İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
ENGINEERING FACULTY
COMPUTER ENGINEERING DEPARTMENT

GRADUATION PROJECT

ARTIFICIAL INTELLIGENCE IN TRANSPORTATION

BACHELOR'S THESIS

Thesis Advisor: Dr. MUHAMMED ERDEM İSENKUL

Prepared by:

Mueyyed GARZUDDİN 1306180132
Muhammed MUBAKEROGLU 1306180164

PREFACE

Today, Artificial Intelligence technologies have become increasingly widespread and can be used in real life. As the computational power of embedded applications has increased, many artificial intelligence models can now work in real time. On the other hand, with the widespread use of these devices in all sectors of life, the use of flying cars has become possible and applicable, and it has become an urgent need in many areas of life, especially in the ability to achieve speed in completing the necessary tasks, especially in the field of ambulance.

Although there are wide spread of systems that support the operations of the flying car, most of these systems do not benefit from the power of artificial intelligence and therefore they cannot provide required interaction.

Therefore, within the scope of this project, a system,that will be developed, provides a sufficient interaction to ensure the safe landing of the aircraft without human intervention and provides a high level of interaction by using computer vision techniques.

We would like to thank our dear teacher, **Dr.Muhammed ERDEM ISENKUL**, for all his support and assistance throughout this project.

In addition, we would like to thank all our professors at Istanbul University-Cerrahpaşa who guided us, gave us motivation and inspiration, and helped us to become an engineer throughout our education life.

Finally, we would like to thank our family who supported us throughout our education life and always stood by us.

MUHAMMED MUBAKEROGLU

MUEYYED GARZUDDİN

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

- **SSD** : Single Shot Detector
- **mAP** : Mean Average Precision
- **FPS** : Frame per Second
- **AI** : Artificial intelligence
- **ESA** : Evrişimli Sinir Ağları
- **IQ** : Intelligence Quotient
- **ML** : machine learning
- **DL** : deep learning
- **UAP** : Uçan Araba Park (Flying Car Parking)
- **UAI** : Uçan Ambulans İnişi (Flying Ambulance Landing)
- **NN** : Neural Network
- **YSA** : Yapay sinir ağları (Artificial neural networks)
- **RGB** : Red Green Blue

SUMMARY

ARTIFICIAL INTELLIGENCE IN TRANSPORT

Nowadays, Artificial Intelligence technologies are becoming more common and are entering most areas of real life. With the increasing computing power of embedded systems' devices, many AI models are now applicable in real time.

On the other hand, with the spread of these devices in all sectors of life, the use of the flying car has become an urgent need in many aspects of life, especially the possibility of achieving speed in completing the required tasks, A good example here is in the field of ambulance.

In this project, a neural network model will be created using deep learning techniques. This network will be able to capture videos from a camera fixed on a flying car. It will be able to take the captured video as inputs and process them to come up with desired results.

There are two different locations where the drone can land:

1- (UAP: Flying Car Park)

2- (UAI: Car Ambulance Park).

The task of this NN is to detect objects (cars, trains,.. _) and labelling them with dynamic squares according to their different sizes. Also, when the Flying Car or Flying Ambulance Car wants to land, the Model must have to verify whether the landing park is empty (safe) or not.

The model existed in this project has to be trained of discovering the desired targets using some modern technologies such as (Computer Vision Models (darknet), OpenCV, Deep Learning, Python (Numpy-Sicktlearn). Using these technologies will increase processing FPS(speed Frames Per) and also producing consistent and accurate results.

As a result, this project is planned to be designed to contribute to the society due to its contribution in the field of providing a safe landing of the flying car, which will play an important role in many areas of life in the future.

1. INTRODUCTION

Based on computer vision and machine learning, this project has trained a model that helps to ensure the landing area of the flying car automatically and safely. By adopting a neural network, the project is ready to support two types of flying cars (regular car and ambulance car).

With this project, the Flying Car will be equipped with a neural network that processes the images it receives from the cameras installed on the aircraft. In addition, this neural network is defined to detect moving objects that may hinder the landing process. Existing models are used and modified to suit the mission of the flying car to ensure the success of the mission required by the network. This is applied to both types of cars. (Normal and Ambulance).

The main problems solved by the project are presented as follows:

- The need to complete the landing process in the right place
- Possibility of occupying the ground by objects that may be endangered during the descent
- Landing without the need for aircraft steers to control the aircraft during landing

Within the scope of this study, the main objective of the project is to be informed of the location of the flying car with artificial intelligence techniques and methods, and to realize the landing process automatically and safely without the need for any human intervention.

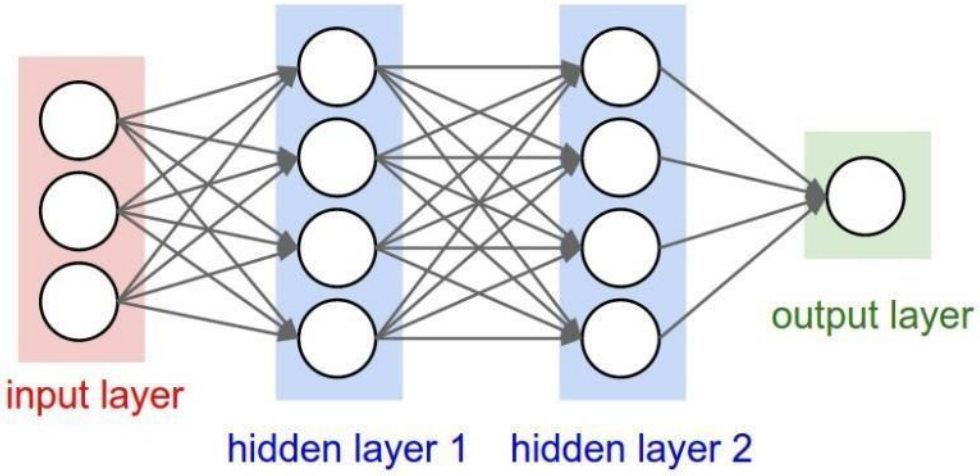
2. GENERAL SECTIONS

2.1. Artificial intelligence

Artificial intelligence is defined in the scientific world as the ability of a computer or computer-assisted machine to perform tasks related to higher logical processes, such as human-specific qualities, finding a solution, understanding, inferring a meaning, generalization, and learning from past experiences.

2.2. Artificial neural networks

Artificial neural networks (ANNs) are computer software in which basic functions such as generating new data from the data collected by the brain by learning, remembering and generalizing by imitating the learning path of the human brain are performed. Artificial neural networks; Inspired by the human brain, it emerged as a result of the mathematical modeling of the learning process. Artificial neural networks are also defined by names such as parallel distributed networks, connected networks, neuromorphic networks.



Shape 2-1 simple YSA

2.3. Computer Vision

Computer vision is basically trying to do tasks or functions that a human can do visually in a computer system environment. It is the process of making decisions in a way that people can decide on digital images or video images and making decisions according to the result. Computer vision uses the methods of creating, processing, analyzing and making meaningful the digital image in order to produce numerical or symbolic information on the image.

Computer Vision backgrounds come in a wide variety. Due to the models created with the help of statistics, physics, geometry and learning theory, each field of study requires separate expertise.

The sub-branches used in the project can be listed like this:

2.3.1. Recognition

The classic problem in computer vision, image processing, and machine vision is to determine whether image data contains a particular object, feature, or activity after it has been received.

2.3.2. Object Detection

It is also known as the detection system. It is basically scanning digital video videos for a specific condition based on simple and fast calculations. As an example, we can give the fire detection of a drone reconnaissance in the forest.

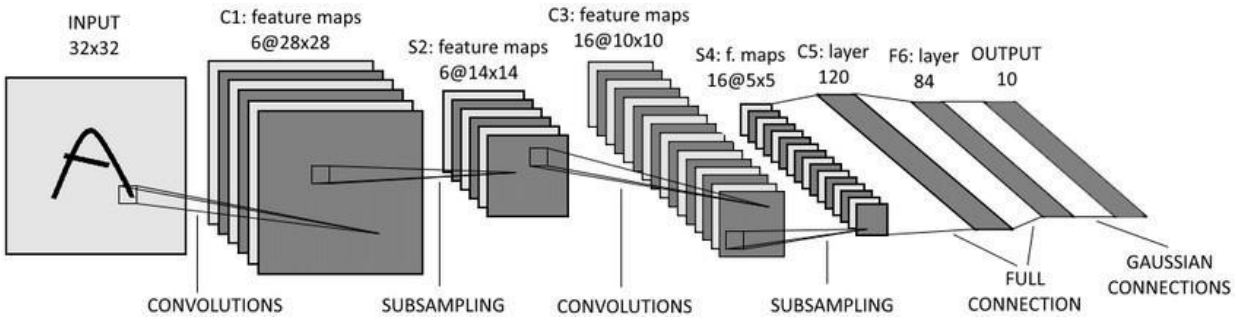
Shape 2-2 UAI and UAP Recognition Example

2.4. Convolutional Neural Networks

A convolutional neural network is a deep learning algorithm that can take an input image and separate various aspects/objects in the image. Convolutional neural networks are deep neural networks that are mainly used to classify images (e.g., name what they see), cluster by similarity (photo search), and perform object recognition in scenes.

The growth of data and reaching more meaningful information from the data necessitate optimization of feature estimations. As explained in previous, articles with a classical neural network model, the connections between neurons and layers and the learned parameters pose enormous computational difficulties. At this point, the importance of convolutional neural networks emerges.

Colorfull images consist of Red-Green-Blue (RGB) 3 channels. In this condition, the convolution operation is done for 3 channels. The channel number of the output signal is also calculated equally with the applied filter channel/number.



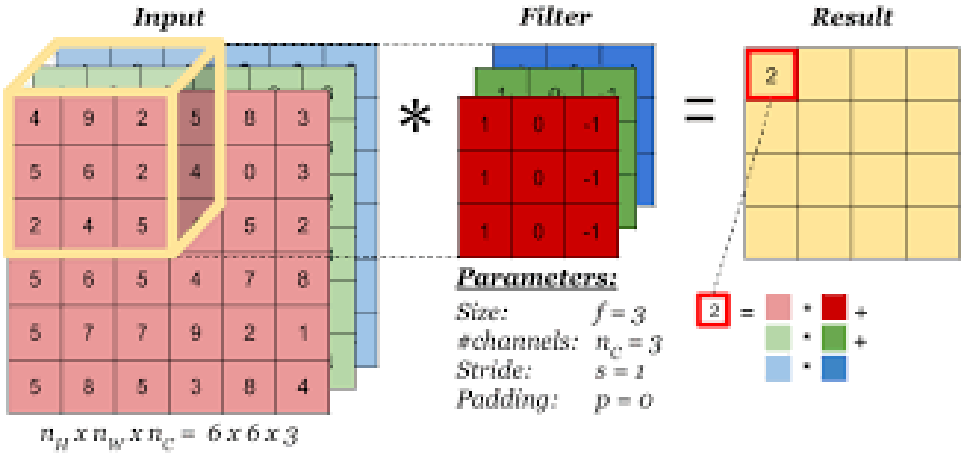
Shape 2.3. LENET

Shape 2-3 CNN

2.5. Convolution Operation

The symmetry of filter to be applied to the two-dimensional information with respect to the x and y axes is taken. All values are multiplied element by element in the matrix and the sum of all values is recorded as the corresponding element of the output matrix.

This is also called a cross-correlation relationship. This can be done simply when the input data (eg, image) is single-channel. However, the input data can be in different formats and number of channels. As a result, as a result of the convolution operation, the data turns into data with obvious properties and relatively easy to process.



Shape 2-4 Convolution Operation Example

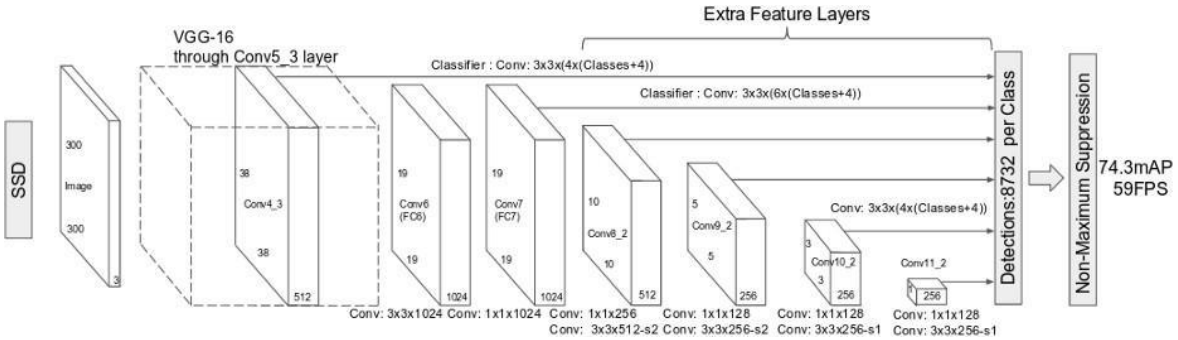
3. Tools and Methods Used

3.1. Method

3.1.1. Object Recognition Method

The use of the YOLOv-8 Psychic model was deemed appropriate in this project, and information about it is explained in this section. YOLO is an algorithm for object detection using convolutional neural networks. One of its biggest advantages is that it can detect objects very quickly and in one go. The reason for the speed of this algorithm is that it passes the entire image through a convolutional network at once, instead of searching for pixel-by-pixel objects in the image. The YOLO algorithm surrounds the objects it detects on the images with a box. Then, it divides the image given as input into NxN grids. Each region

decides whether there is an object in it and when the object exists, it decides whether the center point is in its own area. The region that decides that the object has a center point finds the class, height and width of that object and draws a box around that object. The use of the YOLOv-8 Psychic model was deemed appropriate in this project, and information about it is explained in this section. YOLO is an algorithm for object detection using convolutional neural networks. One of its biggest advantages is that it can detect objects very quickly and in one go. The reason for the speed of this algorithm is that it passes the entire image through a convolutional network at once, instead of searching for pixel-by-pixel objects in the image. The YOLO algorithm surrounds the objects it detects on the images with a box. Then, it divides the image given as input into NxN grids. Each region decides whether there is an object in it and when the object exists, it decides whether the center point is in its own area. The region that decides that the object has a center point finds the class, height and width of that object and draws a box around that object.



3.1.2.

Shape 3-1 SSD Structure

The first version of YOLO was released in 2015, while the last official version, the V8, was released in 2022. Networks under YOLO are divided into three branches: Large ,Medium, Tiny. The Large YOLO is better in terms of accuracy than other variations, but slower in terms of speed. Networks under YOLO are divided into three branches: Large, ' and . The Large YOLO is better in terms of accuracy than other variations, but slower in terms of speed. In Tiny YOLO, however, speed is set as a more important target and is designed for real-time systems. Normal YOLO, on the other hand, is between two variations in terms of both speed and accuracy Table3.1, Table 3.2.

Model	Input size	Train set	Test set	mAP	FPS
YOLOv1	448x448	VOC 2007+2012	VOC 2007	63.4%	45
Fast YOLOv1	448x448	VOC 2007+2012	VOC 2007	52.7%	155
YOLOv2	416x416	VOC 2007+2012	VOC 2007	76.8%	67
Tiny-YOLOv2	416x416	VOC 2007+2012	VOC 2007	57.1%	207
YOLOv2	608x608	COCO	COCO	48.1%	40
YOLOv3	608x608	COCO	COCO	57.9%	20

Table 3-1 YOLO V3 – V4 comparing

Method	FPS	mAP(%)
YOLOv3	49	52.5
YOLOv4	41	64.9
YOLOv3-tiny	277	30.5
YOLOv4-tiny	270	38.1
Proposed method	294	38.0

Table 3-2 Development by Yolo version

3.1.3. Car Recognition Method

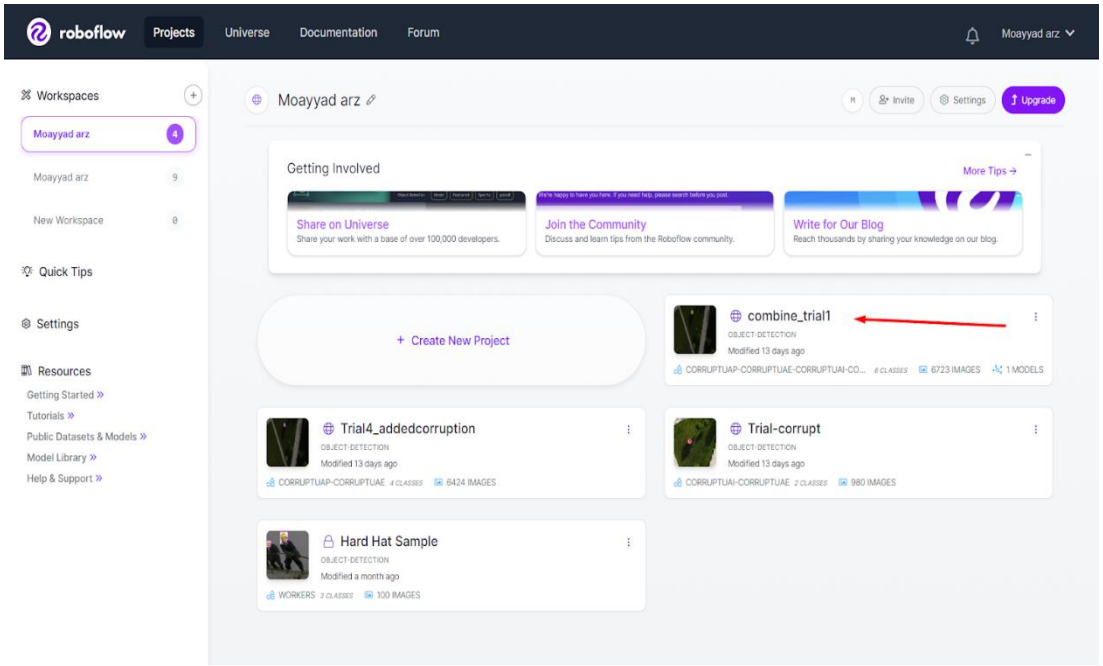
PKLot is the name of the cars recognition system proposed in the article Unified embedding for car recognition and clustering. Labeled cars achieved state-of-the-art results in many comparative car recognition datasets such as LFW and Youtube Car Database. They proposed an approach where it produces high-quality car mapping from images using deep learning architectures such as ZF-Net and Inception. He then used a method called triple loss as a loss function to train this architecture. PKLot is basically

based on detecting the face by extracting the feature vector from different cars and measuring the Euclidean distance between the vectors, ideally the Euclidean distance between two vectors belonging to the same car is 0.

3.2. Tools

3.2.1. VisDrone Recognition Dataset

Vehicle Detection dataset is a dataset of approximately 9 million images annotated with image-level labels, object bounding boxes, object segmentation masks, visual relations, and localized narratives. It contains a total of 16 million bounding boxes for 600 object classes over 1.9 million images, making it the largest dataset available with object location annotations. The boxes are largely hand drawn by professional explainers to ensure accuracy and consistency. Classes presented in VisDrone were also considered suitable as data sources as they matched the scope of the project.



Shape 3-2 COCO Example

3.2.2. Train platform

Within the scope of this project, it was deemed appropriate to use the Tensorflow platform offered by Google, as it offers models that are increasingly popular and will work with machine learning applications. According to the structure of the Tensorflow platform, the models are saved as .py files, then converted to the sub-platform presented under the name Tensorflow and saved as .json files. In this way, it becomes usable in embedded systems.

3.2.3. Other Train Araçları

- Since the images in UAVDT are in UAVDT format, they cannot be used directly in Tensorflow training and must be converted to VOC format. The necessary scripts were also found suitable to be coded in the Pycharm editor offered by Microsoft and made in Python language.
- In order to observe whether the training process is going well, it was deemed appropriate to use the Tensorboard offered in Tensorflow.

3.2.4. Development Editor of Artificial Boundary Model

It has been found appropriate to make the relevant NN Model in the Visual Studio editor, which is compatible with the Python language.

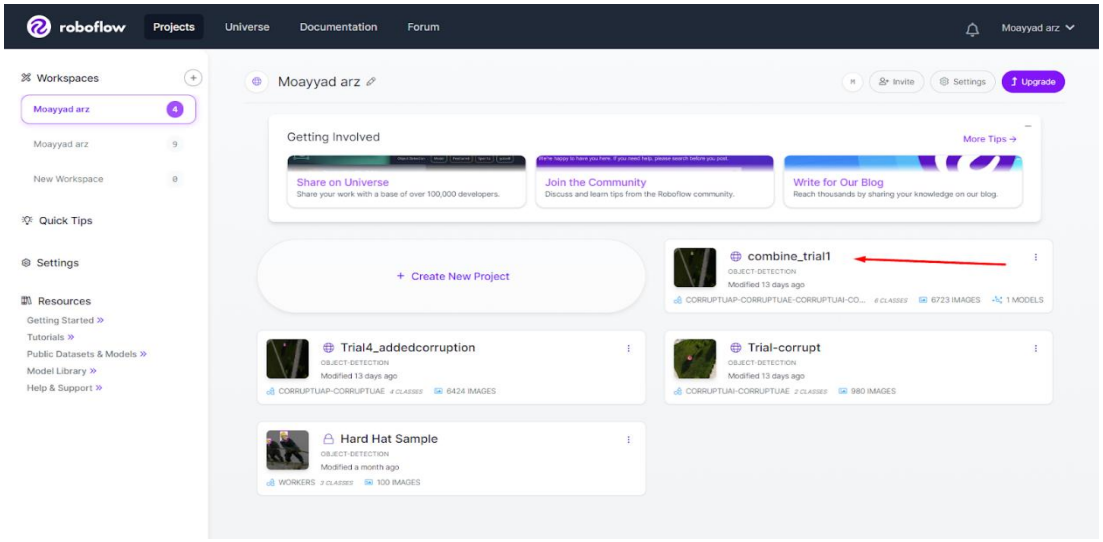
3.2.5. Version Control and Documentation

It was decided to use GitHub as a version control system because it is popular in this project and will make a more significant contribution, and GitBook as a document system because it works integrated with GitHub.

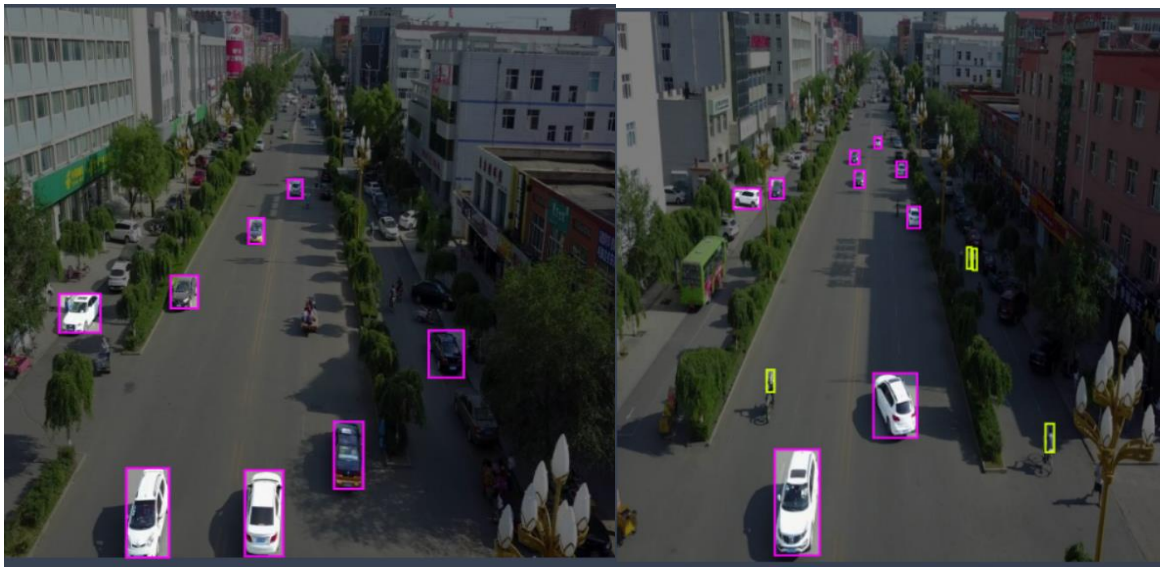
4. FINDINGS

4.1. Veri-Set labeling

Figure 4.1. As seen in the screenshot in , the welcome screen appears when Robo-flow is first opened, and then the main screen. As shown in the figure, there are several Data-Sets that we have labeled on the main screen. We trained our model using the data set we marked.

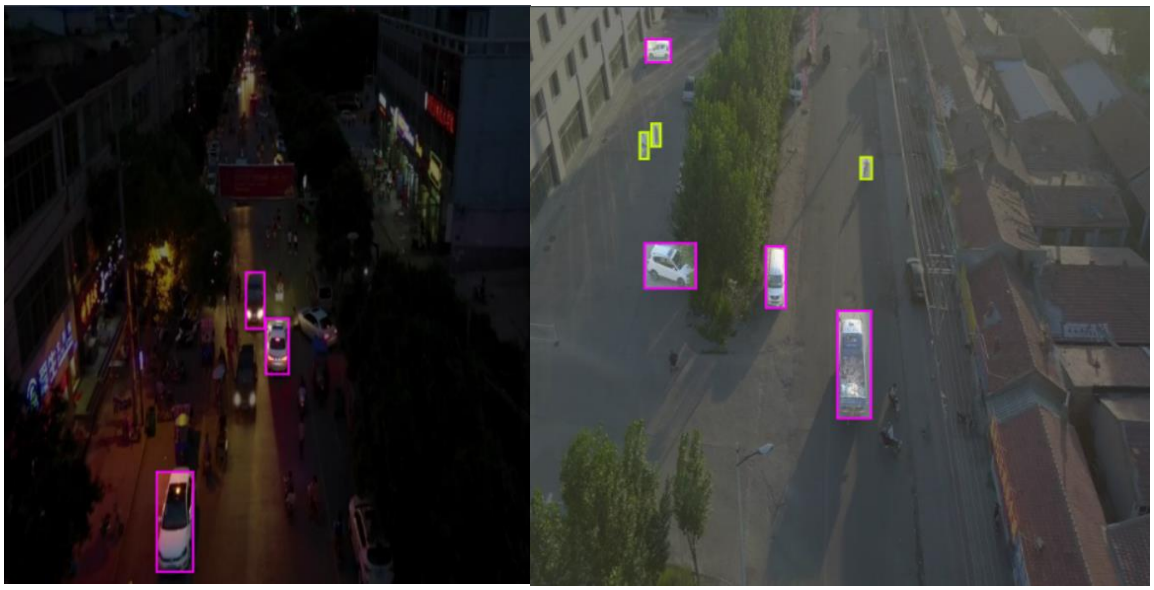


Shape 4-1 Preparing Dataset



Shape 4-2 labelling

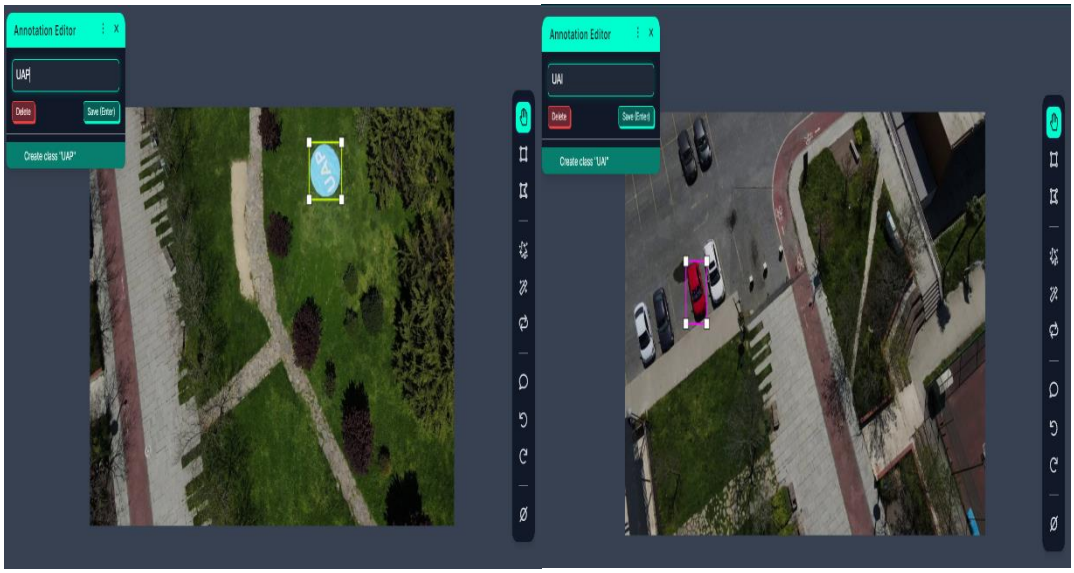
Shape 4-3 labelling



Shape 4-4 Labeling

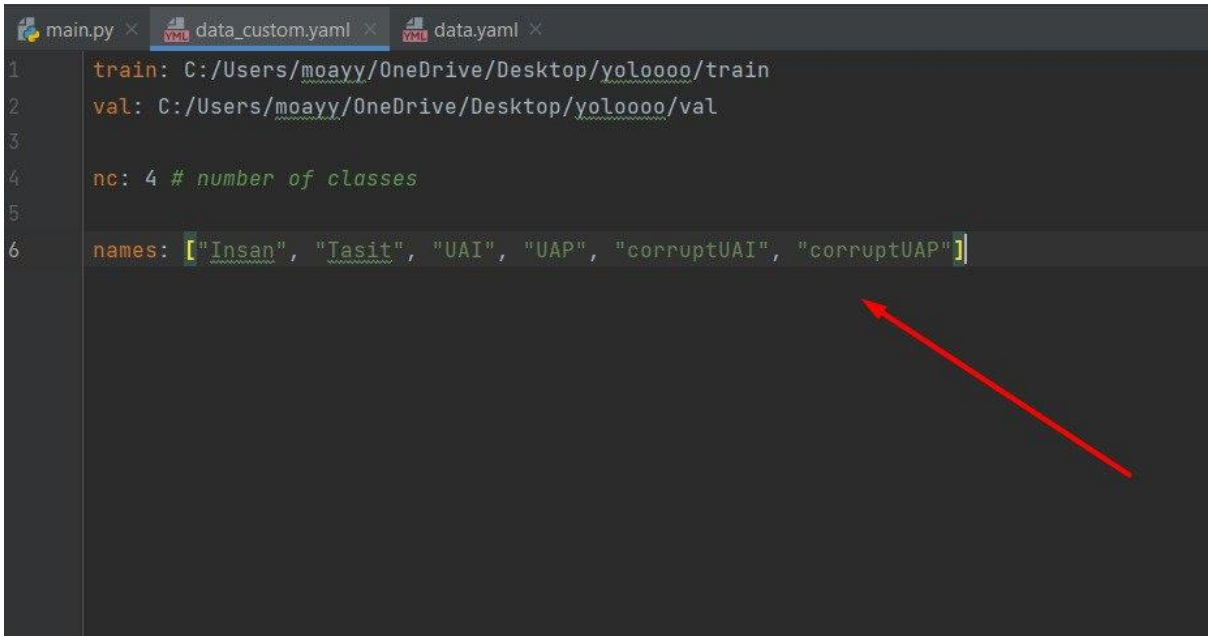
Shape 4-5 labeling

As seen in the images, the object finder using the tools offered by Roboflow searches for objects in the image and box defines and encloses the objects we find. The object, after entering its name, calculates the location of the object in the last box in the background as a vector and records its location in the weight file to be used for labeling later along with the name.

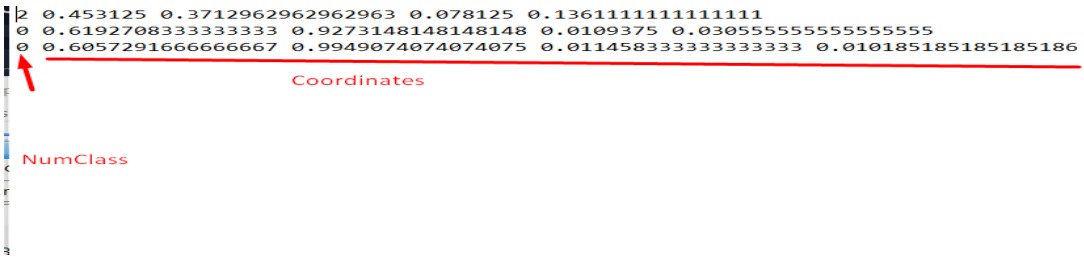


Shape 4-6 labeling

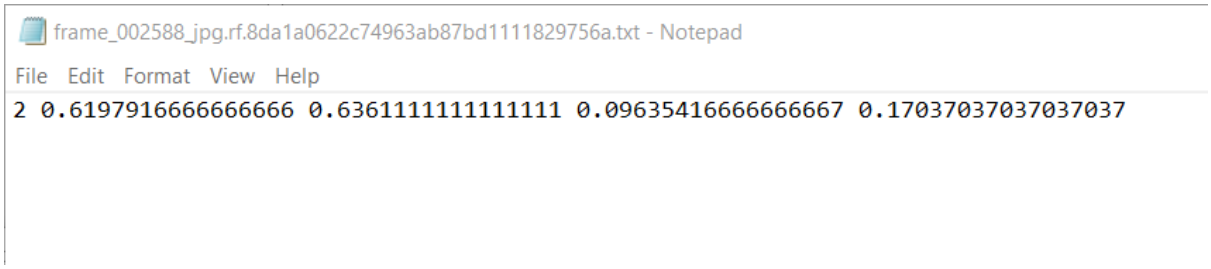
Shape 4-7 labeling



Shape 4-8 class names



Shape 4-9 box sizes



Shape 4-10 box sizes

4.2. Object Recognition

Within the scope of this project, an object recognition system was established using the YOLOv8 network structure.

4.2.1. Preparing Environment

In the official work of Yolov8, Python 10 version is given, but in order for the model to be used on the GPU platform, it must be downloaded to Python 7 format. Features of Generated Environment:

Python	3.7
Tensorflow	2.4.1
opencv-python	4.6.0
Torch	1.7.0
Torchvision	0.8.1
Pandas	1.1.4
Seaborn	0.11.0
scikit-learn	0.19.2
CVZONE	Latest

Table 4-1 Information List Environment

After a few attempts to prepare the training environment, libraries suitable for the project environment were selected according to the table above. In order to run the project correctly, the versions have been approved for use in the implementation of new versions of the project.

4.3. Training Veri-Seti:

Dataset collection is considered to be one of the most time-consuming operations in an AI project, as large-scale data are needed to train and achieve acceptable accuracy. For this reason, the process of selecting the type of data set containing images (UAE, UAI, Vehicle, Human) suitable for the project idea has been started. After collecting data from different sources, a good amount of data was obtained and added to the next data preprocessing process.

After the data set was collected, data cleaning was performed. In order to prevent the results that would deceive the training process and cause the required accuracy to not be achieved, data that were not suitable for the project and abnormal data were excluded from the necessary samples. Then, the tagging process was performed for the data set and the tagged data were divided into training, validation and test sets according to the size of the data. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to evaluate the model's performance on new data.

Shape 4-11 Training Results

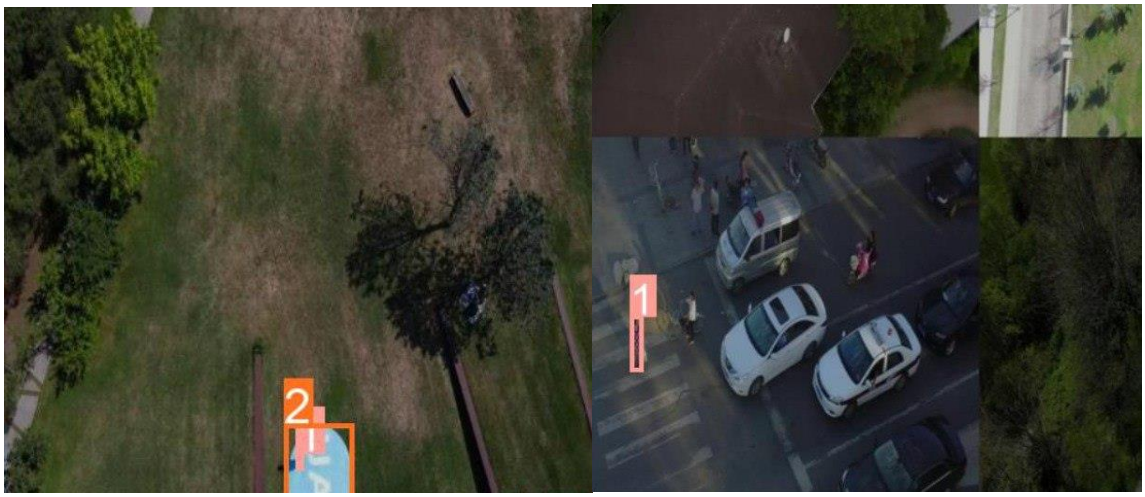
Shape 4-12 Training Results

Shpape 4-13 Training Results

Shape 4-14 Training Results

Shape 4-15 Training Results

Shape 4-16 Training Results



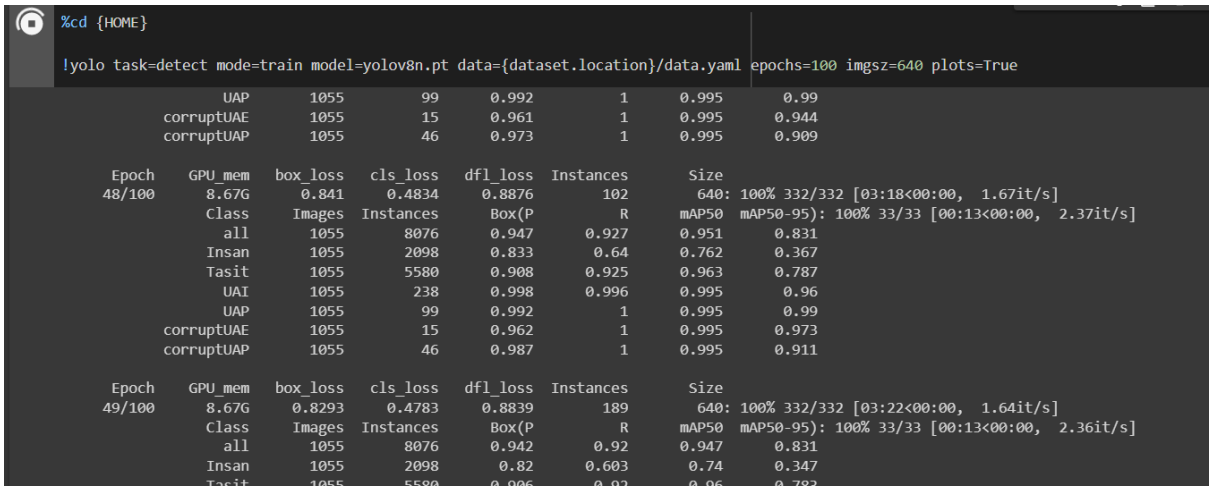
Shape 4-17 Training Results

Shape 4-18 Training Results

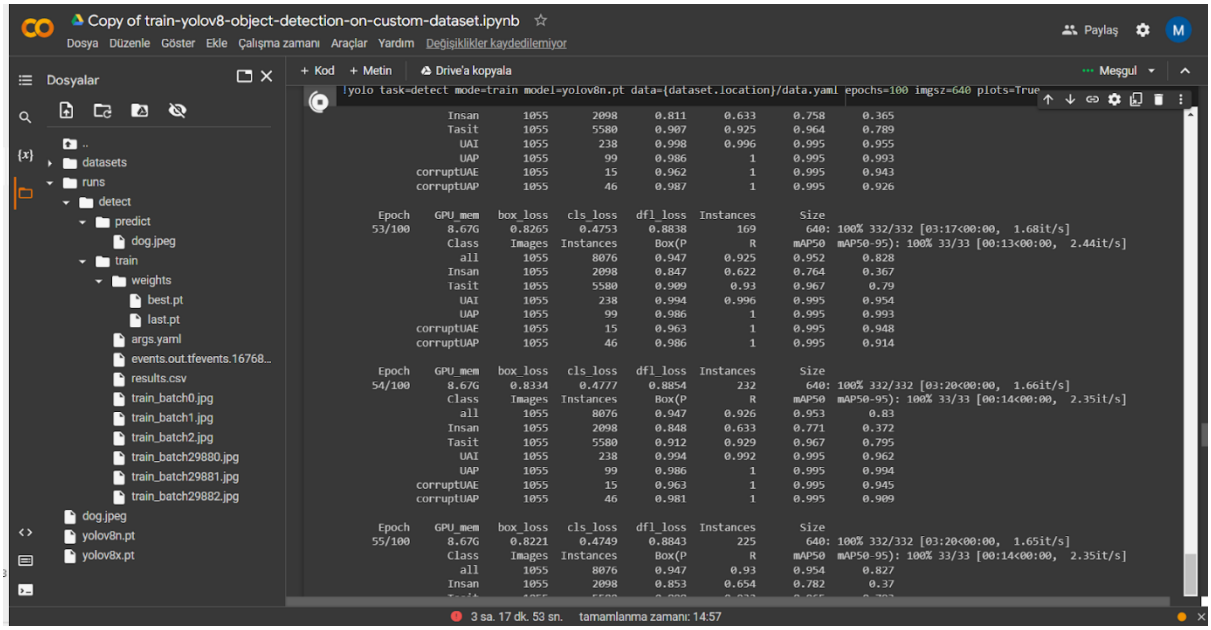
Shape 4-19 Training Results

4.4. Training Process

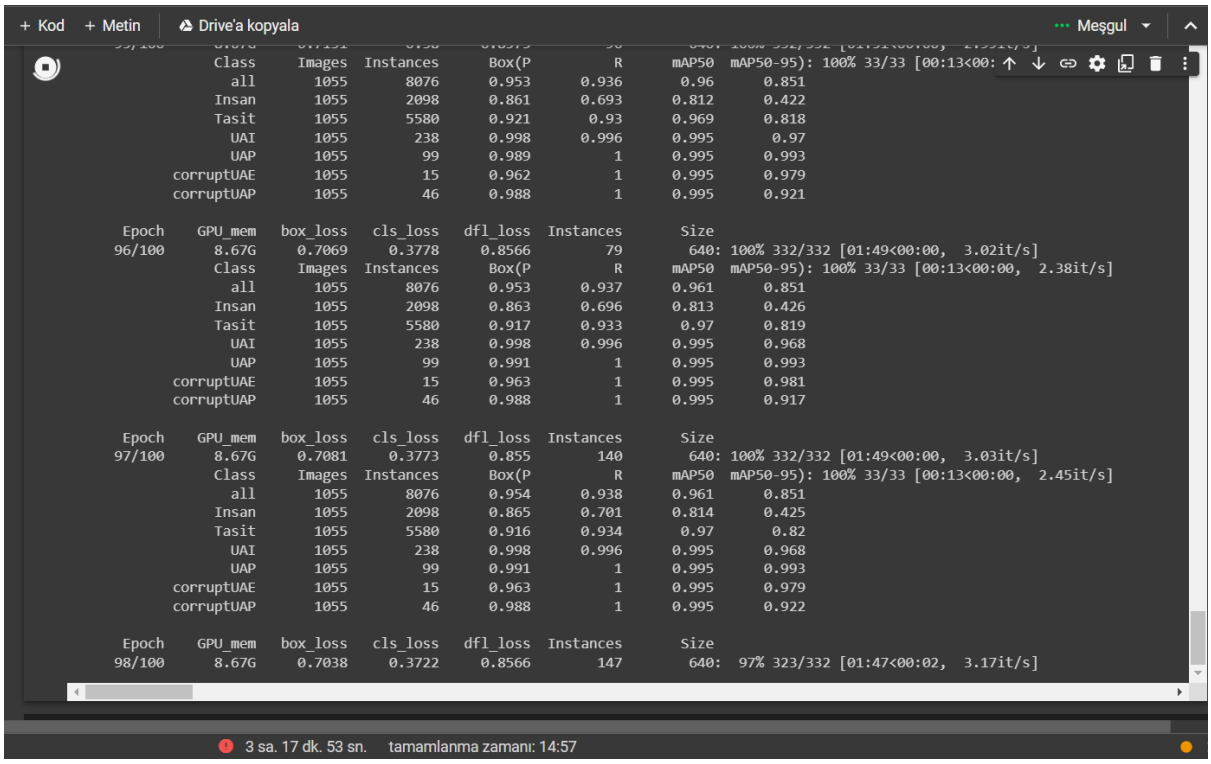
After the existing models were compared and classified according to their accuracy and validity, the structure of yolov8 was chosen to be used in the training process. The picture below shows a part of the training process performed on the pre-prepared data set. It has been trained several times using the model's data to obtain the required accuracy in accordance with the project objective described in the project plan presented earlier.



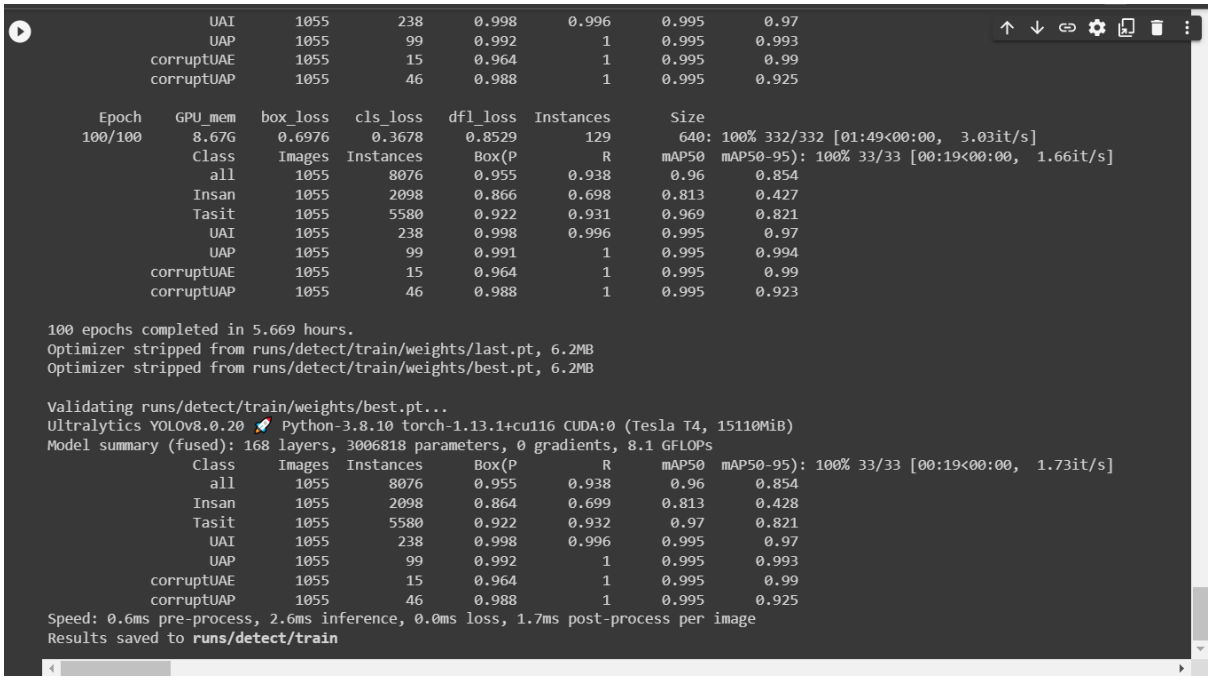
Shape 4-20 Training Process



Shape 4-21 Training Process

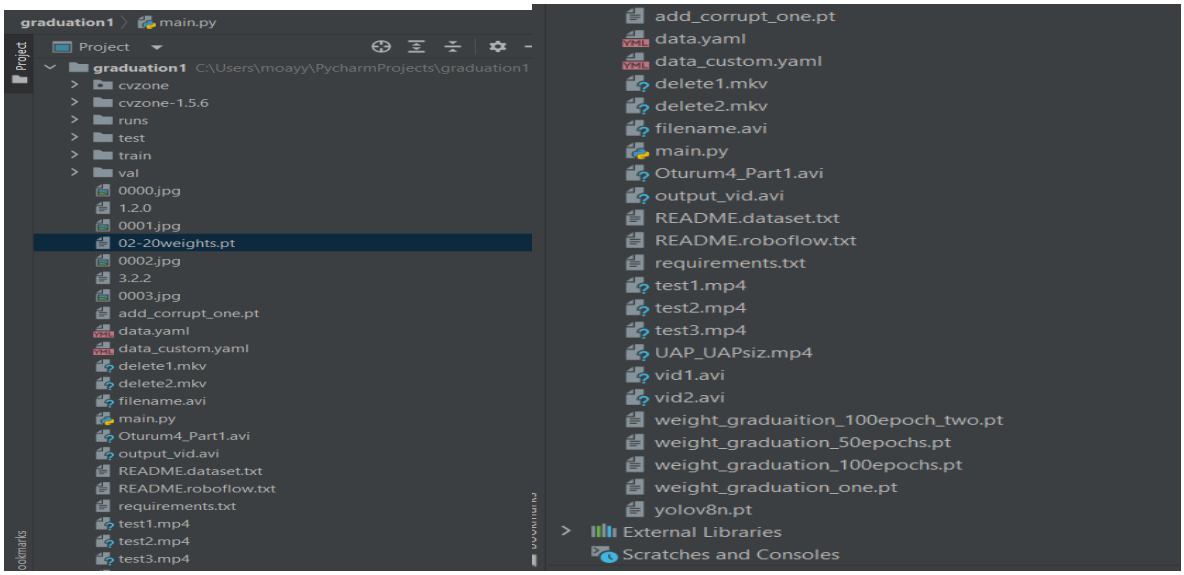


Shape 4-21 Training Process



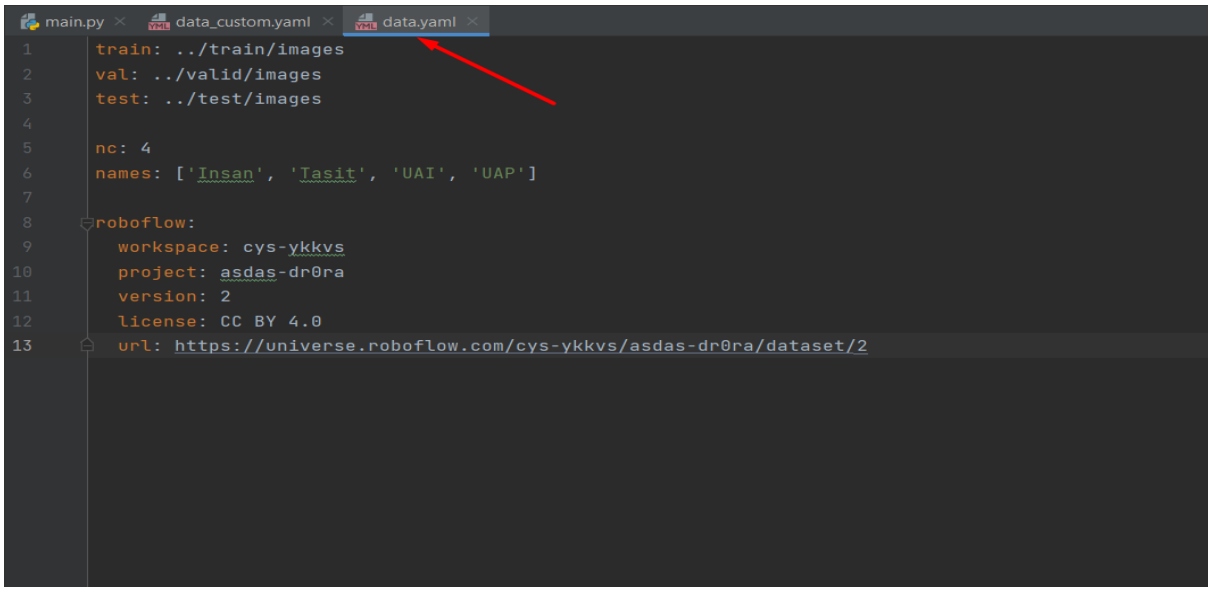
Shape 4-22 Training Process

After finishing the training process several times, the performance of the model was evaluated on the test set to determine how well it generalized to the new data. This step may include calculating various performance metrics such as accuracy, precision, recall, F1-score, and ROC curves.



Shape 4-23 Project Files

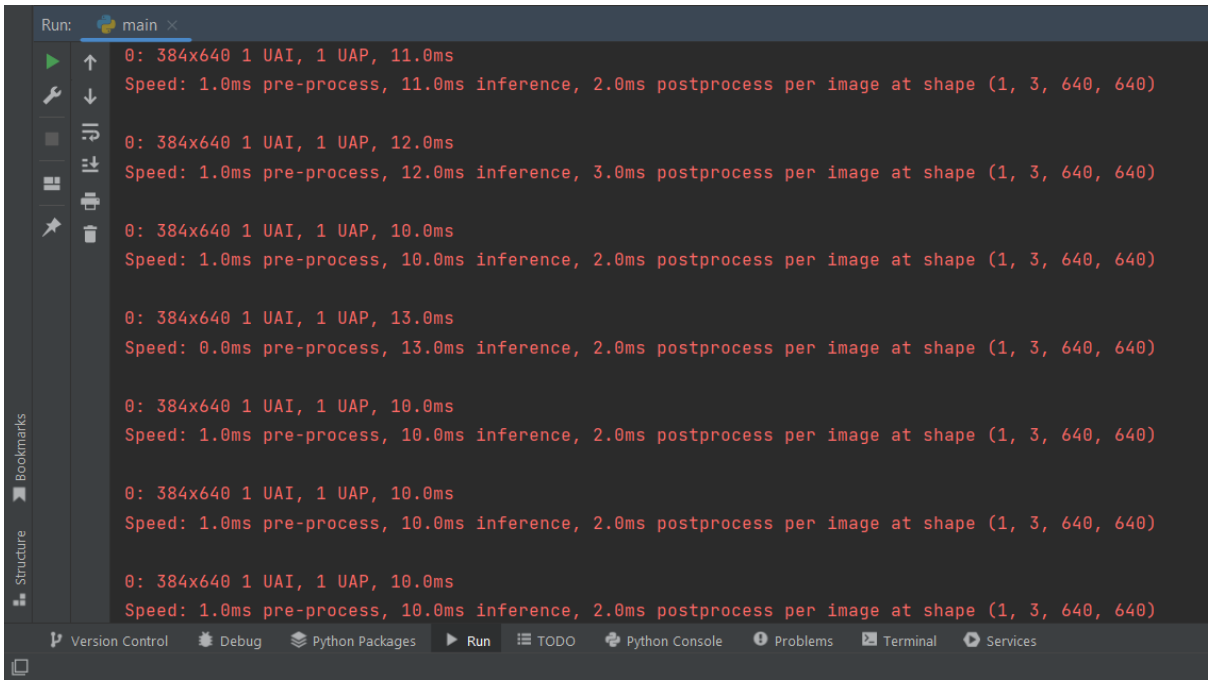
Shape 4-24 Project Files



Shape 4-25 Dataset link

In Figure 4.26 we see the connection of the dataset we are working on. We will use this data training, validation and testing.

Shape 4-26 Benchmarks



Shape 4-27 Benchmarks

Figure 4.27 has an example of inference, at the top you can clearly see the features of the GPU we use to run it,

In the second line, the number of layers and parameters used to make this inference can be seen.

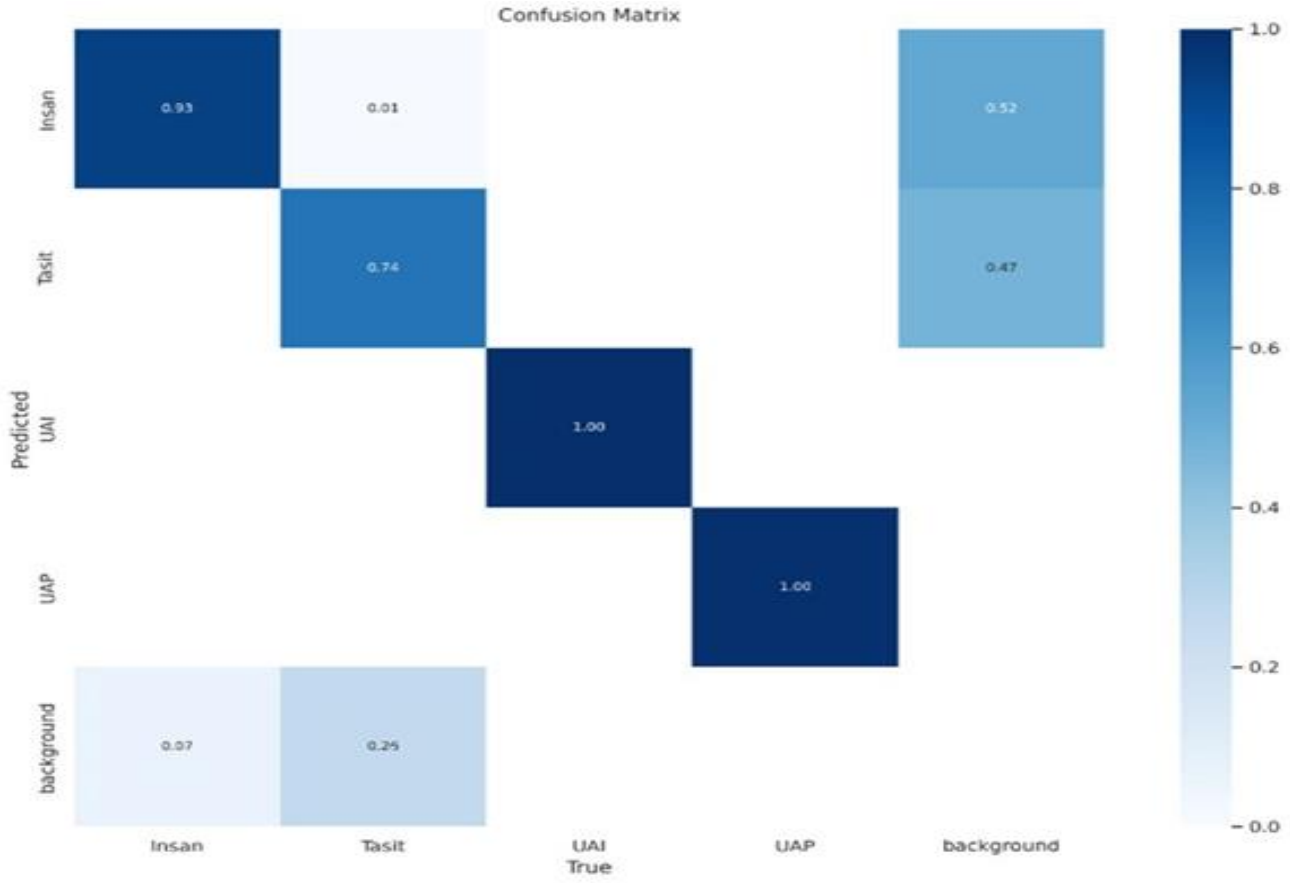
The rest of the lines showing what was removed in each frame

Example:

The first Frame is described, which has:

1 UAI and 1 UAP and inference rate (detects Frame at 11.0ms which is very impressive and incredible compared to CPU) etc.

4.5. Confusion Matrix :



Shape 428- Confusion Matrix

A graph showing how our model handles different classes.

Person:

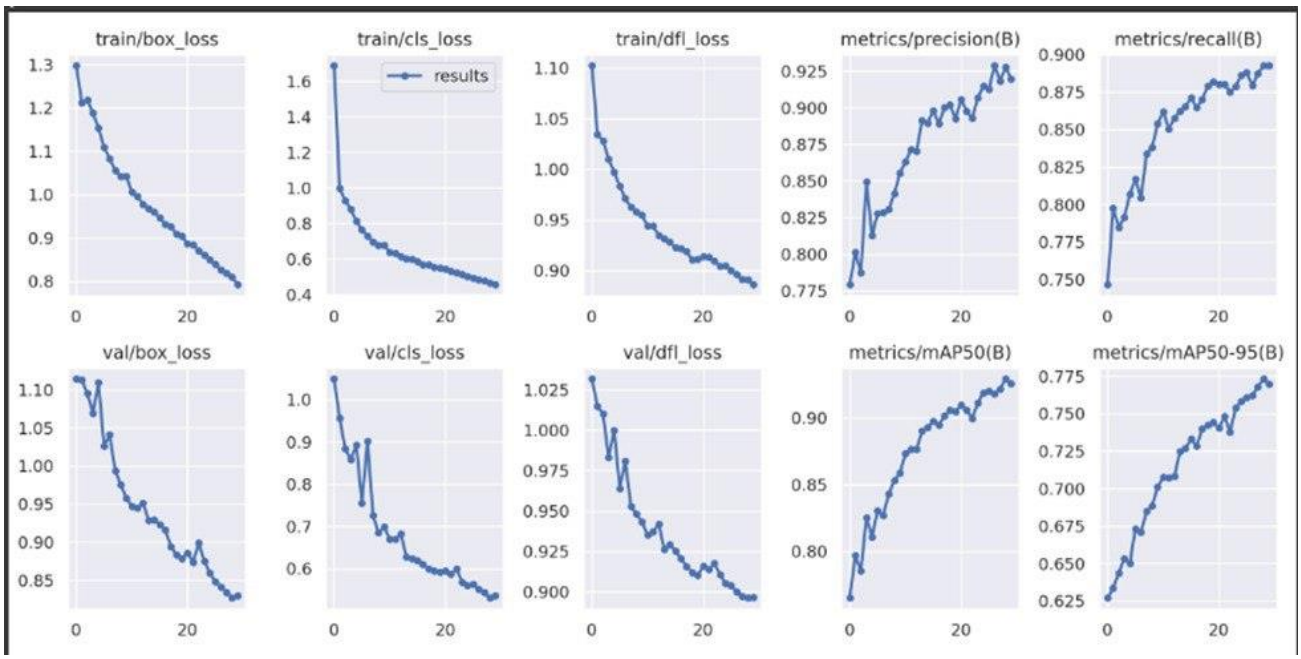
- It is detected with 93% accuracy and classified as Human.
- 7% of objects were there but not detected

Car :

It is a small object and difficult for the model to detect.

- It is detected with 74% accuracy and classified as Taxit.
- 25% of the objects were there but not detected
- 1% incorrectly detected and classified as Human, and so on for other classes.

4.6. Key Matrix



Shape 4-29 Key Matrix

Above is a graph showing the key matrix tracked by Yolov8 object detection.

Training and validation The most important graphs dealing with box loss and class loss for the Data-Set. It is clear how accurate the model's behavior is. The model is correct and converging. The curve in these graphs is still pretty steep. In other words, more of these models are still available.

The flat part of the diagram means that we are training the model a lot and wasting time on our GPU without getting much results. Our key matrix is not evolving.

4.7. Validation Set:

```
/content
2023-05-07 11:10:20.342389: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3006428 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /content/datasets/ASDAS-2/valid/labels.cache... 991 images, 0 backgrounds, 0 corrupt: 100% 991/991 [00:00<?, ?it/
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 62/62 [00:25<00:00,  2.39it/s]
                   all        991       8015       0.93      0.892      0.929      0.774
                 Insan        991       5580      0.914      0.917      0.965      0.789
                 Tasit        991       2098      0.825      0.656      0.762      0.371
                   UAI        991         99      0.987          1      0.995      0.993
                   UAP        991        238      0.996      0.996      0.995      0.945
Speed: 1.0ms pre-process, 4.5ms inference, 0.0ms loss, 2.4ms post-process per image
```

Şekil 4-30 Doğrulama Kümesi sonucu

The validation set is the previously unused portion of the data set. This part is used to test the dataset. Above is the actual mAP (Mean Average Precision) we care about.

5. Discussion and Conclusion

With the spread of artificial intelligence-based devices, companies want to use these technologies as they gain great benefits such as optimum use of time thanks to the use of these technologies. Considering the traditional processes carried out by the manual method, the weaknesses in these traditional methods based on the use of the human element have been identified and applications that do what the human factor does more cheaply and in a way have been developed. This ensures optimum use of resources.

In this study, applications made for companies are examined. An NN Model has been created that contributes to the development of companies. In the first part of the study, an artificial intelligence model is trained that recognizes the objects frequently seen by the Camera in the flying car. The selected model YOLOV8 was rewarded for accuracy and given importance to speed, for real-time operation and offering the desired interactivity. At the end of the training, an attempt was made to obtain a mAP value that was close to the results presented in YOLOV8's article and could be considered sufficient.

As a result, thanks to this study, a structure was established that ensures the continuous improvement of the object detection process with the Flying Car, especially during the landing process. In addition, since the project is open source, it also makes a contribution to the software field.

6. Future Studies

In future studies, it is planned to create a copy of the project with the ability to detect people and other objects from above, and also to generalize the idea and make use of visual perception to benefit in the field. In addition, it is aimed to transport things such as cargo within the city between different locations given by entering the system in advance, and for this, it is considered to provide the opportunity to connect to Google maps.

Resources

[1]. Zeyer, A., Merboldt, A., Schlüter, R. and Ney, H., 2020. A new training pipeline for an improved neural transducer. *arXiv preprint arXiv:2005.09319*.

[2]. Safadinho, D.; Ramos, J.; Ribeiro, R.; Filipe, V.; Barroso, J.; Pereira, A. UAV Landing Using Computer Vision Techniques for Human Detection. *Sensors* **2020**, *20*, 613. https://doi.org/10.3390/s20030613

[3]. A. Womg, M. J. Shafiee, F. Li and B. Chwyl, "Tiny SSD: A Tiny Single-Shot Detection Deep Convolutional Neural Network for Real-Time Embedded Object Detection," *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 95-101, doi: 10.1109/CRV.2018.00023.

[4]. Zhu R, Zhang S, Wang X, Wen L, Shi H, Bo L, Mei T. ScratchDet: Training single-shot object detectors from scratch. InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019 (pp. 2268-2277).

[5]. Zhu, Pengfei, et al. "Visdrone-det2018: The vision meets drone object detection in image challenge results." *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[6]. Long, Xiang, et al. "PP-YOLO: An effective and efficient implementation of object detector." *arXiv preprint arXiv:2007.12099* (2020).

[7]. TERVEN, Juan; CORDOVA-ESPARZA, Diana. A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. *arXiv preprint arXiv:2304.00501*, 2023.

[8]. Luque, A., Carrasco, A., Martín, A. and de Las Heras, A., 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recognition, 91, pp.216-231.

[9]. Gomes Correia, A., Cortez, P., Tinoco, J. and Marques, R., 2013. Artificial intelligence applications in transportation geotechnics. *Geotechnical and Geological Engineering*, *31*, pp.861-879.

[10]. Nichols, J.A., Herbert Chan, H.W. and Baker, M.A., 2019. Machine learning: applications of artificial intelligence to imaging and diagnosis. *Biophysical reviews*, *11*, pp.111-118.