ANKARA YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DETECTING AIRPLANES IN UAV IMAGES USING DEEP LEARNING MODEL

M. Sc. Thesis by

Burak ZAHAL

Department of Defense Technology

January, 2025

ANKARA

DETECTING AIRPLANES IN UAV IMAGES USING DEEP LEARNING MODEL

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Master of

Department of Defense Technology

by

Burak ZAHAL

January, 2025

ANKARA

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DETECTING AIRPLANES IN UAV IMAGES USING DEEP LEARNING MODEL"** completed by **BURAK ZAHAL** under the supervision of **Prof. Dr. HÜSEYİN CANBOLAT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Hüseyin CANBOLAT

Supervisor

Prof.Dr. Şerafettin EREL

Jury Member

Prof.Dr. Hasan Şakir BİLGE

Jury Member

Prof.Dr. İlyas ÇANKAYA

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: **Signature: ……..**

Name & Surname: Burak ZAHAL

ACKNOWLEDGMENTS

First of all, I would like to thank my mother and father who supported me throughout my life. I would like to express my respect and gratitude to my advisor Prof. Dr. Hüseyin CANBOLAT for the valuable support and guidance he gave me throughout my thesis work. His experience and valuable suggestions have made a great contribution to this process. I would also like to express my gratitude to my dear wife Esma SAN ZAHAL for the support me throughout this study.

2025, 16 January **Burak ZAHAL**

DETECTING AIRPLANES IN UNMANNED AERIAL VEHICLE IMAGES USING DEEP LEARNING MODEL

ABSTRACT

Today, Unmanned Aerial Vehicles are actively used in many areas such as reconnaissance and surveillance, scientific research, firefighting, agriculture and forest management, communication, trade, air strikes and border security. The fact that they are lower cost compared to aircraft, require less human factors and make correct decisions faster plays an important role in the preference of these vehicles.

In this study, the detection of aircraft from Unmanned Aerial Vehicle images was investigated with deep learning, which is a subfield of Artificial Neural Networks and machine learning. Deep learning models provide successful results with high accuracy rates in complex tasks such as aircraft detection from Unmanned Aerial Vehicle images. Especially in the field of object detection, Convolutional Neural Networks deep learning architectures are known to be the most suitable model for detecting objects in aerial images. Since images taken from Unmanned Aerial Vehicles are usually in different angles, lighting conditions and resolutions, deep learning models can be trained to detect aircraft even in these variable conditions. Creating large and balanced data sets for model training, using data augmentation techniques and appropriate hyperparameter adjustments increase detection success. The development of this technology plays a critical role in air traffic management, military operations and search and rescue operations in disaster situations.

As a result of the study, it was seen that Unmanned Aerial Vehicles, deep learning and object detection studies will have a very critical role in the future. With the software and hardware developments in this field, autonomous air vehicles and unmanned air traffic systems will be equipped with artificial intelligence systems that can make more independent and faster decisions. The need for use in many places, especially military applications, will increase and become widespread.

Keywords: Artificial neural network, unmanned aerial vehicle, aircraft, algorithm, artificial intelligence, convolutional neural networks, deep learning, object detecetion

DERİN ÖĞRENME MODELİ KULLANILARAK İNSANSIZ HAVA ARACI GÖRÜNTÜLERİNDEN UÇAKLARIN TESPİTİ

ÖZET

Günümüzde insansız hava araçları, keşif ve gözetleme, bilimsel araştırmalar, yangın söndürme, tarım ve orman yönetimi, iletişim, ticaret, hava saldırıları ve sınır güvenliği gibi birçok alanda aktif olarak kullanılmaktadır. Uçaklara kıyasla daha düşük maliyetli olmaları, insan faktörüne daha az ihtiyaç duymaları ve daha hızlı doğru kararlar almaları, bu araçların tercih edilmesinde önemli rol oynamaktadır. Bu avantajlar, insansız hava araçları kullanımının hızla yaygınlaşmasına katkı sağlamaktadır.

Bu çalışmada yapay sinir ağları ve makine öğrenmesinin alt çalışma alanı olan derin öğrenme ile insansız hava araçları görüntülerinden uçakların tespiti incelenmiştir. Derin öğrenme modelleri, insansız hava araçları görüntülerinden uçak tespiti gibi karmaşık görevlerde yüksek doğruluk oranlarıyla başarılı sonuçlar sunmaktadır. Özellikle nesne tespiti alanında kullanılan evrişimli sinir ağları derin öğrenme mimarileri, hava görüntülerindeki nesneleri tanımlamak için en uygun model olarak bilinmektedir. İnsansız hava araçlarından alınan görüntüler genellikle farklı açılar, ışık koşulları ve çözünürlüklerde olduğu için derin öğrenme modelleri, bu değişken koşullarda dahi uçakları tespit edebilecek şekilde eğitilebilmektedir. Modelin eğitimi için büyük ve dengeli veri kümeleri oluşturulması, veri artırma tekniklerinin kullanılması ve uygun hiperparametre ayarlamaları, tespit başarısını artırmaktadır. Bu teknolojinin geliştirilmesi, hava trafiği yönetimi, askeri operasyonlar ve afet durumlarında arama-kurtarma çalışmalarında kritik bir rol oynamaktadır.

Çalışma sonucunda insansız hava araçları, derin öğrenme ve nesne tespitine yönelik çalışmaların gelecekte çok kritik bir role sahip olacağı görülmüştür. Bu alanda yapılan yazılımsal ve donanımsal gelişmelerle otonom hava araçları ve insansız hava trafik sistemleri daha bağımsız ve hızlı kararlar alabilecek yapay zeka sistemleriyle donatılacaktır. Başta askeri uygulamalar olmak üzere bir çok yerde kullanım gereksinimi artacak ve yaygınlaşacaktır.

Anahtar Kelimeler: Yapay sinir ağları, insansız hava araçları, uçak, algoritma, yapay zeka, evrişimli sinir ağları, derin öğrenme, nesne algılama

CONTENTS

NOMENCLATURE

Roman Letter Symbols

b	bias (slope value)
e	Exponential value
f	Function
h	Hidden layer data
i	Index , Input layer data
n	A series of input data up to n numbers
o	Output layer data
x_0	Input value
w_0	Weight in the dendrite

Greek Letter Symbols

Σ	The sum of a set of numbers or expressions
σ	Tanh function
Φ	Sigmoid function

Subscripts

cm	Centimeter
ft	Feet
g	Gramme
h	Hour
kg	Kilogramme
km	Kilometer
kt	Knots
m	Meter
max	Maximum

Acronyms

AGM	Air to Ground Missile
AGL	Above Ground Level
AI	Artificial Inteligence
ANN	Artificial Neural Networks
AP	Average Precision
BoF	Bag of Freebies

BoS	Bag of Specials
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CPU	Central Process Unit
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Networks
DOTA	Dataset for Object Detection in Aerial
DoD	Department of Defense
FAA	Federal Aviation Administration
FAIR	Facebook's Artificial Intelligence Research
FNN	Feedforward Neural Networks
FPR	False Positive Rate
FPS	Frame Per Second
GB	Gigabyte
GCS	Ground Control Station
GELAN	Generalized Efficient Layer Aggregation Network
GPU	Graphics Processing Unit
GPS	Global Positioning System
HALE	High Altitude Long Endurance
IoU	Intersection Over Union
IoT	Internet of Things
LSTM	Long Short-Term Memory
MALE	Medium Altitude Long Endurance
mAP	Mean Average Precision
MLP	Multi Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
MQ	Multi Q
MQM	Multi Q Missile
MR	Miss Rate
MRI	Magnetic Resonance Imaging
MSL	Mean Sea Level

MTOW	Maximum. Take-Off Weight
NATO	North Atlantic Treaty Organization
NMS	Non-Maximum Suppression
OQ	Ordnance Q
Q	Quail (category of UAV)
PGI	Programmable Gradient Information
RC	Radio Controlled
R-CNN	Region-Based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
SSD	Single Shot Detectors
TPR	True Positive Rate
TPU	Tensor Processing Unit
UAS	Unmanned aircraft system
UAV	Unmanned Aerial Vehicle
US	United State
VOC	Visual Object Classes
YOLO	You Only Look Once

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Today, with the development of unmanned aerial vehicles (UAVs), camera, sensor and communication technology, the ability to view very large areas in high spatial detail has enabled the widespread use of UAV images for the detection of different objects, especially aircraft detection. As the number of UAVs with high-resolution optical cameras has increased, the temporal resolution has increased, and thus the number and speed of data obtained has also increased. However, UAV images have a complex structure and in some cases, the objects targeted to be detected are very small compared to the image, making object detection difficult.

In object detection, some objects can be detected with higher reliability and ease due to their physical properties. For this reason, a civilian and military aircraft detection study was conducted from UAV images, considering that it may be useful in the field of defense industry. The most common problem in object detection studies is the lack of meaningful data and datasets. Studies generally used in the literature include civil aircraft images obtained from civil airports. It is thought that the military aircraft image dataset used in this study will also contribute to the literature. Developing technology has created new alternatives for aerial image acquisition. In this context, UAV technology emerges as a new image acquisition platform. As the name suggests, there is no pilot sitting on the image acquisition platform in this technology, and the platform is controlled remotely.

Nowadays, object detection can be done more effectively and quickly by developing deep learning architectures. Thanks to the ongoing studies in this field, object detection has become a popular field in recent years. It has become an element that makes our lives easier in many areas. In addition, as a result of the studies carried out successfully in our country, especially in the last decade, different types of unmanned aerial vehicles (UAV) are successfully applied in the military field. Detection of moving objects in UAV images will be an important factor that increases mobility and ensures successful results, especially in unmanned aerial vehicles (UAVs).

In a UAV image taken from a bird's eye view, many objects are tracked and detected. Apart from military areas, it is also widely used in areas such as human recognition, vehicle security systems and solving traffic problems. Deep learning is a new field developing within Artificial Neural Networks (ANN). This process is a machine learning method that uses libraries such as Pytorch, Tensorflow, OpenCV, etc. developed in recent years based on Python, C and C++ languages for object detection. The basis of deep learning is a perception state consisting of many layers called Multi Layer Perceptron (MLP).

Another deep learning system, Convolutional Neural Networks (CNN), is based on the YOLO Model and R-CNN based Fast R-CNN, Faster R-CNN and the latest version Mask R-CNN models are preferred in object detection. These detect objects and model the detected object by processing it.

Recently, with the development of technology, unmanned aerial vehicles (UAVs), which have spread rapidly among sensing platforms, have begun to be used for object detecting, remote sensing and photogrammetry purposes. These platforms have begun to be used in many areas such as military, agriculture, archaeology and disaster management (earthquake, forest fire, etc.) due to their low-cost, re-measurable, fast and precise monitoring features, especially in limited work areas. Initially used mostly for military and surveillance purposes, unmanned aerial vehicles have recently become more accessible and their costs have decreased with the advancement of technology. Thus, they have begun to be used in scientific activities and engineering fields, most notably in civilian use. In addition to their use in military fields, unmanned aerial vehicles are frequently used in fields such as mapping, meteorology, mining fields, natural disasters, construction, and archaeology.

Unmanned aerial vehicles collect data in these areas with their active or passive sensors. These sensors they carry provide high data accuracy. UAVs offer lower cost and faster analytical solutions compared to satellite images and provide an effective and efficient service for the data requested by the user. In this context, the systems carried by unmanned aerial vehicles and their areas of use will also be explained in our studies.

In this thesis, the use of unmanned aerial vehicles for remote sensing and object detection purposes has been examined. The methods of collecting and producing this data have been explained. In addition, projects and expectations regarding the areas of use of unmanned aerial vehicles in the coming days have been mentioned.

1.1 Thesis Overview

The sections of the thesis I have prepared are briefly explained below;

Chapter 1; In this section, the content of the thesis is mentioned and the reason for the study is explained. What will be explained in the thesis is explained and similar studies in the literature are briefly summarized.

Chapter 2; The definition, historical process, classification of unmanned aerial vehicles, which are the main tools of the thesis study, and the main systems of these vehicles are examined.

Chapter 3; The definition, structure and types of neural networks are explained, and information about deep learning is given.

Chapter 4; Information about object detection and how it is done, the data required for the thesis study and how this data is used in which stages are presented.

Chapter 5; The analysis of the experiments conducted for the thesis study and the analysis results are shown.

Chapter 6; The conclusion reached in line with the objectives of the thesis and ideas about future studies are given.

1.2 Literature Review

In the article written by five friends (2023), Guangyi Tang, Jianjun Ni, Yonghao Zhao, Yang Gu and Weidong Cao focused on researching articles on deep learning-based UAV object detection published in the past five years. First, all articles were classified based on one-stage and two-stage object detection, and then the common problems in UAV object detection were summarized. In addition, research was conducted on

applications such as geological environment exploration, traffic monitoring and precision agriculture. Finally, classical UAV object detection algorithms based on deep learning were selected to target these common problems. Considering the above selection criteria, they explained the improvement methods proposed by various scientists. At present, the interest in UAV object detection algorithms is increasing, and good detection results have been achieved with existing algorithms. However, existing algorithms still have false alarms and missed detection problems in dense environment or in an environment with a large number of similar objects. They summarized the advantages and shortcomings of existing methods in solving problems such as the increase of small objects, complex background, object rotation, scale change and category imbalance in UAV object detection. Datasets in this area have also been introduced [1].

In the study conducted by Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun (2021), the YOLOX model was compared with YOLOv5 and YOLOv4 to evaluate the performance of object detection models. The researchers used the COCO 2017 dataset, which was divided into 118,000 images for training and 5,000 images for testing. The input image size was set to 640x640 for most experiments. Data augmentation techniques such as mixup and mosaic augmentation were applied to the images. For optimization, the Stochastic Gradient Descent (SGD) algorithm was employed with a batch size of 128. The experiments were conducted on high-performance hardware. Average Precision (AP) was used as the primary evaluation metric to measure model performance. The study compared various versions of YOLOX, YOLOv5, and YOLOv4, highlighting how these models performed under different architectural designs and data processing methods [2].

Basim Azam, Muhammad Jaleed Khan, Farrukh Aziz Bhatti, Abdur Rahman M. Maud, Syed Fawad Hussain, Ali Javed Hashmi, and Khurram Khurshid (2022) compared the performance of YOLOv3 and SSD models. The evaluation was conducted using a dataset of 26 satellite images with varying resolutions ranging from 565x369 to 1484x865. The dataset was split into 80% for training and 20% for validation. Mean Average Precision (mAP) was used as the performance metric to assess the models. The results showed that the SSD model achieved an mAP of 0.55,

while the YOLOv3 model outperformed it with an mAP of 0.77. This study highlights that YOLOv3 demonstrates superior object detection performance compared to the SSD model [3].

In their 2018 research, Volodymyr Kharchenko and Iurii Chyrka employed the YOLOv3 algorithm to detect airplanes. They utilized the Darknet53 architecture as the backbone of the model. The training dataset consisted of 204 images with a resolution of 416x416, sourced from Google Maps. For validation, they collected an additional 50 images from Google Maps, resizing them to 224x224 and 608x608 for evaluation purposes. The experiments were conducted with a batch size of 64. The resulting mean Average Precision (mAP) scores were 0.817 for images at 224x224 resolution, 0.907 for 416x416 resolution, and 0.909 for 608x608 resolution. The findings indicate that increasing image resolution enhances mAP performance but comes at the cost of slower processing speeds [4].

In the project executed by Mustafa Burgaz and Cafer Budak from Dicle University (2022), it is aimed to detect weapons from images obtained from Unmanned Aerial Vehicle (UAV) by using deep learning algorithms. Images were obtained from the UAV at 200 different angles and heights. Images from different angles and heights obtained from the unmanned aerial vehicle are trained by Regional Based Convolutional Neural Networks (R-CNN) and Residual Neural Network (ResNet). Two-thirds of the images we obtained were split into training images and one-third into test images. The feature maps extracted from the images used for training were compared with the test images. By bringing these compared images closer to the desired images, 99% of the desired image detection is achieved. Performance evaluation of the algorithms was made using Loss plot, mAP curves, Precision, Recall and F1-Score. The performance evaluation of the detected images is discussed, and the success of deep learning algorithms used in object detection is presented. The ResNet model showed higher performance with 64% accuracy, 94% recall and 76% F1 score [5].

CHAPTER 2

UNMANNED AERIAL VEHICLES

Unmanned Aerial Vehicle (UAV); is an aircraft that does not have a passenger or a pilot, has equipment (video camera, camera, GNSS, laser scanning device, etc.) according to its intended use, and can perform its duties remotely controlled and/or automatically. Initially used for military missions, UAVs have now become indispensable elements for most militaries. As in the rest of the world, the use of UAVs in our country is rapidly increasing day by day for military, civil (commercial or hobby) and scientific professional purposes [6].

In some foreign sources, it is defined as "Unmanned Aerial Vehicle System". The unmanned aircraft system (UAS) was first used by the United States Department of Defense (DoD) and the United States Federal Aviation Administration (FAA) in 2005. UAVs have been given different names since their emergence. Some of these are pilotless aircraft, remotely piloted vehicles, pilotless planes, and flying robots. Basically, UAVs are "drones" with independent control systems equipped with the latest technology.

Figure 2.1 MQ-9 Reaper [7]

These aircraft are flown by remote control or through systems with software integrated with the Global Positioning System (GPS). Many different unmanned aerial vehicles

are produced in terms of size, shape, configuration and character. Along with software, autonomous drones also use an integrated set of advanced technologies that enable them to perform tasks without human intervention, such as cloud computing, computer-aided, thermal sensors, artificial intelligence, machine learning, and deep learning [7].

2.1 History of UAV

The first known unmanned aerial vehicle in history was used by the Austrians in their attack on Venice, Italy, on August 22, 1849. It was the Volcano, a 200-pilotless balloon loaded with approximately 15 kilograms of explosives and a timed fuse. Some of these balloons, released from Austrian ships, moved towards the target, while others, under the influence of the wind, caused damage to the Austrian troops (Figure 2.2). These balloons were later used for the first aerial reconnaissance and observation purposes during the Spanish-American War in 1898, by attaching cameras.

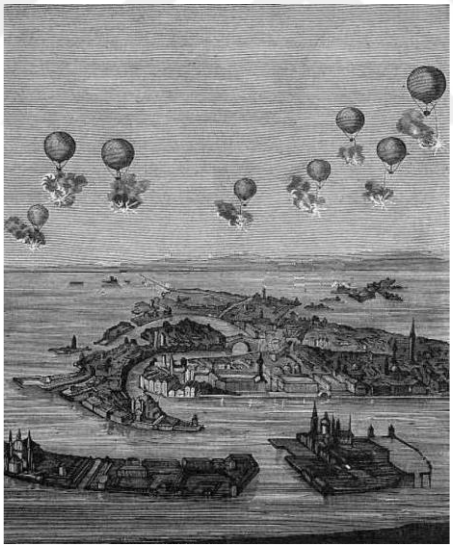


Figure 2.2 Venice balloon attack, 1849 [8]

During World War I, due to the lack of technological developments, unmanned aviation experienced problems with remote control, accurate movements and autonomous driving. First used by Elmer Ambrose Sperry for maritime activities, large and heavy gyroscopes were developed by Glenn Hammond Curtiss in 1911, resulting in a system that could move on three axes, was smaller and had servo motors. The first

known unmanned aircraft was developed by Archibald Montgomery Low in 1916. This aircraft, named Ruston Proctor Aerial Target, used radio control technique. The first unmanned use was carried out with the "Hewitt-Sperry" gyroscope-controlled automatic aircraft known as "flying bombs". In November 1917, the "automatic flying aircraft" became the official aircraft of the US Armed Forces and made its first flight in 1918 [8].

The development of UAVs slowed down for a while due to the First World War, and in 1922, the ‘De Bothezat Helicopter’ (First manned) was built by Dr. Georg de Bothezat and Ivan Jerome. In 1930, the first radio-controlled unmanned aerial vehicle, Queen Bee, was given to the British Royal Navy. The purpose of this drone was developed so that pilots could easily target. The United States’ work in this field was in 1939 with the Radioplane OQ-2, a remote-controlled aerial vehicle. Later, in 1950, the Radioplane MQM-33, which began reconnaissance missions in the American Army, was the first target aircraft. The first series MQ-1 Predator modern UAV, which was used for many years, was developed in 1995 in partnership with General Atomics and the Pentagon [6].

Over time, UAVs have proven to be cheaper, more capable war machines that can be deployed without posing a risk to aircrews. Initially designed as surveillance aircraft, by the end of the 20th century, the MQ-1 Predator, capable of launching air-to-ground missiles such as the AGM-114 Hellfire, had begun production.

Figure 2.3 MQ Predator [9]

The European Union conducted a project on UAVs called CAPECON between 2002-2005. More than 50 countries, primarily the United States Air Force, used UAVs until 2013. In addition to major powers, Turkey, Iran, Israel, Pakistan, and other developing countries designed and produced their own UAVs. Today, there is no clear information about the type and production points.

2.2 UAV Classification

Since UAVs have many structures and uses, it is not possible to define them in a single way. In order to make a classification, we must consider many factors and variables. These include structure, engine, size and altitude. Below, we will define the main headings of these categories and show the main headings in tables.

These categories are;

- The usage and application area of the aircraft;
- The type of control system;
- Flight rules;
- General category (UAV maximum take-off weight, range, airtime, maximum altitude values);
- The type of aircraft;
- The wing type;
- The direction of the lift force at take-off and landing;
- Flight altitude ;
- The engine type of the aircraft;
- The fuel system [10].

The classifications of UAVs include:

Tablo 2.1 UAVs are classified by range and endurance [7]

Category	Very Close range UAVs	Close range UAVs	Short range UAVs	Medium range UAVs	Long range UAVs
Range (km)	Less than 5	5 - 50	50 - 150	150 - 650	More than 650
Endurance (hr)	0.5 - 0.75	1 - 6	8 - 12	12 - 48	More than 48

Tablo 2.2 UAVs are classified by size [7]

Category	Micro/Very Small UAVs	Mini/Small UAVs	Medium UAVs	Long UAVs
Length/Wingspan (m)	0.5	0.5 - 2	5 -10	Bigger than 10

In Turkey, all rules and instructions regarding UAVs are controlled and implemented by the General Directorate of Civil Aviation.

Tablo 2.3 UAV classification of Türkiye [11]

Class	**MTOW** (Max. Take-Off Weight)
UAV0	Between 500 g and 4 kg
UAV1	Between 5 kg and 25 kg
UAV2	Between 26 and 150 kg
UAV3	More than 150 kg

The North Atlantic Treaty Organization (NATO) has its own criteria and categorization methods. The military often uses a tier system for classification, such as weight categories. During September 2009 NATO's UAV meeting, UAVs were divided into 3 weight-based categories which each class being sub-categorized according to specific parameters [12].

Tablo 2.4 UAV classification of NATO [12]

CLASS	Category	Normal Operating Altitude	Normal Mission Radius	Example Platform
CLASS I (less than 150 kg)	Micro (0-2 kg)	Up to 200 ft AGL	5 km	Black Hornet
	Mini (2-15 kg)	Up to 3K ft AGL	25 km	DJI Mavic 3
	Small (15-150 kg)	Up to 5K ft AGL	50 km	Scan Eagle
CLASS II (150 kg to 600 kg)	Tactical	Up to 10K ft AGL	200 km	Hermes 450
CLASS III (more than 600 kg)	MALE (Medium Altitude Long Endurance)	Up to 45K ft MSL	Unlimited	Heron
	HALE (High Altitude Long Endurance)	Up to 65K ft	Unlimited	Global Hawk
	Strike /Combat	Up to 65K ft	Unlimited	MQ9 Reaper

2.3 UAV Components

An Unmanned Aerial Vehicle (UAV) is an aircraft that can fly a without pilot; that is remote controlled. So it have mainly hardware and soft ware components. UAV parts determine the functionality and performance of these specialized vehicles. The vast majority of UAVs have similar systems: body and wing, battery and power unit, camera, sensors, flight and ground control units that all UAVs have.

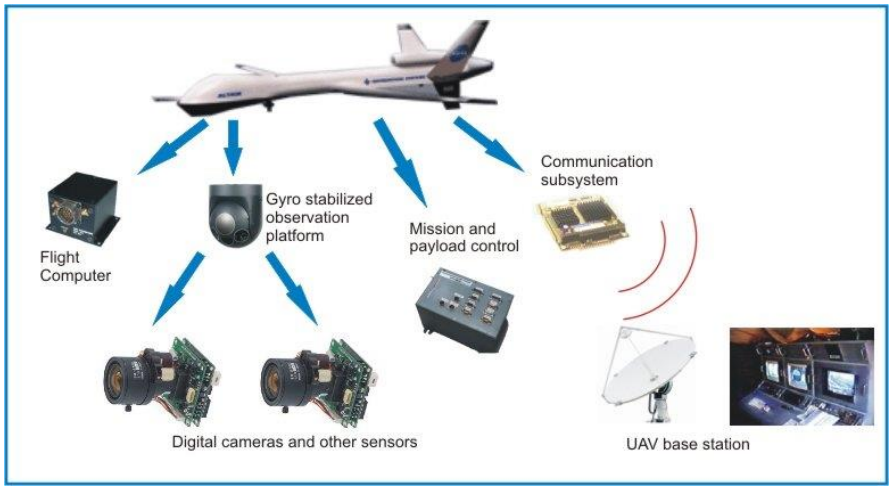


Figure 2.4 Main components of a UAV system [13]

The physical dimensions of UAVs, such as body and wings, vary depending on the category they are in. They can be a few centimeters or up to 5-10 meters. The structural dimensions and shapes of UAVs affect their aerodynamic performance. The use of lightweight and durable materials is important for longer ranges and higher speeds in difficult conditions and for long periods of time.

Batteries are one of the most important components of aircraft. In small-sized radio-controlled (RC) aircraft, the thrust engines are generally Lithium-based batteries. In addition, polymer batteries are also used. Long-range UAVs have their own internal combustion engines. The electric power systems consist of the alternator and batteries produced by the engines. In case of any failure, there are temporary batteries on them to continue critical demands and to be able to exit the emergency situation. Users allocate very high budgets for these power systems. UAVs perform their tasks and perform safe flights through the sensors and cameras they have. They perform their activities with positioning, communication, scanning and detection radars. High quality and resolution cameras, imaging devices for different purposes and other sensors allow the UAV to monitor and collect data as long as it remains in the air.

All UAV systems used today have an operating system to analyze the collected data and calculate flight parameters. These make very fast decisions and enable the powerful UAV to perform its real-time tasks. The flight control system allows the UAV to fly stably in the air, receive commands and respond to these commands. While small-sized UAVs perform their tasks via radio link antennas, large UAVs, which are of strategic importance to countries, perform their tasks via satellites. These electronic systems used allow data to be transferred very quickly and accurately, allowing errors to be minimized.

Ground Control Station (GCS) is a software combination that allows the UAV to communicate with a ground-based computer, telephone or antenna/satellite transmitter. This software communicates with the UAV via wireless RF/telemetry. It displays real-time data on the UAV's operations and location. Data from cameras and sensors are monitored and analyzed via GCS and necessary precautions are taken. Flight parameters are entered before or after the flight to perform a fast and effective operation.

CHAPTER 3

NEURAL NETWORK

3.1 Artificial Neural Networks (ANN)

Neural networks, also known as artificial neural networks (ANN) is a type of machine learning algorithm designed according to the structure and task similar to the human brain. It is a method that teaches neuroscience processing by providing data input to computers. It is designed to detect, diagnose and predict data or make decisions with written code and programming. Neural networks process and transfer data, in this way it is widely used in image and speech recognition, natural language processing and other fields.

Neural networks are a fundamental tool for artificial intelligence and machine learning. They examine the information entered, learn, make predictions, classify the inputs according to a predetermined system and make appropriate generalizations. Thus, they can recognize images, sounds and shapes.

Neural networks train and learn themselves using previous data. In other words, the more examples are trained, the more useful they are to reach the desired result. On the other hand, Neural Networks process the raw data they receive and make up-to-date generalizations. Thus, in unknown and unpredictable situations, they automatically make the right or closest decisions to the right without the need for a human. Driverless vehicles, natural language recognition, object or image recognition are the areas where neural networks are most frequently used [14].

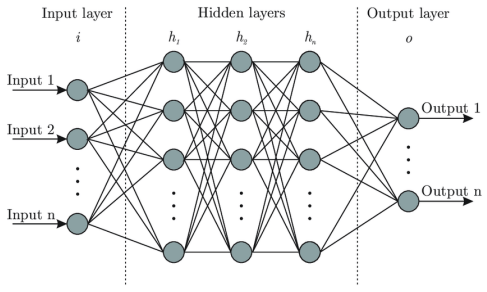


Figure 3.1 Neural network architecture [14]

Each neural network consists of layers of nodes or artificial neurons; It has a 3-layer structure. These are; an input layer, several hidden layers and an output layer. Each node is connected to another node and establishes a relationship between them. If the output of any node is above the specified threshold value, that node is activated and data is sent to the next layer of the network. Otherwise, no data is transmitted to the next layer of the network [15]. The structure of an entire artificial neural network consists of:

- Input layer: Receives input data via nodes, examines, uses, analyzes or classifies the data. Then, it transmits it to the hidden neuron layer via synapses. The more data input in this layer, the more nodes there are. In this layer, the data contains matrices of values belonging to problems such as sound, object and image detection and processing, depending on the type of experiment.

- Hidden layer: The hidden layer is like a black box. It is impossible for anyone to understand the data here without analyzing it. It receives the data it receives from the input layer as requested, analyzes it and categorizes it by performing certain mathematical operations. This processed data is transmitted to another hidden layer or output layer via nodes.

- Output layer: Receives data sent from the hidden layer and outputs the mathematical result. Each model can have multiple input data, but only one output data. This layer can have many nodes. For example, if we have a binary (yes/no) classification problem, the output layer will have a single output node that will give the result as 1 or 0 [16].

In artificial neural networks, which are transmitted just like the human brain, each node is its own linear regression model consisting of input data, weights, a bias (or threshold), and an output. The formula will look like this:

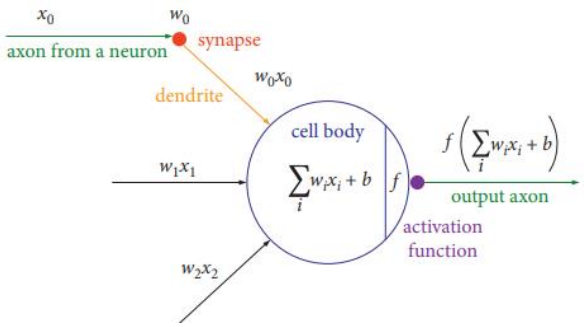


Figure 3.2 Structure of artificial neuron [17]

Our weights are assigned to each layer along the channels called dendrites. The larger of these weights contributes more to the output data. In addition, we have an input value (x_0) that may have come from another neuron entering the dendrites. After our input value and our weight in the dendrite (w_0) are multiplied (w_0x_0), it is transmitted to the neuron and then added with a bias (b) and the activation function is transferred to the output. In this process, the weights from all dendrites and the input products are added with the weight addition process. This output can be the final output or the input of another cell. If this output exceeds a certain threshold, it "fires" (or activates) the node and transmits the data to the next layer in the network. This results in the output of a node becoming the input of the next node [15,18].

In the operations of Artificial Neural Networks, the parameters w (weight parameter) and b (slope value) have the most critical point for the best results of the model. Weight is a value that carries data when neurons are connected to each other. The higher this value, the larger the weight and the more important the neurons in the input layer are. Bias is also a weight. There are some factors in a decision making process that are unpredictable or unexpected. Similarly in a Neural Network, these unpredictable or unobservable factors are called biases. Every neuron that is not in the input layer has a bias attached to it and bias, like a weight, carries a value [19].

3.2 Types Of Neural Networks

Neural networks consist of different types and layers depending on the purpose for which they will be used. The most common categories are as follows;

3.2.1 Feedforward Neural Networks (FNN)

The most basic neural network structure in the neural network algorithm. In addition to the input and output layers, there are several hidden layers. The data to be trained follows the process in one direction without going into a loop. Areas of use are tasks such as image and speech recognition.

3.2.2 Recurrent Neural Networks (RNN)

These networks are designed for sequential data that must repeat the information entered, such as temporal operations or natural language processing. The data on the model is continuously fed back, allowing the information to be remembered. In this way, the information is transferred between neurons in the most up-to-date way.

3.2.3 Convolutional neural networks (CNN)

The CNN are primarily used to analyze data such as images, video or audio signals. CNN uses special layers such as input layer, convolutional layers, activation layers, pooling layers, fully connected layers and output layer. It processes data with a grid-like model. More detailed information about this network will be explained under the heading of deep learning [14,20].

3.3 Deep Learning (DL)

Deep neural networks (DNN) which the milestone of deep learning (DL), have a more complex structure than artificial neural networks and more hidden layers than ANN models. They are the deepened version of traditional artificial neural networks with neurons. Just like artificial neural networks, there are a wide variety of models for deep neural networks, such as DNNs, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTMs). Here, Deep Learning (DL) is a machine learning technique that allows a computer to be trained and perform tasks that are difficult to observe and predict in a complex structure through various programming techniques. Neural network algorithms are inspired by the human brain and its functions: like our human minds, they are designed to work

not only by following a predetermined list of rules, but also by predicting solutions and drawing conclusions based on previous iterations and experiences [21].

Deep learning, machine learning and artificial intelligence are terms with different meanings. Deep learning can be summarized as a sub-branch of machine learning, and machine learning as a sub-branch of artificial intelligence. Deep learning, unlike machine learning, generally has algorithms that eliminate some pre-processing of data. These algorithms can take and process unstructured data such as text and images. Thus, it allows systems to work automatically by minimizing human dependency and error.

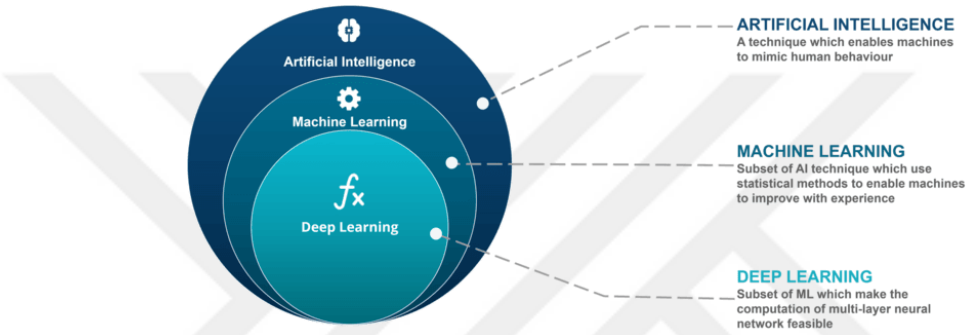


Figure 3.3 Artificial Intelligence, Learning Machine and Deep Learning [22]

With deep learning models, you can successfully train a large amount of data by applying certain operations. With these activities, you can reach accurate predictions. The techniques frequently used in this process are [23]:

- Backpropagation
- Activation functions
- Dropout and regularization
- Stochastic Gradient Descent (SGD)
- Adam optimization

Deep learning offers a broader range of work than traditional machine learning in the fields of computation and optimization by generalizing the input information. However, depending on the input examples limits deep learning. Because the deep

learning model can only make sense of what it has seen before. Any change in this input information will result in different results. Thus, the model must be retrained and passed through functions [24].

Deep learning is encountered in every field today. The main important areas of use;

- Recognition systems; Using computer vision and voice recognition systems, images and sounds are detected and defined.
- Natural Language Processing: These are the activities of detecting, analyzing and interpreting human speech ability by computers with artificial neural networks.
- Automotive: Automatic system control and driverless vehicle activities in vehicles with autonomous driving features.
- Health Research: It easily detects patients' diseases and makes predictions about possible treatments. It allows diseases that are difficult to detect, such as cancer, to be automatically detected in a short time.
- Military/Security: Deep learning provides convenience to units by detecting objects or regions through UAVs and satellites. It recognizes unknown or suspicious activities against cyber threats and performs intervention activities.
- Games and Entertainment: Access to fast recommendations is provided with digital assistant facilities. Thanks to artificial intelligence, it provides application opportunities similar to real life in sectors such as computer games and flight simulators.
- Finance: It provides strong recommendations for individuals or institutions to develop their trading strategies. It also facilitates the follow-up and analysis of activities aimed at the development of monetary management.

Deep neural networks consist of many layers that are interconnected. All of these layers are used to correctly predict and categorize the result. This computational progression through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. Data is taken from the input

layer and fed into deep learning processes with biases. It is then sent to the output layer to perform prediction and classification.

Another one in the deep learning stage is the Backpropagation process. It is used to calculate the erroneous predictions made. Thanks to its algorithms, it has the ability to move backwards on the model and change the function weights and biases. Deep learning can correct the errors on the system by using forward propagation and backpropagation processes. Thus, the algorithm becomes more accurate.

3.4 Convolutional Neural Network (ConvNet vs CNN)

Convolutional neural networks are a branch of deep learning that automatically learns and makes predictions through computer-based systems. They generally solve image classification problems through filters. This feedforward model is also used in processing many different types of data, including text and audio. Features in images, video, text and audio are detected. Thus, it performs tasks such as object detection, face and image recognition, and natural language processing. These networks use principles from linear algebra, especially matrix multiplication, to identify patterns in an image [25].

CNNs are a specific type of neural network that consists of an input layer, an output layer, and a number of hidden layers with different features. Each node on the layers is connected to the others. The hidden layers of a CNN generally consist of a series of convolutional layers based on the theory of multiplication matrices.

- Input Layer: This is the layer where the input data is entered into the system. Here, matrices are found according to the pixel values of the data belonging to images and objects. The higher the image quality (e.g. 8K (7680×4320 pixels)), the more accurate the prediction becomes. The role of CNN is to convert the features of the images into a suitable format without changing them and make a good prediction.

- Convolutional Layer: This is the layer that uses convolution operations to determine image features. It performs pattern and feature detection by separating images into grids.

  After passing through a convolutional layer, the image is converted into a feature map or an activation map. Each convolutional neuron processes the data only for its own receptive field.

  The images are convolved into a matrix called a ‘kernel’. In image processing, the kernel is a matrix used to produce different effects through the convolution operation. It is also called ‘convolution’ or ‘filter’ in some sources. The convolution of the kernel and the image produces different types of effects such as blurring, sharpening, embossing, etc. [26].

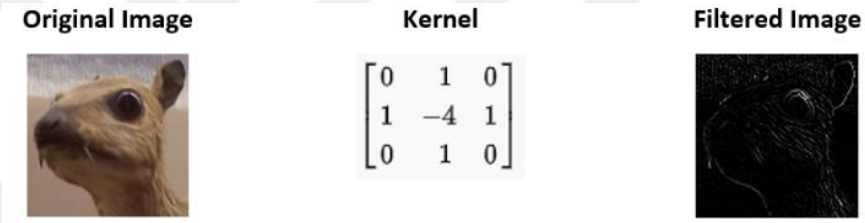
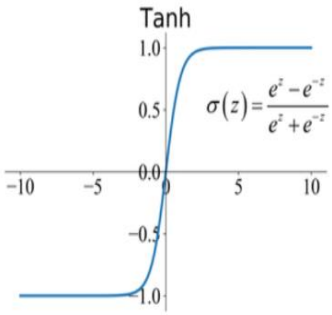
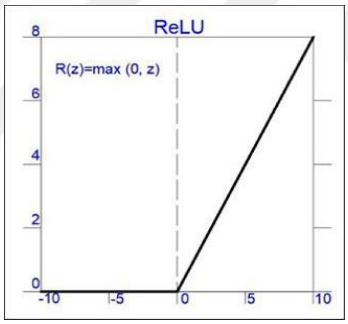
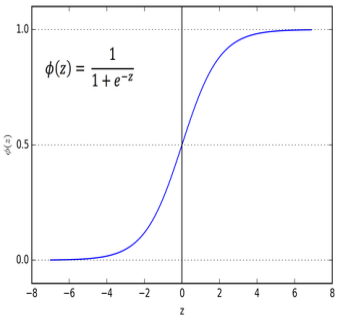


Figure 3.4 Kernel (Image Process) [26]

Activation functions subject the data received from the Convolution layer to mathematical operations. Thus, it enables learning complex relationships between input and output. Since artificial neural networks are mostly used in non-linear classifications, a non-linear function is usually selected for the activation function. Depending on the method used in these studies, activation functions such as Tanh Function, Sigmoid Function, ReLU (Rectified Linear Unit) Function are used.

Tablo 3.1 Activation functions [28]

Activation Function	Model Graph	Explanation
Tanh		Tanh function produces values between -1 and +1 for each input value. It performs better than the sigmoid function. However, it could not solve the vanishing gradient problem experienced by sigmoids. This problem was eliminated with ReLU activations.
Rectified Linear (ReLU)		In this function, when the input value is below zero, the output is zero, but if the input value is above zero, the output is equal to the input value and a linear relationship is formed with the dependent variable.
Sigmoid		The sigmoid activation function produces a value between zero and one for each element in its domain.

- Pooling Layer: To reduce the computations on the system, it reduces the dimensions of the data by merging the outputs of the neuron clusters in one layer into a single neuron in the next layer. Thus, it reduces the spatial dimensions. In addition, it helps in the selection of some fixed features that are invariant. In local pooling, small clusters with dimensions such as 2 × 2 are merged, while in global pooling, operations that will affect all neurons of the feature map are performed.

  Two types of pooling are commonly used;

  Maximum pooling is the type that gives the maximum value from the part covered by the kernel in the feature map. Average pooling gives the average of all values in each neuron cluster in the image part covered by the kernel.

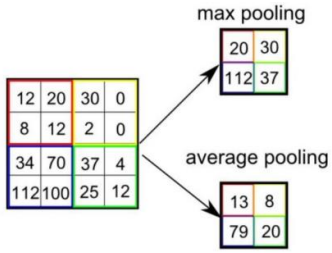


Figure 3.5 Max pooling and average pooling [27]

- Dense (Fully Connected) Layer: It simplifies the dense connection network formed after the pooling layer and turns it into a one-dimensional vector. Thus, it becomes a one-dimensional vector. This vector transforms the input data into a classified data group through a matrix. Therefore, the data is flattened and fully connected layers are needed.

- Output Layer: This is where the model makes predictions, classifies them and displays the final results.

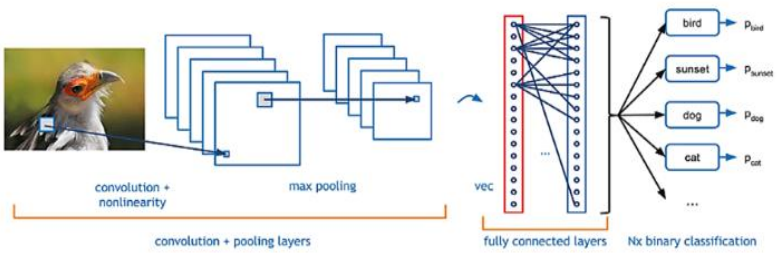


Figure 3.6 The CNN architecture [26]

CHAPTER 4

OBJECT DETECTION METHODS AND EXPERIMENT

4.1 Object Detection

Object detection is a computer vision task used to detect visual objects of certain classes (e.g., people, animals, cars, or buildings) in digital images such as photos or videos. The aim of object detection is to develop computational models by performing computer vision applications. Through these models, it is to determine the location and boundaries of objects in an image and classify objects into different categories.

Advanced algorithms powered by deep learning can now detect objects with high accuracy even in complex environments or adverse conditions such as low light. Object detection is one of the core topics in computer vision. For example, it forms the basis of many other downstream computer vision tasks, such as instance and image segmentation, image captioning, object tracking, and more. Specific object detection applications include pedestrian detection, animal detection, vehicle detection, human counting, face detection, text detection, pose estimation, or license plate recognition [30].

In recent years, with the rapid development of deep learning methods, significant changes have occurred in object detection technology. Especially with the improvement of the quality of software and physical conditions, high performance increases have occurred in object detection and classification.

Today, object recognition with deep learning is the basis of vision-based artificial intelligence software and programs. The main areas of use for object detection are;

- Robotics and Autonomous Vehicles
- Surveillance and Security
- Healthcare and Medical Imaging

- Retail and E-commerce
- Military

There are some known constants for making accurate and fast estimations of the object detection sequence. Their metric values facilitate the analysis. The two most commonly used computational constants are the Intersection Over Union (IoU) and Average Precision (AP) metrics.

- Intersection Over Union (IoU): IoU helps in fine-tuning the object localization accuracy and calculating the errors. The calculation is done by finding the ratio of the intersection area set of bounding boxes on the object to the total union area set. This gives a good estimate of how close the estimated bounding box is to the original bounding box. IoU values range from 0 to 1, with higher values indicating better localization accuracy. An IoU of 1.0 indicates perfect alignment. A higher IoU value indicates a better match between the predicted and ground-truth bounding boxes.

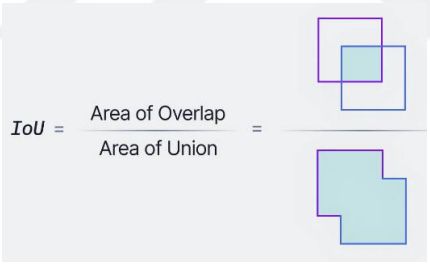


Figure 4.1 IoU [32]

- Average Precision (AP): AP is a single value that summarizes the precision and recall performance of the model by calculating the area under the precision-recall curve to make a prediction. Recall calculates the proportion of true positives among all true positives and measures the ability of the model to detect all instances of a class. Precision expresses the proportion of true positives out of the total predictions made by the model.

  The area under this precision-recall curve gives the Average Precision per class for the model. The average of this value taken over all classes is called Mean Average

Precision (mAP). mAP extends the concept of AP by calculating the average AP values across multiple object classes. This increases the performance of the model and facilitates object detection. In object detection, precision and recall are not used for class predictions. Instead, they serve as predictions of the boundary boxes to measure decision performance. An IoU value greater than 0.5 is taken as a positive prediction, while an IoU value less than 0.5 is a negative prediction. A higher mAP/AP value indicates a better performance and accuracy of our prediction of the model [32].

There are also several metrics that evaluate the accuracy too. For example, True Positive Rate (TPR), False Positive Rate (FPR), F1-score, and Log Average Miss Rate (MR). In addition to these metrics, object detection models can also be evaluated based on their computational efficiency.

There are two main methods in object detection technology: single-stage methods and two-stage methods:

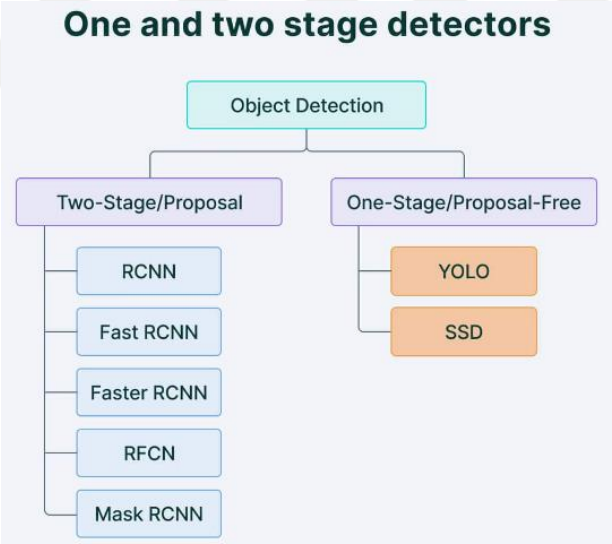


Figure 4.2 The most common object detection algorithms [32]

Single-shot object detection algorithms offer the advantage of computational efficiency by making predictions in a single pass of the input image. This makes them suitable for real-time applications and resource-constrained environments. However, single-shot detection methods may have limitations in accurately detecting small

objects and may be less accurate overall compared to two-shot detection methods. Single-stage methods prioritize inference speed, and example models include YOLO, SSD, and RetinaNet [31].

Two-shot object detection methods involve two passes of the input image; the first pass generates object proposals and the second pass refines them. Although they offer higher accuracy, they are computationally more expensive and may not be suitable for real-time applications. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN, and Cascade R-CNN [31].

4.1.1 Region-Based Convolutional Neural Networks (R-CNN)

Region-based convolutional neural networks (R-CNN) first divide the given input image into approximately two thousand regions (for example, anchor boxes are a type of selection method). Then, a CNN is applied to each region in turn. The categories and bounding boxes of the images to be trained are assigned to predefined labels. Then, CNN calculations are used to make predictions from each proposed area. A lot of time is needed to separate the regions into correct classes and calculate their sizes. Therefore, the training time is longer than YOLO.

In 2015, Fast R-CNN was developed to reduce the training time. While the original R-CNN divides the image into about two thousand regions of interest and calculates them, Fast R-CNN runs the neural network once on the entire image. Fast R-CNN is similar to YOLO in terms of architecture. However, YOLO is preferred more because its code is simpler and more plain [30].

4.1.2 RetinaNet

RetinaNet is a single-stage object detection model designed to address the class imbalance problem that often affects detection tasks of hard-to-detect objects, operating in a similar manner to the basic Faster R-CNN principles. RetinaNET can process images at multiple scales simultaneously. It addresses the class imbalance problem by using a focal loss function. It is preferred for processing images that are hard to classify, such as objects that are small in size or partially obscured [35].

4.1.3 Single Shot Detectors (SSD)

SSD is an object detection model that can predict multiple classes simultaneously using a single deep neural network. The method divides images into bounding box areas to capture multi-scale information. This way, it produces a trace matrix with various aspect ratio sizes. It predicts multiple bounding boxes per grid cell and assigns class probabilities to each box. It achieves efficient performance on smaller input image sizes in a single step by fluently using convolutional layers [30].

SSD detector is easy to train and integrate into software systems when performing object detection. The image object detector generates scores for the presence of each object category in each default box and adjusts the box to better match the object shape. In addition, the network processes objects of different sizes by combining predictions from multiple feature maps with different resolutions [34].

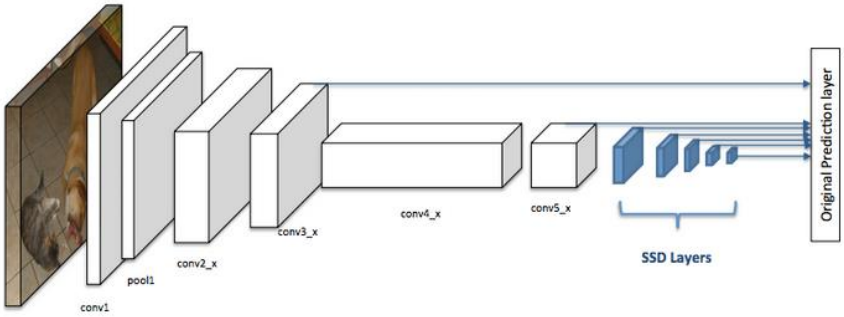


Figure 4.3 Architecture of CNN with a SSD Detector [33]

The method is called “single shot” because it detects objects in a single pass of the CNN without any region proposals or processing steps. SSD is preferred for real-time applications such as video surveillance or autonomous vehicles due to its high speed and efficiency.

4.1.4 YOLO (You Only Look Once)

YOLO is a deep learning algorithm that simply uses convolutional neural networks to detect objects. YOLO is used in our study because it is the most common and easy-to-use object recognition algorithm. It is used in the field of computer vision (CV) and

performs object detection by separating the object or objects in the image. YOLO uses a fully convolutional neural network for real-time object detection. CNN, a forward-looking neural network, was inspired by the visual cortex of animals. YOLO was first introduced in 2015 with Joseph Redmon's article "You Only Look Once: Unified, Real-Time Object Detection."

Unlike R-CNN or similar traditional object detection methods, YOLO performs operations on a given image only once. In this way, it detects and classifies objects in an image in a single step, and provides faster and more successful results than object detection algorithms, without the need to process images in multiple steps.

R-CNN and similar algorithms first find possible locations where objects may be in given images. These are usually based on region proposals and require analyzing an image multiple times. Then, it looks at which of these regions contains the object and classifies the objects.

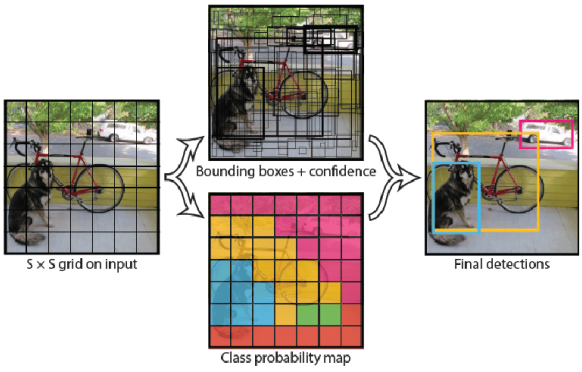


Figure 4.4 YOLO grid system [29]

The working principle of YOLO;

The image used as input data is divided into SxS grids. Each box in the form of bounding boxes is queried to see if there is a target object. Whether there is an object in the grid is determined according to the confidence score. If the confidence score is 1, there is an object in the grid, if it is 0, there is no object in the grid. Thus, each grid creates its own prediction vector.

$$Confidence\ Score = Pr(obj) * IoU \quad (4.1)$$

- Pr(obj): probability of finding the object in the grid
- IoU: Intersection of the predicted box with the box where the object is actually located

The YOLO algorithm enters the entire image into the convolutional neural network system once without making a prior location prediction. After this process, the feature map is generated.

Multiple bounding box predictions are made for the target object in each grid cell. For this reason, it can estimate that more than one grid object is inside itself. In this case, too many unnecessary bounding boxes are formed on the screen. If we consider that each bounding box has a confidence score, this situation will prevent us from making correct predictions. To prevent this situation, the Non-Maximum Suppression algorithm is used.

Figure 4.5 Non-maximum suppression [36]

The Non-Maximum Suppression algorithm draws the highest confidence value of the bounding boxes drawn for the objects detected in the image on the screen. It filters the boxes according to the confidence score through the Non-Maximum Suppression (NMS) algorithm, that is, if our target object is detected in 2 different bounding boxes for a grid cell. It filters the box with the higher score and adds the bounding box with the lower score. Thus, the overlap problem is prevented and the results with the highest

score are obtained. The final output is a set of predicted bounding boxes and class labels for each object in the image.

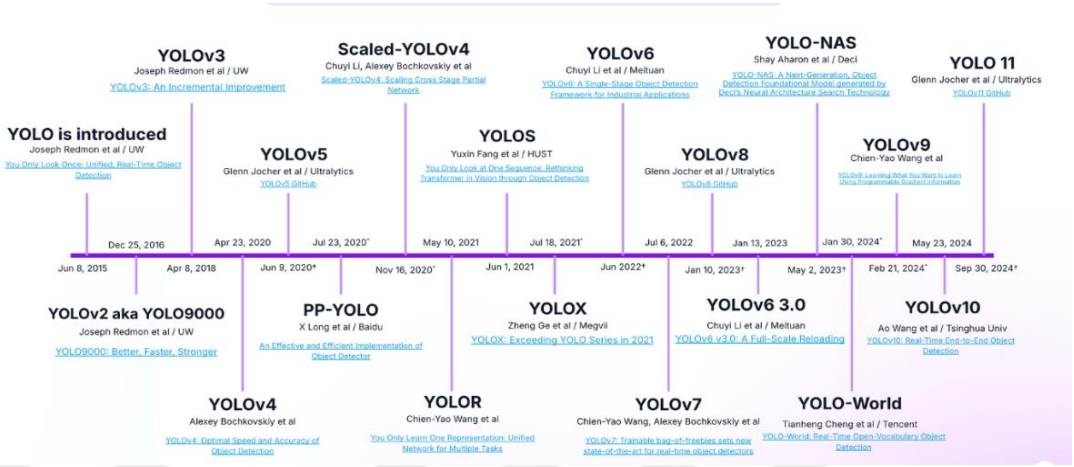


Figure 4.6 YOLO versions [37]

The YOLO (You Only Look Once) algorithm was first developed in 2016 by Joseph Redmon and some friends who were doing artificial intelligence research and published with an article. YOLO, which was one of the most important developments in the field of object detection in those days, has been developed in different versions over time. These versions aim to make object detection faster and more accurate. The main versions of YOLO and the features of the versions will be briefly mentioned below;

- YOLOv1 (2016), the first version of the YOLO algorithm, simplifies object detection by performing it in a single step. It divides the image into a grid and evaluates each grid cell to determine if it contains one or more objects. While YOLOv1 was significantly faster than other object detection algorithms, it struggled with accuracy when detecting small objects and was capable of recognizing only a limited number of objects.

- YOLOv2 (2016) also known as “YOLO9000” brings significant improvements over YOLOv1. This version uses higher-resolution inputs to better detect smaller objects. Additionally, it incorporates Anchor Boxes to enhance the precision of

object localization. YOLOv2 achieved substantial advancements in both speed and accuracy and is capable of detecting approximately 9,000 different classes.

- YOLOv3 (2018) is a model designed to improve object detection across different sizes by providing multi-scale predictions. It processes an image at three different scales and makes object predictions at each scale, enabling more accurate detection of both small and large objects. YOLOv3 introduces a new architecture called Darknet-53, offering a deeper and more powerful model. However, it is slightly slower compared to its predecessors.

- YOLOv4 (2020) aims to enhance the performance of its predecessors by achieving a better balance between speed and accuracy. It offers an efficient and scalable model that delivers high performance across various applications. This version improves model performance through techniques known as the "bag of freebies" (BoF) and "bag of specials" (BoS). Additionally, YOLOv4 employs the CSPDarknet53 architecture, achieving higher accuracy rates.

- YOLOv5 (2020), although not an official version of the YOLO family, has gained significant popularity due to its development by a large community. Built on Python and PyTorch, YOLOv5 offers a more user-friendly and flexible framework compared to its predecessors. One of its key strengths is its ease of use during the training process, making it accessible even to less experienced developers. YOLOv5 introduces a variety of model sizes nano (N), small (S), medium (M), large (L), and extra-large (XL) allowing users to choose the most suitable configuration for their specific applications, balancing speed, accuracy, and resource requirements. This flexibility makes it an attractive choice for a wide range of use cases, from lightweight applications on edge devices to more demanding scenarios requiring high accuracy.

  Furthermore, YOLOv5 includes various built-in features like data augmentation techniques, optimized anchor box generation, and pre-trained weights, streamlining the deployment process and improving the overall efficiency of object detection projects. Its open-source nature and active community support further contribute to its rapid adoption and continuous evolution.

- YOLOv6 (2022), while not an official version of the YOLO series, introduces advanced optimizations tailored for modern applications. This version focuses on delivering a smaller and more efficient model, making it particularly suitable for industrial and resource-constrained scenarios. YOLOv6 employs a variety of optimization techniques to achieve high performance in both speed and accuracy. These optimizations include improved model architecture, enhanced training strategies, and efficient computational methods, all designed to reduce latency and resource usage without compromising detection quality.

  The design of YOLOv6 prioritizes practical deployment in real-world environments, such as embedded systems, IoT devices, and industrial automation, where computational efficiency is critical. Its compact structure allows it to operate effectively on hardware with limited processing power, enabling faster inference times and lower energy consumption. Additionally, YOLOv6 benefits from a community-driven approach, leveraging insights from both research and practical deployment experiences. These improvements make YOLOv6 a strong contender for applications that require reliable and efficient object detection while addressing the demands of scalability and real-time processing.

- YOLOv7 (2022) is one of the most significant releases in the YOLO series, offering enhanced object detection capabilities with both higher speed and accuracy. Its optimized architecture makes it particularly well-suited for real-time applications, where rapid and precise performance is critical. YOLOv7 introduces several innovations to improve efficiency and effectiveness in complex object detection tasks. The model is designed to handle more intricate scenarios while maintaining a lightweight structure, ensuring compatibility with a wide range of hardware, including edge devices and mobile platforms. This efficiency does not come at the cost of performance; YOLOv7 achieves state-of-the-art results in benchmarks, outperforming many predecessors and competitors.

  One of the key strengths of YOLOv7 is its ability to process diverse and challenging datasets, delivering reliable results even in environments with varying object sizes, occlusions, and lighting conditions. Its advancements make it an ideal choice for

applications such as autonomous vehicles, surveillance, robotics, and other real-time systems. By refining the balance between computational cost and detection accuracy, YOLOv7 represents a significant leap forward in the YOLO series, solidifying its reputation as a go-to solution for cutting-edge object detection.

- YOLOv8 (2023) is the most used version of the YOLO series, integrating all the advancements from previous versions while pushing the boundaries of both accuracy and speed. This version has been optimized for the highest performance, making it ideal for a wide range of applications, from industrial use cases to cutting-edge research projects. One of the standout features of YOLOv8 is its enhanced flexibility. The model offers various sizes, allowing users to choose the best configuration based on their specific needs, whether they require lightweight models for edge devices or larger models for high-accuracy tasks. This versatility enables YOLOv8 to easily adapt to different deployment environments, such as real-time object detection in autonomous vehicles, security surveillance systems, robotics, and more.

  YOLOv8 builds on previous optimizations, such as better handling of small objects, improved anchor box generation, and more efficient use of computational resources, while maintaining a robust architecture. As a result, it delivers faster inference times without sacrificing accuracy, making it suitable for both real-time applications and scenarios requiring complex detection tasks. In terms of usability, YOLOv8 is also designed with a user-friendly interface, offering ease of integration and customization, which is particularly valuable for developers working on custom object detection solutions. Its high flexibility, combined with strong community support, makes YOLOv8 a popular choice for both commercial applications and experimental research in the field of computer vision.

- YOLOv9, introduced in February 2024, represents the latest evolution in the YOLO (You Only Look Once) series of object detection models. This version introduces two groundbreaking features: the Programmable Gradient Information (PGI) framework and the Generalized Efficient Layer Aggregation Network (GELAN).

The PGI framework addresses a critical challenge in deep neural networks—information bottlenecks by ensuring the availability of reliable gradient information during training. It also supports deep supervision mechanisms for lightweight architectures, enabling both compact and deep models to achieve significant improvements in accuracy. By mitigating gradient-related limitations, PGI enhances the model's ability to learn effectively and make precise predictions.

The GELAN architecture is designed to optimize object detection tasks through a highly efficient and lightweight approach. It excels across various computational blocks and depth configurations, making it versatile for deployment on a wide range of devices, including those with limited computational resources, such as edge devices. By integrating PGI and GELAN, YOLOv9 sets a new benchmark in lightweight object detection, outperforming its predecessor, YOLOv8, in key areas. Specifically, it reduces model parameters and computational overhead while improving Average Precision (AP) by 0.6% on the MS COCO dataset. Despite being a recent release, YOLOv9 demonstrates impressive competitiveness, solidifying its role as a significant advancement in the field.

- YOLOv10, a next-generation object detection model introduced in 2024, offers significant improvements in speed and accuracy for real-time applications. This model focuses on areas where fast and accurate decisions are critical, such as autonomous vehicles, agricultural technologies, and surveillance in low-power devices. Unlike previous models, YOLOv10 works without the need for NMS. Although traditional NMS increased accuracy by eliminating overestimations, it slowed down the processing speed.

  YOLOv10 solves this process faster and more efficiently with the double-labeling method. YOLOv10 offers a faster and smaller model compared to YOLOv9, without losing accuracy. It shows higher performance in tests on the COCO dataset.

Table 4.1 YOLO variant comparison [38]

Version	Date	Contributions	Framework
V1	2015	One-shot object detector	Darknet
V2	2016	Multi-scale training, dimensional clustering	Darknet
V3	2018	SPP block, Darknet-53	Darknet
V4	2020	Mish-based activation, CSPDarknet-53 backbone	Darknet
V5	2020	Anchor-free detection, SWISH-based activation, PANet	PyTorch
V6	2022	Self-attention, anchor-free object detection	PyTorch
V7	2022	Transformers, E-ELAN reparameterization	PyTorch
V8	2023	GANs, anchor-free detections	PyTorch
V9	2024	Programmable Gradient Information (PGI), Generalized Efficient Layer Aggregation Network (GELAN)	PyTorch
V10	2024	NMS-free training approach, dual label assignments, holistic model design for enhanced accuracy and efficiency	PyTorch

4.2 Dataset

An object detection dataset is a specialized dataset designed for training, testing, and validating object detection models. These datasets include images or videos annotated with information about the objects they contain, typically in the form of bounding boxes and class labels. The components of an object detection dataset:

- Images or Videos:

  The primary data, which may contain multiple objects of different types in various settings.

- Annotations:
  - Bounding Boxes: Rectangular coordinates defining the region of interest where an object is located.
  - Class Labels: The category or type of the object within the bounding box (e.g., "car," "dog," "tree").
  - Additional Metadata (optional):
    - Object confidence scores.

  - Object attributes (e.g., color, orientation).
  - Segmentation masks for pixel-level object outlines (common in instance segmentation datasets).

- Formats:

  Object detection datasets are typically stored in specific formats that include both the image data and annotation data:

  - COCO (Common Objects in Context): JSON files with bounding box coordinates, categories, and segmentation masks.
  - Pascal VOC: XML files with bounding box coordinates and class labels.
  - YOLO (You Only Look Once): Text files containing normalized bounding box coordinates and class labels.

The most commonly used datasets in object detection studies are as follows;

- COCO (Common Objects in Context): Contains over 200,000 labeled images with more than 80 object categories. Widely used for object detection, segmentation, and image captioning tasks.
- Pascal VOC: A foundational dataset for object detection and segmentation, containing images with 20 object categories.
- Open Images Dataset: A large-scale dataset with millions of images and over 600 object categories, including detailed annotations.
- KITTI Dataset: Designed for autonomous driving research, it includes images with annotations for cars, pedestrians, and cyclists.
- ImageNet (Object Detection Challenge): A dataset with millions of annotated images across thousands of categories, used in large-scale detection challenges.
- DOTA (Dataset for Object Detection in Aerial Images): Specially curated for aerial image object detection tasks.
- MNIST (for machine learning): A dataset of handwritten digits.
- Kaggle Dataset: A wide collection of datasets on various topics.

- Titanic Dataset: Contains survival data of passengers from the Titanic disaster and is commonly used for machine learning practice.

The application areas where datasets are most commonly used today are as follows.

- Autonomous Vehicles: Detecting pedestrians, vehicles, and traffic signs.
- Surveillance Systems: Identifying suspicious objects or activities.
- Retail Analytics: Recognizing products and tracking inventory.
- Healthcare: Detecting abnormalities in medical images like X-rays or MRIs.
- Robotics: Enabling robots to interact with their environment.

When well-prepared, cleaned, and labeled, datasets play a crucial role in the success of data science and machine learning projects.

4.3 Test Phases And Platform

4.3.1 Test Phases

The YOLO algorithm first divides the entire image into a grid of size S×S. Each grid cell is processed through a neural network. The goal is to detect the object located within the grid cell and encapsulate it in a bounding box. While passing the grid cells through the neural network, the algorithm checks if the center point of an object lies within a specific grid cell. If the object's center point is found within the grid, the algorithm calculates the height, width, class, and confidence score of the detected object.

To use the YOLO algorithm for object detection on images or videos, the model must be trained with a relevant dataset. Here are the steps I followed for training the model:

4.3.1.1 Dataset Creation

The main source and most time-consuming node of our work is datasets. The dataset must have the correct labeling of each object to be detected. Since we will perform the analysis on images, we need to prepare datasets suitable for the system. Datasets are created by bringing together a large number of images. While we use a large portion of these images for training, we use a small portion for validation or testing. Images

or videos containing the objects you want to detect can usually be found on the internet as ready-made packages. These datasets are created by selecting one or more object classes. The issue you need to pay attention to in photos is that the image dimensions are larger than 416 x 416. You can also use smaller-scale photos for training, but this will affect the quality of the training. Therefore, it causes the object detection value to be low. In order to provide effective training, these examples must accurately represent the target scenarios.

Figure 4.7 The example pictures of Dataset [40]

In the thesis study, the aircraft category was used by focusing on single object detection. In the aircraft category, a total of 295 images, most of which were taken from UAVs with high resolution and different pixels and sizes, were used. 171 of these images were used for training, 82 for validation and 42 for testing. The efficiency level of the system can be increased by increasing these numbers. In addition, training can be done with moving images by preparing the necessary files instead of images. Objects in the dataset are labeled with horizontal bounding boxes. The datasets used were taken from the most popular dataset websites called Kaggle and Roboflow. The size of the dataset taken from Kaggle is approximately 10.2 GB and consists of 10600

images. A working dataset containing images taken from UAV was prepared manually from these images. The dataset was trained in the system to have ideal image dimensions of 640 pixels height and 640 pixels width [39,40].

4.3.1.2 Dataset Labeling

Labeling is done to make the objects that will be detected on the collected data suitable for the model by marking them in each image or video frame. This process usually involves creating bounding boxes around the objects and assigning object class labels. I used the Makesense.ai web site for labeling in my study. I preferred it because it is both open source and provides output compatible with multiple labeling formats, especially YOLO. There is no fixed and standard format for labeling. This varies depending on the deep learning model you will use. The most commonly used formats are: COCO (.json), Pascal VOC (.xml) and YOLO (.txt), but in my study, .txt, which is a labeling model compatible with YOLO, was used [41].

```
0 0.475089 0.516904 0.756821 0.681495
1 0.517794 0.528470 0.889680 0.729537
```

Figure 4.8 The example label of Dataset [41]

The general YOLO layout includes the identification number (object class), coordinates (x-y) and distances and width values of each object. Separate lines are created for each object. Since we want to recognize only the Airplanes in the image in our study, the first value presented as “object class” is seen as “0”. If more than one object training is desired, different label categories are created for each object. In these categories, values would be written by creating lines with increasing numbers starting with “1” in the .txt file for each object.

4.3.1.3 Darknet Organization

Darknet is an open source structure written in C and CUDA programming languages used to train neural networks. Darknet environment is set up and model configuration is shown by preparing configuration files, class names and paths to labeled data. It is

preferred because it is easy to install, can be processed with CPU and GPU and is fast. Source files can be accessed from GitHub. Darknet code structure of the version used in my study was taken from GitHub. These files are constantly updated by experienced software developers to ensure more secure and effective use of the system. At this stage, a data.yaml file is created to train the weights of the relevant codes. This file is a notebook file that is used to detect and classify the objects trained by the model. This data can be transferred to the YOLO files in the training and changes can be made during the learning stage. In models up to YOLOv3, the Darknet library is downloaded and installed from the site via the GoogleColab application. In versions after YOLOv5, the PyTorch library developed by Facebook's AI Research (FAIR) laboratory is used. The PyTorch library is an open source machine learning library used for applications such as computer vision and natural language processing. Thus, it can be used in a cloud computing environment without having to rewrite all system requirements. The programming language used throughout the study is Python, which is the most suitable language for object recognition [42].

4.3.1.4 Model Training

The prepared dataset, Darknet or PyTorch configuration, is used to train the model using a virtual computer. In my study, I used GoogleColab, a free application offered by Google to everyone, for the experiments conducted with YOLOv3. In this system, I performed the necessary code experiments for the model with the Python programming language using the Darknet library. In YOLOv5 and later versions, open-source PyTorch library data was used on a computer with a high operating system and memory based on the Ubuntu operating system. This library is highly preferred for developing deep learning models with its flexible structure and ease of use. The images and configuration files required for YOLO training, the weights required for activation functions and other configuration data can be loaded into the system as compressed files or virtual packages. This data includes parameters such as learning rate, batch size and number of iterations. These are the values that increase the accuracy and efficiency for our training. It takes a long time to train the loaded data by writing the necessary codes. The more data is desired to be trained in the system,

the longer the training period. After all this training, the model learns to predict object classes, bounding box coordinates, and confidence scores.

4.3.1.5 Validation and Testing

After the completion of the photographs trained for object detection, data on the performance of the system is produced. The values of the input data and output data of the model can be seen. The success of the training is revealed. In particular, the higher the mAP value, the higher the object detection rate. The mAP value is calculated using only the IoU (Intersection over Union) threshold value of 0.5. In other words, for a prediction to be accepted as correct, the predicted and real bounding boxes must overlap by at least 50%. Therefore, in cases where a lower accuracy standard is sufficient (e.g., for rapid prototyping or simpler applications), this value is used to evaluate the basic performance of the model. However, in projects requiring higher accuracy (e.g., autonomous vehicles, medical imaging), the mAP@0.5:0.95 value is used to comprehensively evaluate the overall performance of the model. This is the critical value that will be referenced in the study. The mAP@0.5:0.95 value is calculated at different thresholds from IoU threshold values 0.5 to 0.95 (usually at 0.05 intervals) and these values are averaged. This metric more comprehensively measures how well the model performs at different sensitivity levels. In academic studies or competitions, if there is no special requirement, mAP@0.5:0.95 is usually preferred because it is a more comprehensive metric. However, the choice can be made according to your project requirements.

Table 4.2 Number of images used in the thesis

Training	171 images
Validation	82 images
Test	42 images

Transfer learning is the process of training a machine learning model for one task and then adapting that model to another task. In deep learning in particular, transfer learning is used to train a new model faster and with less data by using the knowledge and capabilities of a previously trained model. Transfer learning is a powerful method for creating an effective model with less data and resources. It is particularly effective

when used on small datasets or in a new application area. There are several approaches to transfer learning. These are Feature Extraction, Fine-Tuning, and Frozen Model. In our YOLO models, transfer learning involves freezing some or all layers of the previously trained model, then unfreezing and maintaining it on the new dataset during the training process. This allows the model to adapt its learned features to the features of the new task while preserving the general knowledge from the original task.

If an error occurs during or after training, the training is restarted by making adjustments to the training start command or data. After the model is completed, information about the result data, Excel tables about the output values and statistical graphics about the result analyses can be obtained. These are also transferred from the system to the computer via codes. The data about the predictions are not definite and generalizations are made. Therefore, in order to ensure closeness to reality, correct datasets, quality images and high-labeled training models must be made.

4.3.2 Test Platform

In the testing phase of our study, it was planned to use Google Colab, which Google has been offering for free use since 2017. Colab is a free online cloud-based Jupyter Notebook environment that allows you to train your machine learning and deep learning models on CPUs, GPUs and TPUs. Thanks to this system, you can create CPU environments that are not possible at hand and use them for your high-data-requiring operations. Data for Colab is taken from a cloud space and processed. After the study, the data is loaded to the drive again through codes. In the training process planned for the thesis, it was initially planned to use Google Colab for the use of all YOLO versions. During the training, Python 3.10.12 version was used while configuring with Tesla T4 GPU processor on Google Colab platform. Colab system memory space allowed approximately 15GB. In YOLOv3 training, 2/3 (10GB) of the memory was used and it was observed that the system was overloaded and slowed down and overheated the hardware. In order to obtain more accurate results in line with the purpose of the thesis study, the latest version and the most capable YOLO versions were selected for the algorithms. The YOLOv5s, YOLOv5x, YOLOv8x, YOLOv10x versions used in the continuation of the training required larger memory and a powerful system. For this reason, Python 3.8.10 version was installed in a

computer environment with Ubuntu operating system and PyTorch 2.4.1 version was used as the algorithm. This platform environment with high data processing speed and memory helped the training to be faster and error-free.

In order to get accurate predictions and fast results in the tests, reference values were selected as image size 640 pixels, batch size 10, and iteration called epoch 50 repetitions.

- Image Size: YOLO models process images of fixed sizes. This fixed size is necessary to increase the consistency and performance of the training and prediction processes. The model expects the data to be processed in this size during training. In our model, the image size is selected as 640x640. If your images are not 640x640, YOLO automatically resizes them. A smaller size (for example 320x320) speeds up training but may reduce detection accuracy. A larger size (for example 1280x1280) may increase accuracy but requires more GPU memory and slows down training.

- Batch size: Determines how many images the model will train with at a time. Batch size is set according to the hardware. It depends on your GPU memory. A large batch size provides faster training but may not have enough memory.

- Epoch: The model processes the entire training dataset once from beginning to end. It specifies how many times the model will process the entire dataset during training. For example, if you train with 50 epochs, the model will process your entire dataset 50 times. A sufficient number of epochs is necessary for the model to learn, but too many epochs can lead to overfitting.

In order to understand which algorithm is the most suitable, different values were changed in different versions and these analyzes were compared. In the experiments, the initial learning rate, which depends on the update rate of the model weights, was started with the most optimum value of 0.01 (lr0) and increased proportionally according to the size of the epoch number, and the final learning rate (lrf) was run in the system as 0.1.

In training, different optimization algorithms (SGD, Adam etc.) were used and the Warmup Epoch parameter was selected as 3 by YOLO. This parameter can be useful for reducing the error in optimization instabilities and increasing the learning rate slowly and stably in more complex and less stable models. YOLO estimates the center coordinates (x, y) of the objects and creates bounding boxes by adding width and height values to these coordinates. These boxes enable the model to learn the target objects more effectively and efficiently. Therefore, the ideal bounding box value of 0.5 was used as another constant for the system.

In this study, a threshold value of 0.3 was determined for the probability that the predicted object belongs to a specified class. This threshold helps to ignore low-probability predictions and obtain more reliable results. In addition, the default value of 1.0 was selected for the class prediction weight (cls_pw) and thus a balanced evaluation of the class predictions was provided. The IoU (Union Over Intersection) threshold was set to 0.2. All experiments were performed using the Python programming language to ensure more efficient and mass loading of data, and the worker parameter was set to 8, which allowed the process to be run in parallel on 8 branches at the same time. This configuration contributed to the reduction of processing times. The strategies implemented in this study aimed to optimize both accuracy and performance speed.

CHAPTER 5

TEST RESULTS

In our study, object detection is aimed using YOLO deep learning model. For this purpose, trainings were performed with the same values in different versions to make a general evaluation of the training. During these trainings, the detection estimates of the object were compared by making changes on the parameters in the same version. In the trainings, the training time varied according to the efficiency of the object in the versions. Training times varied according to the size of the dataset and the quality of the images in its content, as well as between the versions. Training times lasted at least 23 minutes and at most 9 hours. While the shortest training time was in the training of 50 validation images in YOLOv3, the longest training time was when the 750 image pixel value was selected in the training with YOLOv10x.

In YOLO, the main metrics used to evaluate the accuracy, precision and overall success of the model are metrics/precision, metrics/recall, mAP (Mean Average Precision) and mAP@0.5:0.95. Each of them measures different aspects of the model.

- metrics/precision: Shows the accuracy of the model's predictions [43].
- metrics/recall: Shows how well the model detects objects in total [43].
- mAP@0.5: Measures the detection accuracy across all classes when the model's IoU threshold is 0.5 [43]. It is the reference value for object detection belonging to a single class.
- mAP@0.5:0.95: A more comprehensive metric that evaluates the model's performance at different IoU thresholds [43].

In particular, mAP@0.5:0.95 is considered the most comprehensive and standard metric for modern object detection competitions. In our study, the mAP@0.5 value is the metric that should be taken into consideration more due to the limited number of images and the distribution of objects in a single class.

5.1 YOLOv3 Tests and Analysis

In the first stage of our test, training was done with YOLOv3. The object label for the airplanes that were to be detected was given the "aircraft" label. As we mentioned in previous topics, the parameter to be followed is the "mAP@0.5" value. The approach of this value to "1" is an indication that the object to be detected in the images given for validation has been detected. In other words, the larger the mAP@0.5 value, the more objects were detected. After training 171 images on YOLOv3, values were taken for validation of 82 images. Then, the performance of the model was tested on 42 images. In all trainings, the image size was selected as 640 pixels, the batch size as 10, and the epoch value as 50 as reference values.

Table 5.1 Result analysis of using YOLOv3 82 validation images

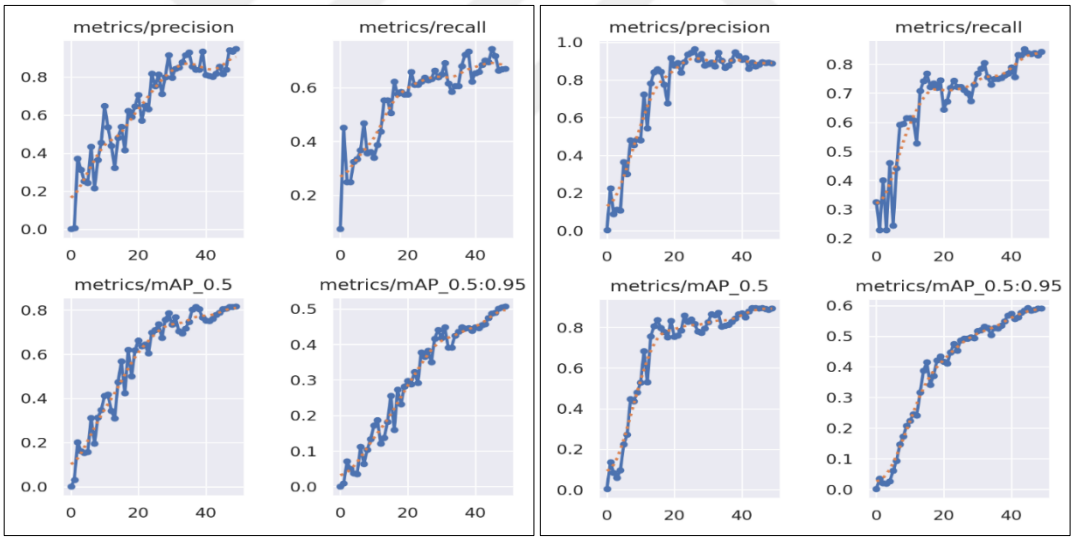
epoch	metrics/ precision	metrics/ recall	metrics/ mAP_0.5	metrics/ mAP_0.5:0.95
1	0.0035366	0.32584	0.0038379	0.0011517
2	0.22669	0.22846	0.13528	0.034796
3	0.088604	0.40075	0.083186	0.020824
4	0.11329	0.22846	0.059395	0.018423
5	0.10883	0.46067	0.097534	0.026329
⋮	⋮	⋮	⋮	⋮
46	0.88178	0.83806	0.89541	0.59224
47	0.8956	0.83537	0.89674	0.58404
48	0.89246	0.83919	0.89343	0.58638
49	0.89307	0.83146	0.8889	0.59049
50	0.88941	0.8427	0.89403	0.59017

Table 5.2 Result analysis of using YOLOv3 50 validation images

epoch	metrics/ precision	metrics/ recall	metrics/ mAP_0.5	metrics/ mAP_0.5:0.95
1	0.0011111	0.074468	0.00066541	0.0001533
2	0.0057776	0.45213	0.031566	0.0092104
3	0.37244	0.25	0.20103	0.071181
4	0.31327	0.25	0.16169	0.049718
5	0.25171	0.32447	0.15281	0.037485
⋮	⋮	⋮	⋮	⋮

46	0.8163	0.74468	0.80452	0.48667
47	0.83947	0.71809	0.80742	0.49012
48	0.94045	0.66489	0.81427	0.50091
49	0.93323	0.66916	0.81519	0.50462
50	0.94794	0.67021	0.81666	0.50715

The result tables for training different numbers of images for object detection with YOLOv3 are shown above. When the analyses are examined, the rate of correctly detected objects in the model (precision) increased as the number of validation images (Table 5.1) decreased. On the other hand, when we look at how many of all real objects were detected (recall), the higher the number of images (Table 5.2) to be verified, the higher the recall value. This is also similar to mAP@0.5, a metric that measures the overall object detection performance of the model. This means that the higher the number of images for training and validation, the better the object detection performance.



(a) (b)

Figure 5.1 Result charts of using YOLOv3 validation images 50 (a) and 82 (b)

When the above graphs (Figure 5.1) are examined, it is seen that the number of trained images exhibits differently structured graphs with similar parameters. When the number of images decreases, a sharper line graph emerges, while when the number of verified images increases, an oval and smooth graph is exhibited.

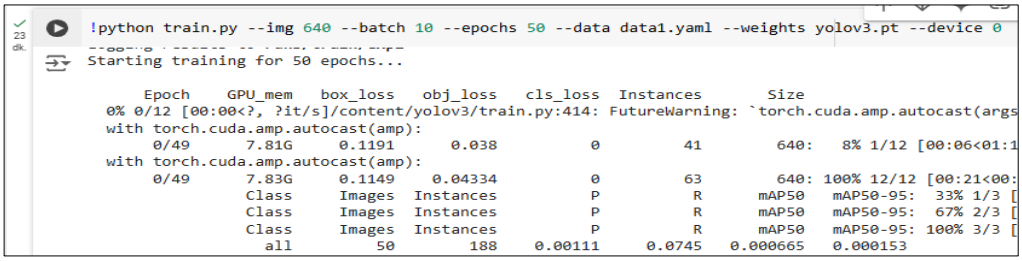
(a) (b)

Figure 5.2 Object detection of using YOLOv3 validation images 50 (a) and 82 (b)

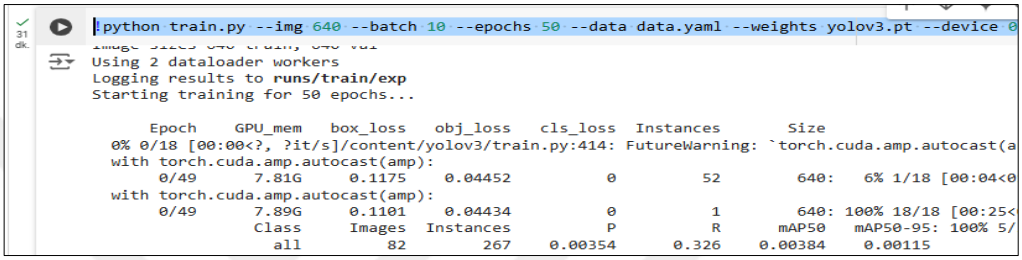
As seen in the images (Figure 5.2), similar to the analysis tables, the performances in detecting objects in the same images were similar. For example, both the 50 (a) image validation training and the 82 (b) image validation training could not detect the airplane in the images shown with a green square. However, there were differences in both trainings. For example, while the 50 (a) image validation training could detect the airplane in the images shown with a blue square, the 82 (b) image validation training could not detect it. In another image shown with a yellow square, the 50 (a) image validation training gave an incorrect result by perceiving the shadow of the airplane as an object even though there was no airplane in the image, while the 82 (b) image validation training correctly detected two airplanes. When the number of validation images was less, training took 23 minutes, and when there were more images, the training of the model was completed in 31 minutes.

For training and validation of the model, Python train.py is used (Figure 5.3 a and b). Thus, the best.pt weight file is created in the analysis results. It is the modeling where the best performances are obtained. In order to test the model, test images are written

with detect.py, the images are tested by specifying the best.pt weight file and image sources (Figure 5.3 c).



(a)



(b)

(c)

Figure 5.3 Python Codes for training the model using 50 (a) and 82 (b) validation images and testing (c) in YOLOv3

5.2 YOLOv5 Tests and Analysis

In the second training model for object detection, the datasets were trained in two different versions as YOLOv5s and YOLOv5x versions. YOLOv5 is an object detection model family developed by Ultralytics and has 5 different versions, n (nano), s (small), m (medium), l (large) and x (extra large), optimized for different needs. These versions are balanced according to different criteria such as speed (FPS), accuracy, model size and hardware requirements. While YOLOv5s offers lower

accuracy and faster processing, YOLOv5x is slower for applications that require maximum accuracy.

From this stage of the thesis study, YOLO could not be used with Goggle Colab. Since YOLO versions requiring high memory and processing speed were used, Python 3.8.10 version was installed on a computer with Ubuntu operating system. In the use of Python programming language, it was configured with Pytorch library version 2.4.1 algorithms. Thus, the training of the model was accelerated and the operations were carried out successfully.

First, when the model training "epoch" value was changed, a comparison was made in the YOLOv5s version. Then, when the same parameters were used, the YOLOv5s and YOLOv5x versions were trained. Thus, the variables of the epoch value and different versions in the same version were tested.

Table 5.3 Result analysis of using YOLOv5s 50 epoch

epoch	**metrics/ precision**	**metrics/ recall**	**metrics/ mAP_0.5**	**metrics/ mAP_0.5:0.95**
1	0.0018204	0.078652	0.0014206	0.0003445
2	0.0091356	0.38202	0.01631	0.0040513
3	0.078886	0.18352	0.044011	0.0091638
4	0.2396	0.19476	0.13623	0.040316
5	0.24984	0.25697	0.17233	0.044358
⋮	⋮	⋮	⋮	⋮
46	0.90212	0.72487	0.84179	0.46689
47	0.90444	0.70894	0.82833	0.46772
48	0.90045	0.71141	0.81757	0.46876
49	0.88864	0.7173	0.82194	0.46858
50	0.89217	0.71536	0.81622	0.46454

Table 5.4 Result analysis of using YOLOv5s 100 epoch

epoch	**metrics/ precision**	**metrics/ recall**	**metrics/ mAP_0.5**	**metrics/ mAP_0.5:0.95**
1	0.0015923	0.048689	0.0012022	0.00032385
2	0.0069001	0.3633	0.013087	0.0029095
3	0.12822	0.17228	0.070415	0.016013
4	0.20756	0.14981	0.11843	0.030828

5	0.10958	0.38577	0.079115	0.019508
⋮	⋮	⋮	⋮	⋮
96	0.92187	0.7191	0.8293	0.51661
97	0.92253	0.71536	0.83003	0.51492
98	0.86279	0.75655	0.82823	0.51722
99	0.86592	0.75281	0.82731	0.5146
100	0.89031	0.72959	0.8247	0.51157

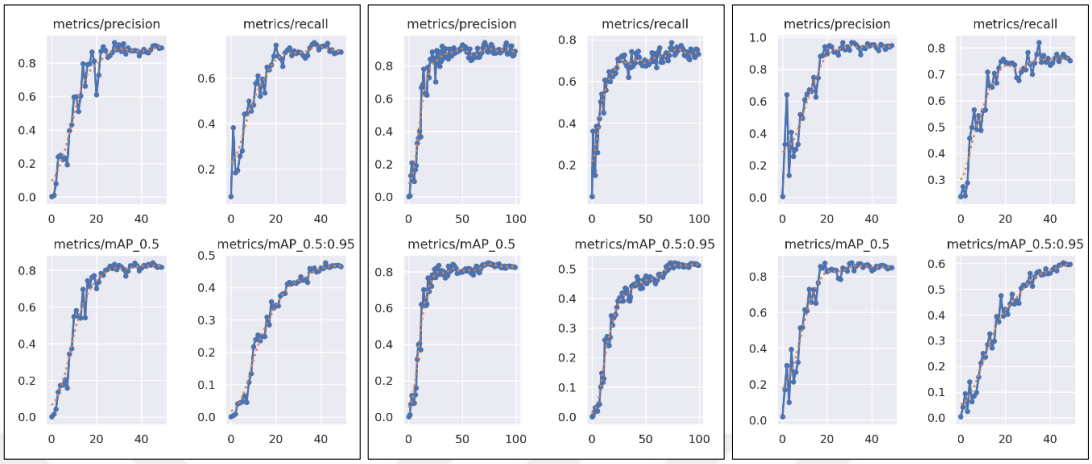
Table 5.5 Result analysis of using YOLOv5x 50 epoch

epoch	**metrics/ precision**	**metrics/ recall**	**metrics/ mAP_0.5**	**metrics/ mAP_0.5:0.95**
1	0.0046667	0.23596	0.018563	0.0034937
2	0.33099	0.27341	0.16954	0.039954
3	0.64093	0.2397	0.30275	0.094256
4	0.138	0.28839	0.098246	0.023896
5	0.4072	0.45693	0.39427	0.13958
⋮	⋮	⋮	⋮	⋮
46	0.94598	0.76404	0.86254	0.59591
47	0.92415	0.77577	0.85538	0.60328
48	0.94581	0.76404	0.84384	0.59868
49	0.93982	0.76037	0.84921	0.59605
50	0.948	0.75111	0.8502	0.5968

As seen in Table 5.3 and Table 5.4 above, when the epoch value is increased in the same model and version, the correctly detected object rate (precision) and model performance (mAP@0.5) are higher. While 50 Epochs are selected for model training in Table 5.3, the Epoch value is selected as 100 in Table 5.4, keeping other parameters constant. The Epoch value indicates how many times the model will process the entire dataset during training. In other words, it is reading the bounding boxes repeatedly. In short, the more the model reads the trained images, the more it learns and, accordingly, the better the object detection it makes. On the other hand, the training time of the model is longer.

Another table Table 5.5 shows the model analysis results using reference parameters with the YOLOv5x version, which is designed as the largest and most powerful model of the YOLOv5 family. YOLOv5x is the largest, highest accuracy, but slowest and most resource-consuming model among other 5th generation YOLO variants. When compared with other tables (Table 5.3 and Table 5.4), it is understood that although

the epoch value of the 5s version is higher, the YOLOv5x result data detects more objects and shows better performance.



(a) (b) (c)

Figure 5.4 Result charts of using YOLOv5s 50 epoch (a), YOLOv5s 100 epoch (b) and YOLOv5x 50 epoch (c)

When the above graphs are examined, it is understood that where the epoch number is high, the points in the graph are more concentrated, meaning that more objects are detected. In addition, in the graph of the test made with YOLOv5x, it is seen from the sharp lines and points in the graph that clearer and more accurate predictions are made.

In general, the summary of the experiments conducted at this stage is that using the latest version within the version and increasing the epoch value improves the performance and accuracy of the model. However, this requires more hardware, GPU and time.

(a) (b) (c)

Figure 5.5 Object detection of using YOLOv5s 50 epoch (a), YOLOv5s 100 epoch (b) and YOLOv5x 50 epoch (c)

When the object detection photos obtained after the 3 model trainings were examined, it was observed that there were differences between them. When the 3 same photos shown in the blue square were examined, it could be seen that there was a difference between the object detection validation values of the YOLOv5s 50 Epoch (a), YOLOv5s 100 Epoch (b) and YOLOv5x 50 Epoch (c) variants used in the test. The YOLOv5x (c) test had the highest validation success. On the other hand, in the photos shown in the yellow square, the YOLOv5x 50 Epoch (c) model could not detect the object in the training. It was revealed that the YOLOv5s 100 Epoch (b) model training perceived the vehicle next to the aircraft carrier as an airplane in the photos shown in the green square and made an incorrect object detection.

5.3 YOLOv8 Tests and Analysis

In this training for object detection, YOLOv8 series, which is one of the most widely used versions of YOLO today, was used. Just like YOLOv5, different variants have been developed according to their sizes. It offers many improvements based on the success of YOLOv5 and includes modern deep learning techniques. YOLOv8 is a powerful computer vision model that has a more modern and flexible structure than previous YOLO versions, offering high accuracy, fast inference and more application support. It offers an ideal solution for both research and commercial projects.

At this stage of the study, the YOLOv8x variant with the highest accuracy rate among the 8th generation YOLO models was preferred. In this model training, the batch size value was changed while the object was detected, keeping other parameters constant. First, the reference value batch size was selected as 10, then the batch size was reduced to 5 and the model was trained. Thus, the effect of batch size on the model in object detection was compared. Batch size refers to the number of data samples given to the neural network at the same time in each training step of the model. The model calculates the loss value by processing this data group and then updates the weights with the backpropagation process. In object detection models such as YOLO, the batch value is adjusted to balance training time, model accuracy and GPU memory usage.

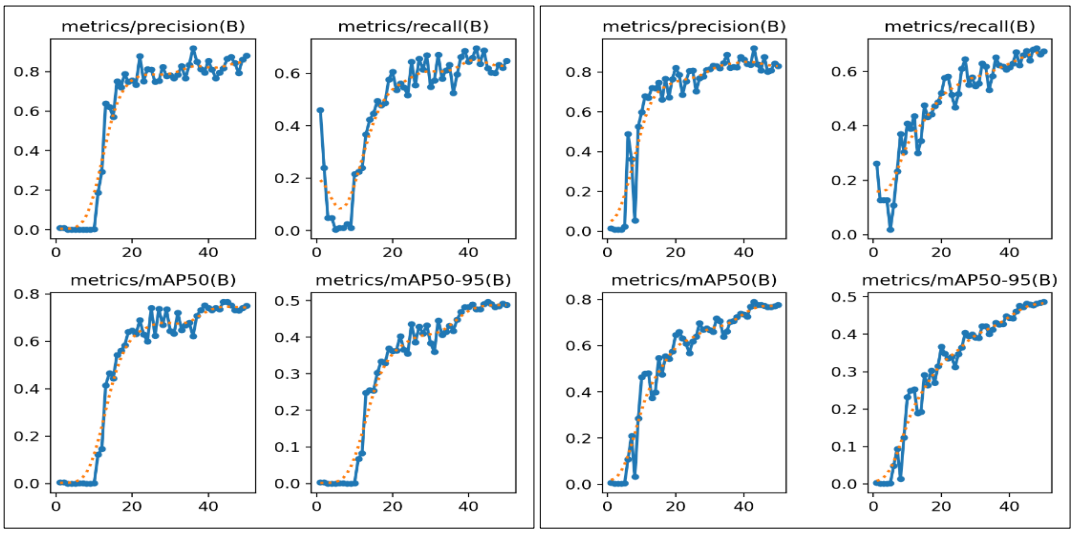
Table 5.6 Result analysis of using YOLOv8x 10 batch

epoch	**metrics/ precision**	**metrics/ recall**	**metrics/ mAP_0.5**	**metrics/ mAP_0.5:0.95**
1	0.00823	0.46067	0.00562	0.00311
2	0.00871	0.2397	0.00438	0.00267
3	0.00053	0.04869	0.00028	0,00005
4	0.00053	0.04869	0.00028	0,00005
5	0.00016	0.00375	0,00007	0,00001
⋮	⋮	⋮	⋮	⋮
46	0.87288	0.603	0.75267	0.49107
47	0.84238	0.60049	0.73247	0.4823
48	0.79247	0.63296	0.72982	0.48436
49	0.86015	0.62199	0.74048	0.49181
50	0.88031	0.64794	0.74951	0.48874

Table 5.7 Result analysis of using YOLOv8x 5 batch

epoch	metrics/ precision	metrics/ recall	metrics/ mAP_0.5	metrics/ mAP_0.5:0.95
1	0.01416	0.26217	0.00772	0.00382
2	0.00808	0.12734	0.00377	0.00152
3	0.00808	0.12734	0.00377	0.00152
4	0.00808	0.12734	0.00377	0.00152
5	0.02417	0.01873	0.00576	0.0021
⋮	⋮	⋮	⋮	⋮
46	0.87672	0.63925	0.77244	0.47965
47	0.80144	0.68028	0.76681	0.4771
48	0.80753	0.68539	0.76727	0.48141
49	0.84392	0.66292	0.77092	0.4841
50	0.83032	0.67416	0.77683	0.48633

As can be seen from the analysis results, reducing the batch size had little effect on the performance of the training datasets and object detection in our study. Also, when the batch size was reduced, the rate of correctly detected objects in the model (precision) decreased. On the other hand, when we look at how many of all real objects were detected (recall), the recall value increased as the batch size decreased. The training of these models took approximately 8-9 hours. This version, which required a lot of time and data, was lower than the previous versions in terms of performance in our study. However, the precision value was at the highest levels compared to the others.



(a) (b)

Figure 5.6 Result charts of using YOLOv8x 10 batch (a) and 5 batch (b)

When the graphs are examined, it can be seen that decreasing the batch size caused the training estimates to be too unstable, and therefore a linear shape was not formed in the graph. The irregularity in the perception of objects caused a zigzag structure to form in the graphs. Since the values for the model's performance were similar to each other, the graphs were similar.

(a) (b)

Figure 5.7 Object detection of using YOLOv8x 10 batch (a) and 5 batch (b)

When looking at the images above, three examples are given to compare the effects of batch size on object detection. When the images in the green square are examined, it is seen that the object detection performance is higher when the batch size is 10 (a). In the images in the blue square, when the batch size is 5 (b), the detection rate (recall) of real objects is higher, just as seen in the analysis tables. Out of the 3 aircraft on the ship, only 1 was detected when the batch size was 10 (a), whereas 2 aircraft were detected when the batch size was 5 (b). In the yellow squares at the bottom, it is understood that the aircraft in the image cannot be detected at both batch size values, but the car in the image is perceived as an aircraft when the batch size is 5 (b).

5.4 YOLOv10 Tests and Analysis

The final test for object detection was conducted with the latest version of the YOLO deep learning model, YOLOv10. YOLOv10 aims to surpass previous versions with more flexible use, wider device support and higher performance. In the study, the detection of objects was examined by changing the image size from our reference values on this latest version. Pixel value is a numerical expression representing the RGB (Red, Green, Blue) color intensity of each pixel in the image. These values are used as input data for deep learning models such as YOLO and help the model understand the features in the image. Correct scaling, appropriate resolution and editing of color channels allow the model to perform better. While low pixel density makes it difficult for the model to detect small objects, important details may be lost in a high-resolution image and the accuracy of the model may decrease. In addition, the very high image resolution in YOLO allows more detail detection, but it is costly. For optimal training and results, a resolution appropriate to the GPU capacity should be selected. This reference value was selected as 640x640 for the test. This image size value was increased to 750 to see the effect on object detection.

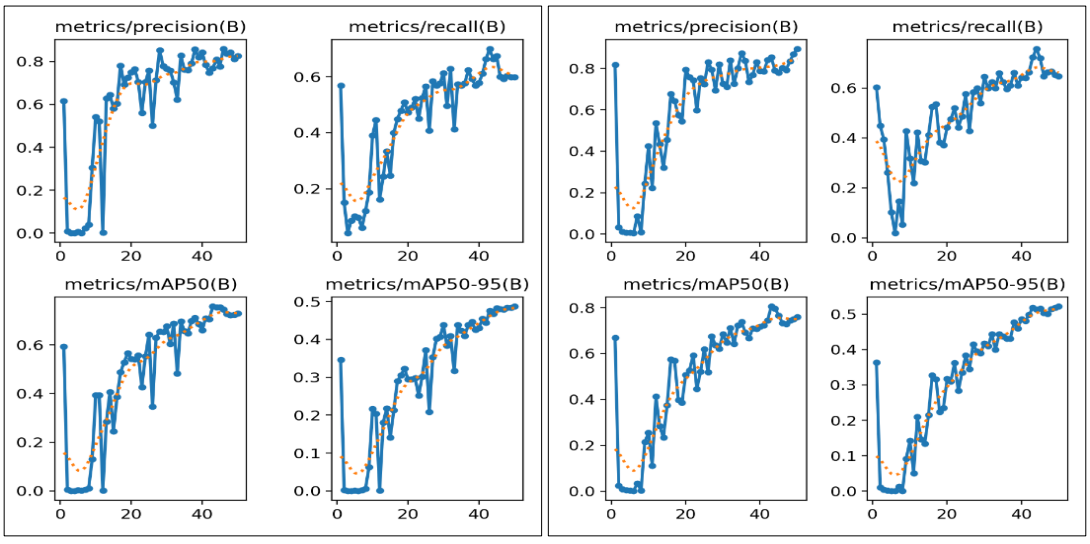
Table 5.8 Result analysis of using YOLOv10x 640 image pixel

epoch	metrics/ precision	metrics/ recall	metrics/ mAP_0.5	metrics/ mAP_0.5:0.95
1	0.61666	0.56929	0.59378	0.34577
2	0.00818	0.14981	0.00424	0.00211
3	0.00047	0.0412	0.00025	0,00009
4	0.00093	0.08614	0.00051	0.00017
5	0.00698	0.10112	0.00297	0.00142
⋮	⋮	⋮	⋮	⋮
46	0.86032	0.59979	0.74602	0.48044
47	0.83601	0.59187	0.72787	0.4791
48	0.84288	0.603	0.72146	0.48436
49	0.81286	0.59925	0.72346	0.48345
50	0.82607	0.59925	0.72959	0.48729

Table 5.9 Result analysis of using YOLOv10x 750 image pixel

epoch	metrics/ precision	metrics/ recall	metrics/ mAP_0.5	metrics/ mAP_0.5:0.95
1	0.81843	0.603	0.66908	0.36357
2	0.03217	0.44944	0.02378	0.01026
3	0.01216	0.39326	0.00934	0.00482
4	0.00799	0.26217	0.0046	0.00206
5	0.00657	0.10238	0.00282	0.00109
⋮	⋮	⋮	⋮	⋮
46	0.80343	0.64794	0.73352	0.50328
47	0.79281	0.66292	0.72946	0.50121
48	0.83613	0.66667	0.74477	0.51417
49	0.86959	0.65169	0.74934	0.5189
50	0.8958	0.64794	0.76038	0.52208

As seen in both tests performed with YOLOv10x, all result values increased as the image pixel value increased. The performance of the model, object detection rate and accuracy levels increased. On the other hand, the system requirements (GPU, CPU, memory etc.) and time required for training the model increased. While training the model took approximately 6 hours in the test using 640 image size (Table 5.8), it took almost 9 hours in the test using 750 image size (Table 5.9). Increasing the image size from 640 to 750 increased the model's running time by 50%. It also overloaded the hardware systems.



(a) (b)

Figure 5.8 Result charts of using YOLOv10x 640 pixel (a) and 750 pixel (b)

As seen in the charts, when the image size is 640 pixel (a) and 750 pixel (b), the graphics are generally similar. If the input data to the trained system is activated with 750 pixel (b), the graphic is in the form of a softer line and more similar to the reference line. It has been analyzed that the recall value is particularly unstable as the image size decreases and jumps to sharp points. It has been experienced from the charts and analyzes that when the image size increases, the model system is more effective, and the object detection rate and accuracy increase.

(a) (b)

Figure 5.9 Object detection of using YOLOv10x 640 pixel (a) and 750 pixel (b)

When training the model with the YOLOv10x version using the same images, changing the image size caused different results. When the same images are compared, it is seen that the model cannot detect the object in the image (a) trained with 640 pixel, namely the airplane, while it can be seen that it detects the object with 750 pixel (b). Another difference was in the detection performance of the objects. The detection performance of the object in the same image is generally higher when the image size is selected as 750 pixel, as seen in the analysis results. However, when the image size is made 750 pixel, a high resolution is created on the image. This caused the model to focus on more details on each image while training. When the images above are examined, the error rate increased in the system that focuses on more details in the images as the image size increases. The model perceived objects such as cars and buildings as airplanes, thus making the object detection incorrect.

After modeling and verification of all training, 42 test images were tested to check how the system works. There are no aircraft objects in 3 images out of 42 test images. All trained models could not detect objects in 3 images as a result of the test, as in reality.

While analyzing the test results of each model, the best.pt weights that exhibited the best performance were used as modeling. During the test, the threshold value was kept as low as 0.1 and it was requested to detect as many objects as possible in the test images. As can be seen in the table below, the post-test analysis results were similar to the training and validation results, and the version that detected the most objects in the tested images and had the most efficient performance was YOLOv3. In the analyses performed on the validation images, it was seen that the YOLOv3 and YOLOv5x models had very close values to each other.

Table 5.10 Test results with best trained models

Version	metrics/precision	metrics/recall	metrics/mAP_0.5
YOLOv3	0.976	0.569	0.779
YOLOv5x	0.876	0.468	0.625
YOLOv8x	0.794	0.495	0.606
YOLOv10x	0.858	0.476	0.603

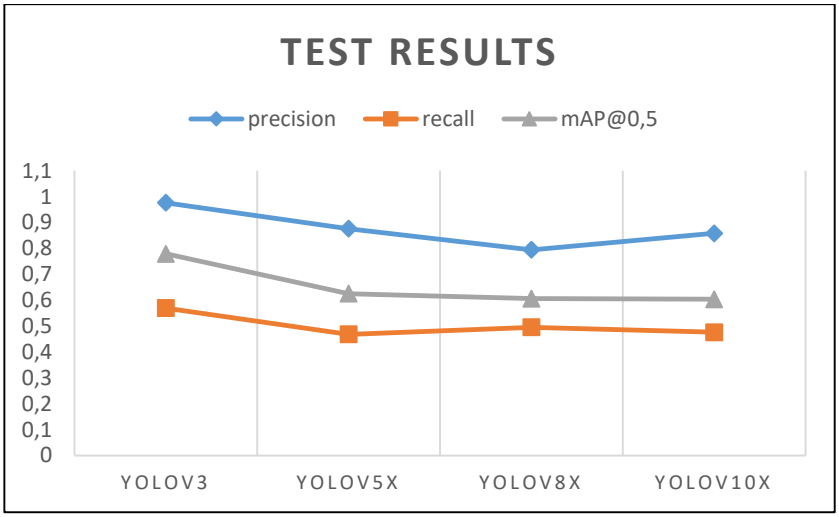


Figure 5.10 Test results charts with best trained models

As can be seen from the table, the model with the highest number of object detections is the YOLOv3 model, which detects objects in 97% of the test images, while the model with the lowest number of object detections belongs to YOLOv8x with a rate of 79%. In addition, in the study on the detection of a single object, it was seen that the version with the best model performance according to the test images was YOLOv3.

As a result of all experiments;

- A higher number of train and validation images in the dataset provides more verification for the detection of the object. For this reason, the dataset should be selected as large as possible for the most efficient and high-performance object detection.

- The higher the Epoch value, which is a criterion expressing how many times the model processes the dataset completely during the training process, the higher the performance for the detection of the object. However, the epoch value directly affects the test time of the model. The important thing is not to choose too many epochs but to choose the most ideal value.

- As a result of our tests, choosing a batch size of 5 or 10 did not affect the training of our dataset much. The batch size determines the number of data samples (images) that the model processes simultaneously during training. This value creates a load on the memory and computer processor. Choosing this parameter in very large numbers will directly increase the cost and duration of the model.

- In the last stage of the study, it was observed that the changed image size is important in training the model and is related to the accuracy of the system. It can be said that although low-resolution images give fast results, the probability of detection and error is higher. The better the image quality, the better the model is trained and the higher the validation rate.

CHAPTER 6

CONCLUSION

Today, object detection with UAV images, which are the most critically important aircraft for countries, has been addressed within the scope of the thesis. Especially due to confidentiality conditions, taking images from critically important places such as airports and military airfields has been prohibited by many institutions and organizations. In fact, air defense systems are used to prevent images from being taken from military airfields and very strict measures are being taken. For this reason, the number of images to be used for the dataset was limited. The results of different variables in the new versions of the YOLO model were examined with the same dataset used within the scope of the experiments. In the tests, various experiments were conducted with the variances of the YOLO deep learning model's YOLOv3, YOLOv5, YOLOv8 and the latest version, YOLOv10. UAV technology, which is rapidly developing day by day and has a keystone role, also serves different usage purposes for users. Artificial intelligence and deep learning technology, which are developing like UAV technologies, have provided some conveniences in daily life by being used together.

The tests conducted for the study also checked the suitability of the reference parameters and versions for model training. These reference values are "epoch, metrics/precision, metrics/recall and metrics/mAP_0.5". The data set size and resolution of the study, the quality of the hardware used for the tests conducted are other factors affecting model training. When all variables are considered, it was concluded that the most effective and efficient result for our study was the test conducted with the YOLOv3 value model. The test results conducted with both the computer hardware we have and the data set we have show this.

According to the test results, the performance (metrics/mAP_0.5:0.95) value of the models made for object detection showed differences between versions. While it was observed that the best trained version was 0.596 with YOLOv5x in the performances exhibited by the system, it can be seen that YOLOv8 is especially insufficient. The

fact that the metrics/mAP_0.5:0.95 value is theoretically below 0.5 indicates that the system needs to be improved. In addition, according to the results obtained with the test images, it was seen that the version with the highest performance is YOLOv3

The precision value, which shows that almost all of the detected objects are correct during the training of the system, is 0.948 for YOLOv5x. This value is the highest value among the validation analyses conducted with other variants and versions. In addition, it can be seen that similar values are obtained in other tests. In the analysis conducted with the test images, the version with the highest object detection was YOLOv3. This means that the labeling made for our data set and the similarity rate of the planes in the images used for training and the validation images are high.

In object detection models such as YOLO, the metric that measures the model's ability to detect objects correctly is the recall parameter. This value was measured in the YOLOv3 version test, where the highest 50 validation images were used. In the YOLOv3 test conducted with 50 validation images, the recall value was 0.894, while in the YOLOv5x experiment conducted with 82 images, it was measured as 0.85. The reason why this value is higher in the basic version than in the other advanced versions is that it does not analyze too many details in the images. Especially the latest versions, although they work more result-oriented, tend to have erroneous detection and measurement when there is very little similarity because they perform very detailed analysis.

In deep learning model studies conducted for the detection of an object, the absolute necessary parameters for the training of the dataset are the dataset size, epoch number, image dimensions and batch size. In the YOLO experiments conducted with different versions, the status of the trained models was observed according to the changes in these parameters. It can be said that these parameters are of critical importance. Because the selection of incorrect values causes the performance for the detection of the object to be insufficient. In this regard, the model can reach high numbers in both cost and training time.

Instant object detection applications are developing day by day. Many institutions are working on different algorithms and deep learning models. Today, YOLO, which is

the easiest and most user-friendly to use in the field of object detection, is updating its current versions and making efforts for the use of YOLO in different areas. Better results can be obtained for each developing model. The reason why the ideal version is YOLOv3 in the study is that the best results emerge for the content of our dataset and the hardware we use. Similar values were observed in the YOLOv5x study. As followed in our study, if the object detection results from the first version to the last version in YOLO are sufficiently high, better results can be obtained.

Object detection technology from UAV images is a rapidly developing field in the field of artificial intelligence and computer vision, where significant innovations are experienced. It will provide faster and more effective results with more powerful deep learning models observed in recent years. It will not be sufficient with system trainings and will be used in real-time applications and will function instantly. It will not need very large datasets by performing self-supervised and low-level learning. In addition, datasets will increase and access to information will be easier. Application areas will keep up with today's technology, such as agriculture, health and autonomous systems, in addition to defense and security.

REFERENCES

[1] Tang, G., Ni, J., Zhao, Y., Gu,Y., Cao,W., A Survey of Object Detection for UAVs Based on Deep Learning. Retrieved August 25, 2024, https://doi.org/10.3390/rs16010149. December 29, 2023

[2] Ge Z., Liu S., Wang F., Li Z., Sun J., YOLOX: Exceeding YOLO Series in 2021. Retrieved August 26, 2024, https:// arxiv.org/abs/2107.08430. August 6, 2021

[3] Azam, B., Khan, M. J., Bhatti, F. A., Maud, A. R. M., Hussain, S. F., Hashmi, A. J. ve Khurshid, K., Aircraft detection in satellite imagery using deep learning-based object detectors. Retrieved August 29, 2024, https://doi.org/10.1016/j.micpro.2022.104630. October, 2022

[4] Kharchenko, V. ve Chyrka, I., Detection of airplanes on the ground using YOLO neural network. Retrieved September 1, 2024, https://ieeexplore.ieee.org/document/8460392. September 13, 2018

[5] Burgaz M., Budak C., Derin öğrenme Algoritmalarıyla İnsansız Hava Araçlarından Elde Edilen Görüntülerde Nesne (Silah) Tespiti. Retrieved September 4, 2024, https://doi.org/10.24012/dumf.1116534. June 28, 2022

[6] İHA Tarihçesi, (n.d.). Retrieved September 9, 2024, https://www.thk.org.tr/iha.

[7] MQ-9 Reaper. Retrieved September 9, 2024, https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle. September 7, 2024

[8] KARATAŞ, M., KEFELİOĞLU O., Sabit Kanatlı İnsansız Hava Aracı Tasarımı, Graduation Project, Karadeniz Technical University. June, 2021

[9] MQ Predator. Retrieved September 9, 2024, https://tr.wikipedia.org/wiki/General_Atomics_MQ-1_Predator. June 13, 2024

[10] Korchenko A.G., Illyash O.S., The Generalized Classification of Unmanned Air Vehicles. Retrieved October 15, 2024, https://ieeexplore.ieee.org/document/6705275. February 15, 2022,

[11] Directorate General of Civil Aviation, İnsansız Hava Aracı Sistemleri Talimatı (SHT-İHA). Retrieved October 15, 2024, https://web.shgm.gov.tr/documents/sivilhavacilik/files/mevzuat/sektorel/talimatlar/2020/SHT-IHA_Rev-04.pdf. December 10, 2021

[12] What Do The UAS Groups Mean?. Retrieved October 20, 2024, https://cuashub.com/en/content/what-do-the-uas-groups-mean. July 14, 2023

[13] Pastor E., Royo P., Rubio J.L., UAV Payload and Mission Control Hardware/Software Architecture, , IEEE Aerospace and Electronic Systems Magazine. Retrieved October 20, 2024, https://ieeexplore.ieee.org/document/4271319. July 6, 2007

[14] Hoş S., Neural Networks Nedir? Nasıl Çalışır ?. Retrieved October 23, 2024, https://www.hosting.com.tr/blog/neural-networks. January 31, 2023

[15] What is a neural network?, (n.d.). Retrieved November 1, 2024, https://www.ibm.com/topics/neural-networks

[16] Sinir Ağı nedir?, (n.d.). Retrieved November 1, 2024, https://aws.amazon.com/what-is/neural-network

[17] Chen X., Human Resource Matching Support System Based on Deep Learning. Retrieved November 1, 2024, https://doi.org/10.1155/2022/1558409. June 10, 2022

[18] Yıldırım E., Yapay Sinir Ağı(Artificial Neural Network) Nedir?. Retrieved November 2, 2024, https://www.veribilimiokulu.com/yapay-sinir-agiartificial-neural-network-nedir. May 2, 2020

[19] Babs T., The Mathematics of Neural Networks. Retrieved November 2, 2024, https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05. July 14, 2018

[20] O'Shea K., Nash R., An Introduction To Convolutional Neural Networks. Retrieved November 2, 2024, https://www.researchgate.net/publication/285164623. 2015

[21] Mercier M., Derin sinir ağı nedir?. Retrieved November 3, 2024, https://botpress.com/tr/blog/deep-neural-network. November 23, 2022

[22] Berchane N., Artificial Intelligence, Machine Learning, and Deep Learning: Same context, Different concepts. Retrieved November 3, 2024, https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts. April 16, 2018

[23] Derin Öğrenme. Retrieved November 3, 2024, https://tr.wikipedia.org/wiki/Derin_%C3%B6%C4%9Frenme. May 11, 2024

[24] Derin Öğrenme Nedir? Makine Öğrenimi ve Yapay Zeka İle Arasındaki Farklar Nelerdir?. Retrieved November 3, 2024, https://bulutistan.com/blog/derin-ogrenme-nedir. March 14, 2022

[25] Holdsworth J., Scapicchio M., What is deep learning?. Retrieved November 5 2024, https://www.ibm.com/topics/deep-learning. June 17, 2024

[26] Koç E., Evrişimli Sinir Ağlarına Giriş. Retrieved November 5, 2024, https://miuul.com/blog/evrisimli-sinir-aglarina-giris. June 6, 2022

[27] Islam M.A., Reduced Dataset Neural Network Model for Manuscript Character Recognition. Retrieved November 5, 2024, https://www.researchgate.net/publication/343675998. July 2020

[28] Nwankpa C.E., Ijomah W., Gachagan A., Marshall S., Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. Retrieved November 10, 2024, https://arxiv.org/pdf/1811.03378. November 8, 2018,

[29] Josifov R.C., Deep learning based computer vision for aerial-view street object detection and classification. Retrieved November 10, 2024, https://www.researchgate.net/figure/YOLO-grid-and-predictionsSource_fig6_362209154. July, 2022

[30] Boesch G., Object Detection: The Definitive 2025 Guide. Retrieved November 10, 2024, https://viso.ai/deep-learning/object-detection. October 4, 2024

[31] Bankalar D., YOLO'nun Gizemini Çözmek: Nesne Algılama Algoritmasını Örneklerle Anlamak. Retrieved November 11, 2024, https://www.datalabelify.com/tr/nesne-algilama-icin-yolo-algoritmasi. April 9, 2022

[32] YOLO: Algorithm for Object Detection Explained. Retrieved November 11, 2024, https://www.v7labs.com/blog/yolo-object-detection. January 17, 2023

[33] Liu W., Anguelov D., Erhan D, Szegedy C., Reed S.,Fu C.Y., SSD: Single Shot MultiBox Detector. Retrieved November 11, 2024, https://arxiv.org/abs/1512.02325. December 29, 2016

[34] Taher C.H., Object Detection Algorithms: A Comprehensive Overview. Retrieved November 11, 2024, https://medium.com/@chaima.hajtaher/object-detection-algorithms-a-comprehensive-overview-ffb78970eb2b. July 30, 2023

[35] Joshi S., Top Object Detection Models in 2024. Retrieved November 11, 2024, https://www.hitechbpo.com/blog/top-object-detection-models.php#introduction-to-object-detection. May 13, 2024

[36] Sambasivarao K., Non-maximum Suppression (NMS). Retrieved November 15, 2024, https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. October 1, 2019

[37] Nelson J., What is YOLO? The Ultimate Guide. Retrieved November 15, 2024, https://blog.roboflow.com/guide-to-yolo-models. July 17, 2024

[38] Alif M.A.R., Hussain M., YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain. Retrieved November 15, 2024, https://arxiv.org/html/2406.10139v1. June 14, 2024

[39] Clemson. Computer Vision Project. Retrieved November 15, 2024, https://universe.roboflow.com/clemson-1rt4q/experiment-2-dvkpm. July, 2023

[40] Nakamura T., Military Aircraft Detection Dataset, (n.d.). Retrieved November 15, 2024, https://www.kaggle.com/a2015003713/datasets.

[41] İmeci O., Derin Öğrenme Algoritması (YOLO) ile Fotoğraflardaki ya da Canlı Görüntü Üzerinden Elektrik Sayaçlarının Tespiti. Retrieved November 16, 2024, https://tr.linkedin.com/pulse/derin-%C3%B6%C4%9Frenme-algoritmas%C4%B1-yolo-ile-foto%C4%9Fraflardaki-ya-onur-i%CC%87meci. August 5, 2023

[42] Jocher G., YOLOv5 in PyTorch. Retrieved November 15, 2024, https://github.com/ultralytics/yolov5. November 22, 2022

[43] Solawetz J., What is Mean Average Precision (mAP) in Object Detection?. Retrieved November 15, 2024, https://blog.roboflow.com/mean-average-precision. May 6, 2020