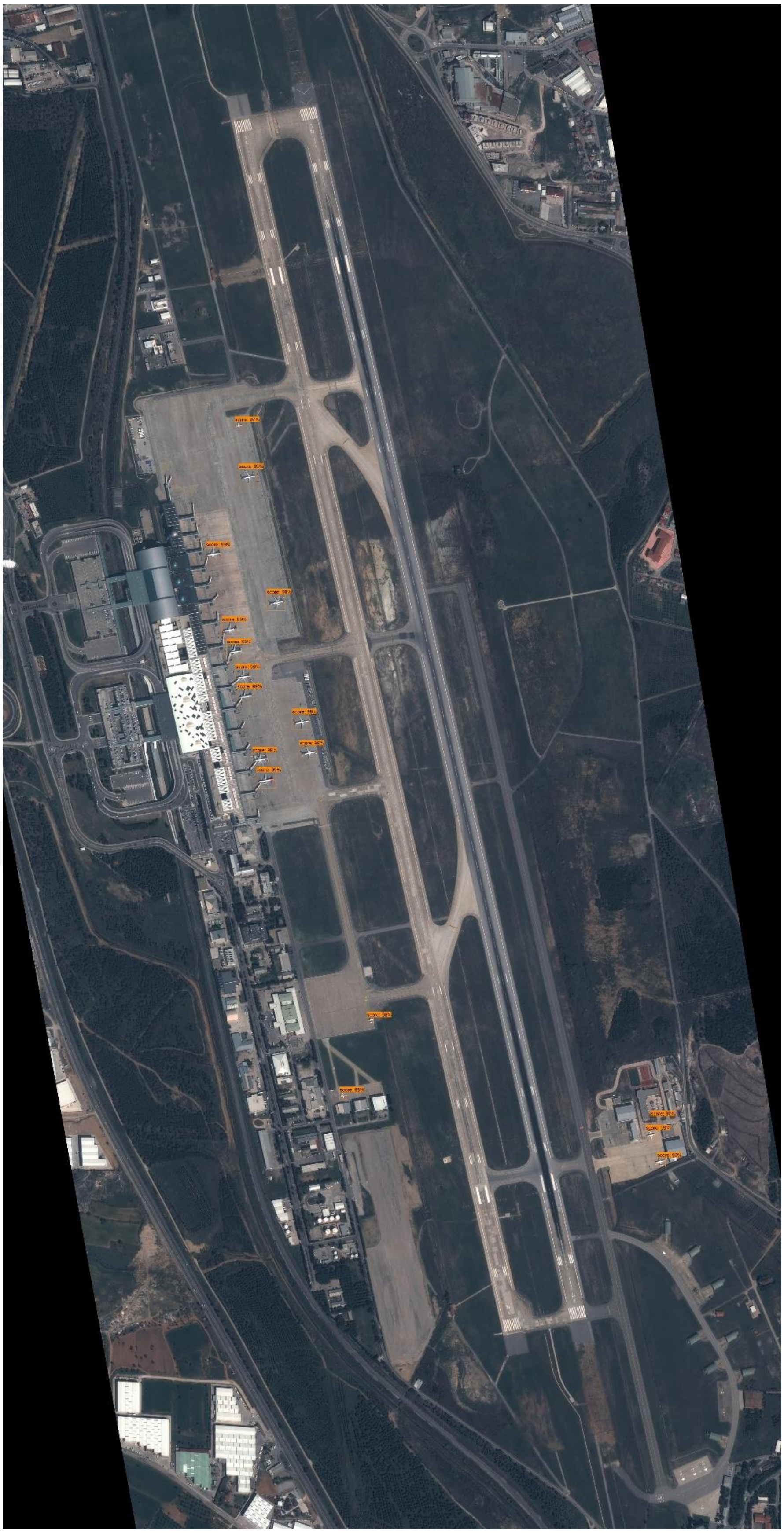


Figure 5.46 : SSD evaluation for the Izmir Adnan Menderes Airport.

6. CONCLUSIONS

With this study, we see the different object detection architectures with the training and evaluation phase after the literature and machine learning techniques review. We obtain the best results with Faster R-CNN network. Yolo-v3 architecture also gave promising results, but SSD could not converge the training data well with low iterations. All the networks have a tendency to learn more with different parameters and more iterations. We can see that Yolo-v3 has faster convergence capability according to the others but the optimization methods also play an important role for this purpose. Although, the worst performance of SSD, it is better to localize objects well.

The imbalance between the object sizes and the diversities also effected the results. In the training of deep learning architectures, imbalances should be avoided or the categories should be divided into finer grains (e.g : airplane, glider, small plane, jet plane, war plane etc).

For the future works we can define the anchor box sizes by weighted clustering according to the sample size of the datasets. Also, for preventing false positives and increasing the recall ratio, we can use all of the networks together and define the offsets of the bounding boxes by averaging predicted bounding boxes. By this way, we think the localization errors could decrease as well. Besides, the different object detection networks can be trained more and used together to obtain better performance. Finding a way to use the ensemble learning methods for object detection architectures could be other improvements. In addition, the object detection networks often use R, G and B bands, because they are mostly developed for natural images. However, satellite imageries can contain much more spectral information. Therefore, in the next studies, object detection networks can be modified and trained to use other band information of multi-band satellite images, and performance can be increased by this extra information.

REFERENCES

[1] **Tello, M., Martinez, C. L.** (2005). A novel algorithm for ship detection in sar imagery based on the wavelet transform, IEEE Trans. Geosci. Remote Sens., vol. 2, no. 2, pp. 201-205.

[2] **Sirmacek, B., Unsalan, C.** (2009). Urban-area and building detection using sift keypoints and graph theory, IEEE Trans. Geosci. Remote Sens., vol. 47, no. 4, pp. 1156-1167.

[3] **LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.** (1998). Gradient-Based Learning Applied to Document Recognition, Proc of the IEEE,

[4] **Lowe, D.G.,** (2004). Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 60, 91–110.

[5] **Dalal, N., Triggs, B.** (2005). Histograms of oriented gradients for human detection. In Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., pp. 886–893.

[6] **Jain, A.K., Ratha, N.K., Lakshmanan, S.** (1997). Object detection using Gabor filters. Pattern Recognit. 30, 295–309.

[7] **Cortes, C., Vapnik, V.** (1995). Support Vector-Newtorks, Machine Learning, 20, 273-297 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

[8**] Url-1** <http://www.scholarpedia.org/article/K-nearest_neighbor>, date retrieved 02.04.2019.

[9] **Krizhevsky, A., Sutskever I., Hinton G.E.** (2012). Imagenet classification with deep convolutional neural networks, in Proc. NIPS, Lake Tahoe, NV, USA, pp. 1097–1105.

[11] **Url-2** <https://arxiv.org/pdf/1409.1556.pdf>, date retrieved 09.04.2019.

[12] **Szegedy C., Liu W., Jia Y. Q., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A.** (2015). Going deeper with convolutions, in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Boston, MA, USA, pp. 1–9.

[13**] 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, pp. 1–9.

[14] **Url-3 <**<https://arxiv.org/abs/1311.2524>>, date retrieved 14.04.2019.

[15] **He K., Zhang X., Ren S., Sun J.** (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904-1916.

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

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

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

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

[20] **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., vol. 88, no. 2, pp. 303-338.

[21] **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, pp. 740-755.

[22] **Blaschke, T.** (2010). Object based image analysis for remote sensing, ISPRS J. Photogramm. Remote Sens. 65, 2-16.

[23] **Blaschke, T., Hay, G.J., Kelly, M., Lang, S., Hofmann, P., Addink, E., Feitosa, R.Q., van der Meer, F., van der Werff, H., van Coillie, F.** (2014). Geographic object-based image analysis–towards a new paradigm, ISPRS J. Photogramm. Remote Sens. 87, 180-191.

[24] **Yildiz C., and Polat E.** (2011**).** Detection of stationary aircrafts from satellite images, in SIU, pp. 518-521.

[25] **Ojala T., Pietikainen M., and Maenpaa T.** (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971–987.

[26] **Grabner, H.; Nguyen, T.T.; Gruber, B.; Bischof, H.** (2008). On-line boosting-based car detection from aerial images, ISPRS J. Photogramm. Remote Sens, 63, 382–396.

[27] **Sirmacek, B., Ünsalan, C.** (2011). A probabilistic framework to detect buildings in aerial and satellite images, IEEE Trans. Geosci. Remote Sens. 49, 211–221.

[28] **Chen L., Jiang Z., Yang J., and Ma Y.** (2012). A coarse-to-fine approach for vehicles detection from aerial images, in Proc. International Conference on Computer Vision in Remote Sensing, pp. 221-225.

[29] **Kembhavi A., Harwood D., and Davis L. S.** (2011). Vehicle detection using partial least squares, IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 6, pp. 1250–1265.

[30] **Zhang, W.; Sun, X.; Fu, K.; Wang, C.; Wang, H.** (2014). Object Detection in High-Resolution Remote Sensing Images Using Rotation Invariant Parts Based Model, IEEE Geosci. Remote Sens. Lett., 11, 74–78.

[31] **Zhang L., Shi Z.W., Yu X.R.** (2014). A hierarchical oil depot detector in high-resolution images with false detection control, in CISP, pp. 530-535.

[32] **Li, F.F., Perona, P.** (2005). A bayesian hierarchical model for learning natural scene categories, In: Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., pp. 524–531.

[33] **Mikolajczyk, K., Schmid, C.** (2001). Indexing based on scale invariant interest points, In: Proc. IEEE Int. Conf. Comput. Vis., pp. 525–531.

[34] **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 Geosci. Remote Sens. Lett., 9, 109–113.

[35] **Cheng, G., Guo, L., Zhao, T., Han, J., Li, H., Fang, J. (**2013a). Automatic landslide detection from remote-sensing imagery using a scene classification method based on BoVW and pLSA, Int. J. Remote Sens. 34, 45–59.

[36] **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 Geosci. Remote Sens. Lett, 9, 109–113.

[37] **Liu, G., Sun, X., Fu, K., Wang, H.** (2013a). Aircraft recognition in high-resolution satellite images using coarse-to-fine shape prior, IEEE Geosci. Remote Sens. Lett. 10, 573–577.

[38] **Malladi C.** (2017). Detection of Objects in Satellite images using Supervised and Unsupervised Learning Methods, Faculty of Computing Blekinge Institute of Technology SE-371 79 Karlskrona Sweden **.**

[39] **Tang J., Deng C., Huang G.H., and Zhao B.** (2014). Compressed-Domain Ship Detection on Spaceborne Optical Image Using Deep Neural Network and Extreme Learning Machine, IEEE Transactions on Geoscience and Remote Sensing, vol. 53, pp. 1174-1183.

[40] **Cheng, G., Han, J., Guo, L., Qian, X., Zhou, P.; Yao, X., Hu, X.** (2013). Object detection in remote sensing imagery using a discriminatively trained mixture model, ISPRS J. Photogramm. Remote Sens., 85, 32–43

[41] **Bi, F.; Zhu, B.; Gao, L.; Bian, M.** (2012). A Visual Search Inspired Computational Model for Ship Detection in Optical Satellite Images, IEEE Geosci. Remote Sens. Lett., 9, 749–753.

[42] **Freund, Y.** (1995). Boosting a weak learning algorithm by majority, Inf. Comput. 121, 256–285.

[43] **Leitloff, J., Hinz, S., Stilla, U.** (2010). Vehicle Detection in Very High Resolution Satellite Images of City Areas, IEEE Trans. Geosci. Remote Sens., 48, 2795–2806.

[44] **Aytekin, Ö., Zongur, U., Halici, U.** (2013). Texture-Based Airport Runway Detection, IEEE Geosci. Remote Sens. Lett., 10, 471–475

[45] **Shi, Z., Yu, X., Jiang, Z., Li, B.** (2014). Ship Detection in High-Resolution Optical Imagery Based on Anomaly Detector and Local Shape Feature, IEEE Trans. Geosci. Remote Sens., 52, 4511–4523.

[46] **Haapanen, R., Ek, A.R., Bauer, M.E., Finley, A.O.** (2004). Delineation of forest/nonforest land use classes using nearest neighbor methods, Remote Sens. Environ. 89, 265–271.

[47] **Zhang, D., Han, J., Cheng, G., Liu, Z., Bu, S., Guo, L.** (2015). Weakly Supervised Learning for Target Detection in Remote Sensing Images, IEEE Geosci. Remote Sens. Lett. 2015, 12, 701–705

[48] **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., vol. 26, pp. 1-16.

[49] **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, vol. 17, no. 2, pp. 1-17.

[50] **Zhang R. , Yao J., Zhang K., Feng C., Zhang J.** (2016). S-cnn-based ship detection from high-resolution remote sensing images, The Int. Arch. of the Photogramm., Remote Sens. and Spat. Inf. Sci., vol. XLI-B7.

[51] **Liu Z. , Yuan L., Weng L., Yang Y.** (2017). A high resolution optical satellite image dataset for ship recognition and some new baselines, in ICPRAM, Porto, Portugal, pp. 324-331.

[52] **Radovic M., Adarkwa O., Wang Q.** (2017). Object recognition in aerial images using convolutional neural networks, J. Imaging, vol. 3, no. 2, pp. 21-31.

[53] **Nie G. H., Zhang P., Niu X., Dou Y., Xia F.** (2017). Ship detection using transfer learned single shot multi box detector, in ITM Web Conf., Guangzhou, China, pp. 1006-1012.

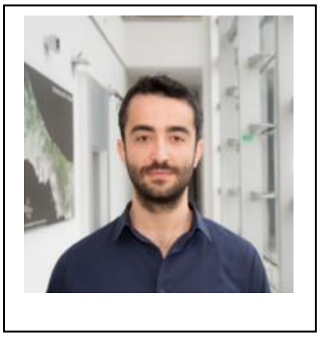
[54] **Wang Y., Wang C., Zhang H.** (2017). Combining single shot multibox detector with transfer learning for ship detection using Sentinel-1 images, in BIGSARDATA, Beijing, China.

[55] **Url-4** <https://pjreddie.com/media/files/papers/YOLOv3.pdf>, date retrieved 17.04.2019.

[56] **Url-5** <http://cocodataset.org/#detection-eval>, date retrieved 17.04.2019.

[57] **Abadi M., …**, (2015). Large-Scale Machine Learning on Heteregeneous Distributed Systems, Google Research

CURRICULUM VITAE



Name Surname : Mehmet Soydaş

Place and Date of Birth : Turkey - 1990

E-Mail : soydas.mehmet@gmail.com

EDUCATION :

- **B.Sc.** : 2014, Yildiz Technical University, Electric-Electronic Faculty, Department of Electronic and Communication Engineering
- **M.Sc.** : 2019, Istanbul Technical University, Informatics Institute, Department of Satellite Communication and Remote Sensing

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2014-2015 Digiturk, Product Development Engineer
- 2017-2018 ITU CSCRS, Image Processing Specialist
- 2018-2019 Arçelik, Project Engineer