# GENERATING SYNTHETIC DATA WITH GAME ENGINES FOR DEEP LEARNING APPLICATIONS

# DERİN ÖĞRENME UYGULAMALARI İÇİN OYUN MOTORLARI İLE SENTETİK VERİ ÜRETME

**KAMİL ANIL ÖZFUTTU**

**PROF. DR. HAŞMET GÜRÇAY**

**Supervisor**

Submitted to Informatics Institute of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Animation and Game Technologies

January 2022

This work named **"Generating Synthetic Data with Game Engines for Deep Learning Applications"** by **Kamil Anıl Özfuttu** has been approved as a thesis for the Degree of **MASTER OF SCIENCE IN COMPUTER ANIMATION AND GAME TECHNOLOGIES** by the below mentioned Examining Committee Members.

Prof. Dr. Haşmet Gürçay
Supervisor .....

Assoc. Prof. Dr. Mustafa Sert
Member .....

Assoc. Prof. Dr. Burkay Genç
Member .....

This thesis has been approved as a thesis for the Degree of **MASTER OF SCIENCE IN COMPUTER ANIMATION AND GAME TECHNOLOGIES** by Board of Directors of the Institute of Informatics.

Prof. Dr. Arif Altun
Director of the Institute of Informatics
Graduate School of Science and Engineering

*To my family..... .*

# ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Informatics of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules.
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using others works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- and any part of this thesis has not been presented as another thesis study at this or any other university.

.../.../2022

KAMİL ANIL ÖZFUTTU

# YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe Üniversitesi'ne verdiğimi bildiririm. Bu izinle Üniversite'ye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversite'ye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan "**Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge**" kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H.Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- ☐ Enstitü / Fakulte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- ☐ Enstitü / Fakulte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir.
- ☐ Tezim ile ilgili gizlilik kararı verilmiştir.

.../.../2022

KAMİL ANIL ÖZFUTTU

# ABSTRACT

## GENERATING SYNTHETIC DATA WITH GAME ENGINES FOR DEEP LEARNING APPLICATIONS

**Kamil Anıl Özfuttu**

**Master of Science, Computer Animation and Game Technologies**

**Supervisor: Prof. Dr. Haşmet Gürçay**

**Jan 2022, 75 pages**

In this thesis, we present a novel method that aims to generate labeled synthetic data for use in training drone detection deep learning models. Modern deep learning methods provide state-of-the-art performance on object detection problems and this performance is increasing thanks to the development of deep learning architectures day by day. The biggest obstacle to this development is the need for large amounts of labeled data for deep learning methods to be successful. Accessible datasets, on the other hand, generally do not contain sufficient data, have incorrect labels and bias. Datasets are traditionally made manually by humans. This human-made process is quite time-consuming and prone to human error. Although it is very difficult to find domain-specific data sets, it is almost impossible to reach data sets for specific problems One of the areas where there is a data shortage is drone detection. Drones are becoming more and more common day by day due to the development of drone technology and decreasing hardware costs. These objects cause security and privacy problems, so the detection of these objects gains great importance. Since drones can be found in any environment due to their nature, a data set with high data amount and variety covering all conditions were needed to train a model that will detect small or hard to visible drones. To solve this data shortage in drone detection, a novel method that can generate synthetic data

with game engines is presented. The created method can generate training-ready, labeled images in a randomized manner using two-dimensional backgrounds and three-dimensional drone models. Also, an ablation study was carried out to optimize the synthetic data generated within the scope of this thesis, and several trainings were carried out to compare it with the actual data performance. The developed method can quickly generate synthetic images with photo-realistic labels, and the models trained with these datasets perform performance close to the models trained with real data.

**Keywords:** synthetic data, deep learning, object detection, game engines

# ÖZET

## DERİN ÖĞRENME UYGULAMALARI İÇİN OYUN MOTORLARI İLE SENTETİK VERİ ÜRETME

**Kamil Anıl Özfuttu**

**Yüksek Lisans, Bilgisayar Animasyonu ve Oyun Teknolojileri**

**Danışman: Prof. Dr. Haşmet Gürçay**

**Ocak 2022, 75 sayfa**

Bu tezde drone tespiti derin öğrenme modellerinin eğitiminde kullanmak üzere etiketli sentetik veri üretmeyi amaçlayan bir yöntem sunuyoruz. Modern derin öğrenme metodları obje tespiti problemlerin oldukça başarılı performans sağlamakta ve gün geçtikçe gelişen derin öğrenme mimarileri sayesinde bu performans daha da artmaktadır. Bu gelişimin önündeki en büyük engel ise derin öğrenme yöntemlerinin başarılı olabilmesi için yüksek miktarda etiketli veriye olan ihtiyacıdır. Erişime açık veri setleri genellikle yeterli miktarda veri içermemekte, hatalı etiketlere ve yanlış eğilimlere sahip olmaktadır. Veri setleri geleneksel olarak insanlar tarafından manuel olarak yapılmaktadır. İnsan eliyle yapılan bu işlem oldukça zaman alıcı ve insan hatasına açık olmaktadır.Bu yüzden alan spesifik veri setlerini bulmak oldukça zor olmakla birlikte bazı problemlere ait veri setlerine ulaşmak neredeyse imkansızdır. Veri sıkıntısının olduğu alanlardan biri de drone tespitidir. Gelişen drone teknolojisi ve azalan donanım amliyetlerinden dolayı dronelar gün geçtikçe yaygınlaşmaktadır. Popüler olarak kullanılan bu cihazlar güvenlik ve gizlilik problemlerine yol açmakta bu yüzden bu objelerin tespiti büyük önem kazanmaktadır. Dronelar yapısı gereği her ortamda bulunabileceğinden dolayı küçük veya düşük görünürülükteki droneları tespit edecek bir model eğitmek için tüm

şartları kapsayan, yüksek veri miktarına ve çeşidine sahip bir veri setine ihtiyaç duyulmaktadır. Bu çalışmada drone tespitindeki bu veri sıkıntısını çözmek için oyun motorlarıyla sentetik veri üretebilecek bir yöntem sunulmaktadır. Oluşturulan yöntem iki boyutlu arkaplanlar ve üç boyutlu drone modellerini kullanarak randomize bir şekilde eğitime hazır, etiketli görüntüler oluşturabilmektedir. Bu tez kapsamında oluşturulan sentetik verileri optimize etmek için bir ablasyon çalışması yapılmış, gerçek veri performansı ile karşılaştırmak için ise bir takım eğitimler gerçekleştirilmiştir. Geliştirilen yöntem hızlı bir şekilde foto realistik etiketli sentetik görüntüler üretebilmekte olup, bu veri setleri ile eğitilen modeller gerçek veri ile eğitilen modellere yakın performans sergilemekte, gerçek veri ile birlikte kullanıldığında ise gerçek modelin performansını arttırmaktadır.

**Anahtar Kelimeler:** sentetik veri, derin öğrenme, obje tespiti, oyun motorları

# *ACKNOWLEDGEMENTS*

I would like to express my gratitude to my advisor Prof. Dr. Haşmet Gürçay, who guided me with his knowledge and experience,

I would like to thank Sinan Onur Altınuç, Fatih Çağatay Akyön, Oğulcan Eryüksel and entire OBSS AI team for their support and contribution to this work.

I would like to also show my sincere appreciation to my thesis commitee members, Assoc. Prof. Dr. Burkay Genç and Assoc. Prof. Dr. Mustafa Sert for taking their time to review and providing insightful comments.

Finally, I would like to thank my family, my mother Ayşegül Özfuttu, my father Ahmet Özfuttu and my brother İbrahim Onur Özfuttu for their unconditional love and endless support

# CONTENTS

**Page**

ABSTRACT ..... i
ÖZET ..... iii
ACKNOWLEDGEMENTS ..... v
CONTENTS ..... vi
FIGURES ..... ix
TABLES ..... x
ABBREVIATIONS ..... xi

1. INTRODUCTION ..... 1
1.1. Overview ..... 1
1.2. The Method ..... 2
1.3. Research Questions ..... 4
2. LITERATURE REVIEW ..... 5
3. EXISTING UAV DATASETS ..... 8
3.1. MAV-VID Dataset ..... 8
3.2. Anti-UAV Dataset ..... 9
3.3. Drone vs. Bird Dataset ..... 9
3.4. Dataset Statistics ..... 9
4. CREATING THE GENERATION ENVIRONMENT ..... 12
4.1. Selecting a Game Engine ..... 12
4.2. Decision between 3D and 2D Environments ..... 14
4.3. Gathering Resources ..... 15
5. GENERATION METHOD ..... 22
5.1. Generation Pipeline ..... 22
5.2. Generation Parameters ..... 24
5.3. Post Processing and Blending ..... 27
5.4. System Output and Performance ..... 31

6. EXPERIMENTS & RESULTS ..... 33
6.1. Experimental Setup ..... 33
6.1.1. Object Detection Framework ..... 33
6.1.2. Training Parameters ..... 34
6.1.3. Metrics ..... 35
6.1.4. Dataset Usage ..... 35
6.2. Ablation Study ..... 36
6.2.1. Base Dataset ..... 37
6.2.2. Smaller Drones Feature ..... 38
6.2.3. Without Propeller Feature ..... 39
6.2.4. Random Drone Color Feature ..... 40
6.2.5. Random Light Color Feature ..... 42
6.2.6. Hue Shift Feature ..... 43
6.2.7. Multiple Drones Feature ..... 45
6.2.8. Distractor Objects Feature ..... 46
6.2.9. Bird Objects Feature ..... 48
6.2.10. Noise Feature ..... 49
6.2.11. Lens Distortion Feature ..... 50
6.2.12. Motion Blur Feature ..... 52
6.2.13. Depth of Field Feature ..... 53
6.2.14. Gaussian Blur Feature ..... 54
6.2.15. Inference Blur Feature ..... 55
6.2.16. Adaptive Blur Feature ..... 56
6.2.17. Best Dataset ..... 57
6.3. Training Results ..... 58
6.3.1. Impact of Amount of Real Data on Model Performance ..... 59
6.3.2. Impact of Amount of Synthetic Data on Model Performance ..... 61
6.3.3. Final Trainings ..... 61
7. CONCLUSION ..... 66
REFERENCES ..... 69

# FIGURES

**Page**


3.1. Statistics of three datasets [1] .UAV location refers to bounding box center and size is the ratio of the object to the image. ..... 10

4.1. A commissioned study conducted by Forrester Consulting on behalf of Unity, October 2019 [2] ..... 13
4.2. Fully 3D rendered image at right, 3D drone model rendered on 2D background at left ..... 15
4.3. Six 3D drone models gathered as representation of different drone models and sizes ..... 16
4.4. Samples from urban backgrounds dataset. ..... 17
4.5. Samples from sky backgrounds dataset. ..... 18
4.6. Samples from nature backgrounds dataset. ..... 19
4.7. Samples from misc backgrounds dataset. ..... 20

5.1. The pipeline of synthetic data generation. Pipeline takes generation parameters as input and gives a output file containing images and annotations.Each step in the loop is repeated at each iteration. ..... 23
5.2. The raw rendering result of the scene as a result of randomizations is above, and the result after post processing and blending is below. ..... 28
5.3. Comparison of four different blending technique. ..... 29
5.4. Comparison of synthetically generated images (left) and real images from Drone vs. Bird dataset (right) ..... 31

6.1. Comparison of EfficentDet and YOLOv5 models [3]. YOLOv5X6 slower than other models but gives best mAP results. ..... 33
6.2. Calculation of mean average precision (mAP) ..... 35
6.3. Illustration of precision and recall ..... 38
6.4. Training graphs of smaller drones and base dataset ..... 38

6.5. Rendering with without propeller feature at left, base at right ..... 39
6.6. Training graphs of without propeller and base dataset ..... 40
6.7. Rendering with randomly colored drones ..... 40
6.8. Training graphs of random drone color and base dataset ..... 41
6.9. Rendering with randomly colored light source ..... 42
6.10. Training graphs of random light color and base dataset ..... 43
6.11. Rendering with hue shift feature ..... 43
6.12. Training graphs of random light color and base dataset ..... 44
6.13. Training graphs of multiple drones and base dataset ..... 45
6.14. Rendering with distractor objects ..... 46
6.15. Training graphs of distractor and base dataset ..... 47
6.16. Rendering with different 3D bird models ..... 48
6.17. Training graphs of bird objects and base dataset ..... 49
6.18. Training graphs of noise and base dataset ..... 49
6.19. Rendering with lens distortion effect ..... 50
6.20. Training graphs of lens distortion and base dataset ..... 51
6.21. Training graphs of motion blur and base dataset ..... 52
6.22. Training graphs of depth of field and base dataset ..... 53
6.23. Training graphs of gaussian blur and base dataset ..... 54
6.24. Training graphs of inference blur and base dataset ..... 55
6.25. Training graphs of adaptive blur and base dataset ..... 56
6.26. Training graphs of best and base dataset ..... 57
6.27. mAP scores of models trained with different amount of ”Real Data” ..... 59
6.28. Validation loss graph.With examples containing more than 25k data, the validation loss in training starts to increase after a certain epoch. ..... 60
6.29. mAP scores of models trained with different amount of ”Synthetic Data” ..... 61
6.30. Comparison of trainings with only real and only synthetic data ..... 63
6.31. Comparison of trainings with mixed datasets ..... 64
6.32. Comparison of trainings with synthetic models finetuned with real data ..... 65

# TABLES

6.1. Comparison table of mAP, precision and recall scores of features. Features with ”*” symbols used in the ”Best” dataset. ..... 37
6.2. Experiments with different proportions of real and synthetic data ..... 62

# ABBREVIATIONS

**UAV** Unmanned Air Vehicle

**mAP** Mean Average Precision

**CNN** Convolutional Neural Network

# 1. INTRODUCTION

## 1.1. Overview

The ability of machine learning techniques to detect objects improves with each passing day as deep learning technology advances. Object detection models based on CNN architecture gained popularity, especially after AlexNet [4] demonstrated the success of deep convolutional neural networks (CNN) [5–7]. The amount of labeled data, on the other hand, is the most significant impediment to using a cutting-edge deep learning model. Although deep learning architectures are developing day by day, the performance of the trained models cannot reach the desired level due to the mentioned data bottleneck. As a solution to this problem, there has been a shift from a model-centric approach to a data-centric approach.

The data-centric approach is aimed to improve the data used in training by keeping the model architectures constant. The datasets used in training deep learning models are created by humans and obtained manually. Thus, traditional data collection and labeling are labor-intensive, expensive, and more importantly, prone to human error. The data-centric approach aims to investigate issues such as automatic labeling, finding faulty labels, and increasing the accuracy of labels to find solutions to these problems. As a different solution to these problems, creating data synthetically instead of correcting real data is becoming an increasingly accepted solution.

Training deep learning models with synthetically generated data is a very appealing approach. Because synthetically generated data not only eliminates the burden of labeling but also allows creating highly accurate labeled data sets. In addition, since it does not have any limit on the amount of data created, it theoretically has the possibility to generate an unlimited number of data. Most importantly, it can be the only solution for some deep learning problems, as it allows creating data sets in areas where real data collection is difficult or impossible.

Computer vision, which is one of the areas where deep learning is frequently used, and object detection, which is one of the most popular problems in this field, are one of the research areas where the mentioned problems are encountered most frequently. Object detection is carried out by training models using labeled data sets consisting of the object or objects to be detected. The fact that each object is precisely labeled on the images has a great impact on the performance of the trained object detection models. In some problems, it is very difficult to label the data collected manually. One of these problems is the detection of unmanned aerial vehicles (UAV). Although drones have been used for defensive purposes in the past, their use has expanded to areas such as weather surveillance, traffic surveillance, agriculture and photography as they have become accessible to everyone. Furthermore, small quad-copters are now widely available, opening up previously unexplored possibilities while raising safety, privacy, and security concerns. It is very difficult to find a labeled data set of these objects, due to the wide area of use and the fact that drone objects can be found in almost every environment and all shapes. In addition, drone objects can be easily confused with objects such as birds and airplanes, so the training dataset should include these and similar objects.

There are some datasets consisting of real images created for drone detection. These datasets will be analyzed in the following sections. The biggest problem with these datasets is that they are all in video format. To train the model, each frame in these videos is transformed into an image and thus converted into a data set. Since the difference between the frames is very small, there are very few images in the data set that will provide different information to the model. As a result, models trained with these datasets have low generalization capacity and show poor performance when they are used in another domain. In line with the idea that a synthetically created dataset can produce many and more informational images, it is thought that synthetic data will solve these problems.

### 1.2. The Method

Our main goal in this study is to develop an infrastructure that can produce large numbers of high-quality data in a fast and controllable way to be used in drone object detection. In this

direction, platforms that allow synthetic data generation has been investigated. Among these platforms, the game engines are the most suitable for our problem.

Although game engines are mainly used for game development, their usage areas have expanded to various fields such as film production, architecture, engineering and simulation, thanks to their customizable and flexible infrastructures. Game engines seem very attractive to use for synthetic data generation due to their real-time rendering features, programmatic infrastructure and popularity. In addition, game engines have started to give results close to programs with offline rendering features, thanks to their developing rendering capabilities. In this way, very fast and high-quality images can be created using game engines.

Game engines are capable of creating high-quality 3D environments. Therefore, creating synthetic data by simulating photo-realistic 3D environments is the first idea that comes to mind. Although generating synthetic data with this method is successful in some studies, it is not a generally accepted solution since each synthetic data problem requires different solutions. In the drone detection problem, the variety of environments in which drones can be located is quite large, and creating a photo-realistic 3D environment in this variety requires a high amount of money, time and human resources. Therefore, in the developed method, it was seen that it was more appropriate to use 3D drone models placed on 2D backgrounds instead of 3D environments. In this direction, background images with different themes and 3D drone models with different characteristics were obtained to be used in the created infrastructure.

The infrastructure that produces labeled data for drone object detection consists of four basic parts; scene, randomizer, post-processing and annotator. The system is based on iteration, and a scene is created with the background image, drone models, and light properties according to the values from the randomizer per iteration. Then the annotations of the objects in this scene are calculated and the scene is rendered. A labeled image is obtained by adding various post-processing effects to this rendered image. As a result of all these processes, a data set file containing the reference and annotation information of all images are created. This developed system can produce images with photo-realistic labels very quickly.

In the experiments, it has been seen that the models trained with the datasets created by the presented method perform comparably to the ones trained with real data. In addition, it has been understood that synthetic data can increase the performance of the model trained with real data when used with real data.

As a result, a system has been developed that can generate labeled data sets that can be used in drone detection model training, quickly and with high quality. This system not only provides solutions to most of the problems encountered in deep learning methods but also enables the creation of models that cannot be trained with real data.

## 1.3. Research Questions

This thesis seeks answers and solutions to the following questions;

- Can the object detection model be trained with synthetically generated data?
- What kind of infrastructure can be established for synthetic data generation?
- How can the domain gap between real data and synthetic data be bridged?

# 2. LITERATURE REVIEW

The data problem encountered in deep learning methods has led researchers to look for alternative methods. To overcome the problem of insufficient amount of data, enriching and duplicating the available data is one of the first tried methods. The process of creating variations of the available data as a result of various modifications is called ”Augmentation” [8, 9]. Although augmentation can be applied to any type of data, it is mostly used in computer vision tasks. In addition, to transform operations such as rotation, flip and shifting, many variations of existing data can be created by adding effects such as noise and blur to images. The model trained as a result of these processes becomes more robust because it sees different variations. The problem with the augmentation approach is that although these images contain different variations, they add little new information to the model. Therefore, although the augmentation method slightly increased the model performance, it could not be a definitive solution to the data problem. Developing rendering technologies and hardware make it possible to create photo-realistic images through computers. Thanks to the new possibilities, it is possible to create the images synthetically instead of augmenting the images. Therefore, training deep learning models with synthetically created labeled images has become one of the popular research topics today.

When it comes to creating images synthetically in the computer environment, the first thing that comes to mind is 3D environments. Generating synthetic data by creating realistic 3D environments is a very attractive approach. Because while having absolute control over the environment, it is possible to obtain results close to real images. In this direction, methods using photo-realistic rendering [10–16] and complex scene compositions [10, 12–21] have been developed to generate synthetic data. Although these methods have a successful approach, it is a very time-consuming and expensive process to create realistic 3D environments specific to each targeted problem. Also, some methods [18] try to overcome this problem by using realistic open world computer games, but the ethics of the method used is controversial. Especially in some problems, it may be more tempting to collect and label real data instead of creating photo-realistic and complex 3D environments. In addition, it is

very difficult to obtain the necessary equipment and human resources to create these environments. Therefore, methods aiming to use photo-realistic rendering reduce the advantages of synthetic data over real data.

Another approach is the field of synthetic data prefers to increase the variation in the images created instead of creating realistic images. These methods use unrealistic scene compositions, lighting and textures to increase variation. ”Domain Randomization”, first introduced in [22], argues that the real world will appear to the model as another variation if sufficient randomization is done in the generated data. Works adopting this method [11, 22–25] argue that with sufficient randomization, the difference between real and synthetic images, also known as ”Domain Gap”, will be overcome, thus eliminating the need for photo-realism. Although this method is limited and can be used in certain domains, it is very difficult to apply in subjects such as drone detection, where the domain is almost the whole world. This can be seen in study [25]. In this study, a drone detection data set was generated with domain randomization. However, due to the mentioned problems, the object detection model trained with the synthetic data set performed very poorly compared to our method. In addition, in many of the problems where domain randomization is applied, it is aimed to train a system that will detect not a single object but usually multiple objects related to each other.

Another approach, similar to our method, uses 2D background images instead of 3D environments. Using real-world images as a background is seen as a successful method to reduce the domain gap. Two different methods were used in studies adopting this approach.

The first method aimed to create synthetic images by placing 2D images of objects on 2D backgrounds. Studies using this approach [13, 26] have suggested that synthetic datasets can be easily created with the help of simple cut and paste operation and various post-processing effects. Although this simple approach is interesting, using 2D images of objects limits the use of this method. First of all, due to the two-dimensional qualities of the objects, the images of these objects under different perspectives and light conditions cannot be represented. This reduces the variation in the data and limits the model performance. Secondly, the images of the objects to be placed in the background should be obtained and processed. This

process imposes a similar workload to real data collection and labeling and removes one of the attractive aspects of synthetic data: the ability to create data in hard-to-obtain domains.

The second method uses 2D backgrounds and 3D models, similar to our approach. This approach eliminates the variation and data collection problems mentioned and creates great flexibility in the generated data. The biggest problem encountered in studies using this method [27, 28] is the localization problem. In cases where the object to be placed on the background is dependent on a certain environment and object, the glass on the table can be given as an example, both the backgrounds suitable for the specific domain are required and the objects must be placed on the background according to certain rules. In the drone detection problem, since the drone objects are independent of the environment, there is no obstacle in both the selection of the background and the placement of the object in the background. Therefore, this hybrid method is seen as the most suitable method for our problem.

# 3. EXISTING UAV DATASETS

Since deep learning methods require large amounts of data, access to datasets appropriate for the problem and domain has become even more important. Although there are many datasets on popular topics like human and car detection, statistical data, and classification, the amount of datasets targeting problems such as UAV detection, where data collection and labeling is difficult, is very small.

Since drones are objects independent of the environment, the problem of drone detection includes all kinds of environmental conditions by its nature. Therefore, an ideal drone detection dataset should cover all possible light, ambient and environmental conditions. In addition, the dataset should represent all drone sizes, predominantly small drones, and include different drones and objects such as birds and airplanes that can be confused with drones.

The camera and features used while obtaining the images to create the data set are also important. The resolution of the collected images is an important factor, especially when detecting small drones. In addition, the camera's features such as exposure and focus should be taken into account when creating a data set, since it creates artifacts such as blur and motion blur on the object to be detected. Finally, the fact that the camera is moving or fixed also creates motion blur in the object and also creates difficulties for the tracking algorithms.

Datasets that mostly fit these conditions, Drone vs. Bird [29], Multi-rotor Aerial Vehicle VID (MAV-VID) [30] and Anti-UAV [31] datasets were obtained for use in training a deep learning object detection model and validating these models. These datasets consist of videos of different lengths and compositions and include only the "drone" class.

## 3.1. MAV-VID Dataset

The MAV-VID dataset consists of videos of a single drone model shot from different angles and environments. These videos are in 1920x1080 full HD resolution and were shot with static and motion cameras. The dataset consists of 40,200 images, including 29,500 for

training and 10,700 for validation. The average drone size in the data set was calculated as 0.60% of the image size. Although the different environments contribute to the generalization of the trained models, the fact that it includes a single drone model has a negative effect.

### 3.2. Anti-UAV Dataset

Anti-UAV dataset consists of 186,494 images with different lighting, environment and drone models. Unlike other datasets, it contains both RGB and IR images. The data set with drone size at an average ratio of 0.50% does not have the desired features because it has low contrast images and IR images are not suitable for the problem we are targeting.

### 3.3. Drone vs. Bird Dataset

This dataset, published under The International Workshop on Small-Drone Surveillance Detection and Counteraction techniques of IEEE AVSS 2020, mainly includes small drones (0.10% of the image size) and consists of 104,760 images. This dataset was created for use in Bird Challenge and aims to reduce false positives and confusion of small drones with objects such as birds and airplanes. Likewise, the fact that the drones have a more homogeneous distribution throughout the image makes it the most ideal data set that can be used to solve the targeted problem.

### 3.4. Dataset Statistics

Statistics of location,size and image histograms of these data sets are shown in Fig. 3.1.

Drone vs. The Bird dataset has the most diverse location distribution. For the performance of the trained model, the distribution of the drone's location in the image and their representation on all kinds of backgrounds are of great importance. As a result of the fact that the

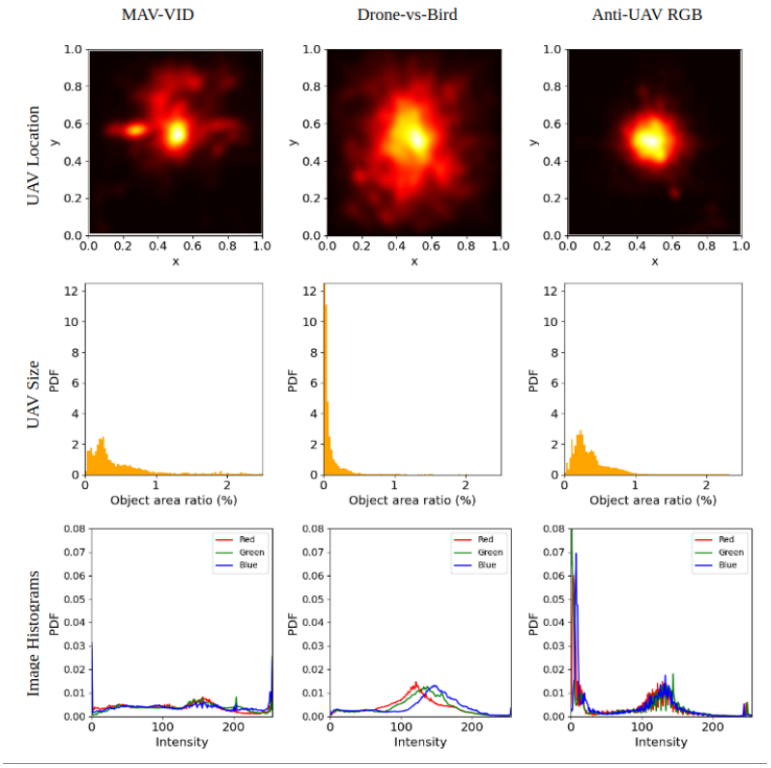


FIGURE 3.1. Statistics of three datasets [1] .UAV location refers to bounding box center and size is the ratio of the object to the image.

MAV-VID and Anti-UAV data sets mostly contain images taken with a motion camera, the drones show density on the horizontal axis and in the center.

Likewise, Drone vs. Bird contains mostly small drone footage. Large and medium-sized drones are easier to find, while small drones are more difficult to detect. Because as drones get smaller, they are more easily confused with objects such as birds and airplanes. Therefore, datasets containing small drones are of greater importance for the problem we are trying to solve. On the other hand, MAV-VID includes larger drones than Anti-UAV and both datasets have a more homogeneous drone size distribution. When the image histograms are examined, it is seen that the Anti-UAV dataset has low contrast. Images with low contrast make object detection difficult. That's why MAV-VID and Drone vs. Bird datasets are a better choice for use in model training.

When the three data sets mentioned are examined, Drone vs. Bird dataset is seen as the most

appropriate option to use for the targeted problem, considering its features such as giving weight to small drones, having a more homogeneous distribution throughout the image and including different drones. All experiments and evaluations mentioned as ”Real Data” in the trainings to be held in the Sec. 6.3. were made using this data set.

# 4. CREATING THE GENERATION ENVIRONMENT

In this section, we will discuss how the synthetic data creation infrastructure is designed and how the synthetic data creation pipeline works. First of all, we will examine the features of the game engines on which game engine it would be logical to build this infrastructure. Next, we will discuss the pros and cons of using 2D backgrounds and 3D environments when generating synthetic data for the drone recognition problem. After explaining how materials such as drone models and backgrounds suitable for 3D and 2D environment design are selected and used, we will explain in detail how we designed a pipeline on the selected game engine, the parametric structure of this pipeline, the importance of the parameters and how they are adjusted. It will be seen that the designed method has a very understandable and easily applicable structure thanks to the game engines, 2D backgrounds and parametric structure.

## 4.1. Selecting a Game Engine

Although game engines were created to develop 2D and 3D games for various platforms, their usage areas have expanded considerably with the development of this technology and ease of access. Game engines are beginning to be accepted by industries such as engineering, architecture, construction and production, thanks to their flexible structures and real-time rendering capabilities [2]. It is also becoming a frequently used tool in research fields such as artificial intelligence, 3D simulation and autonomy.

When the game engines in the market are examined, the success of the rendering and programming infrastructure, the ease of use and the two game engines most popular by the developers, Unreal Engine and Unity 3D, come out. Unreal Engine tries to provide more realistic visuals, but it has a more cumbersome and difficult programming infrastructure. In addition, fewer resources are provided by both developers and users in the field of synthetic data and simulation. On the other hand, although less realistic visuals are obtained in Unity,

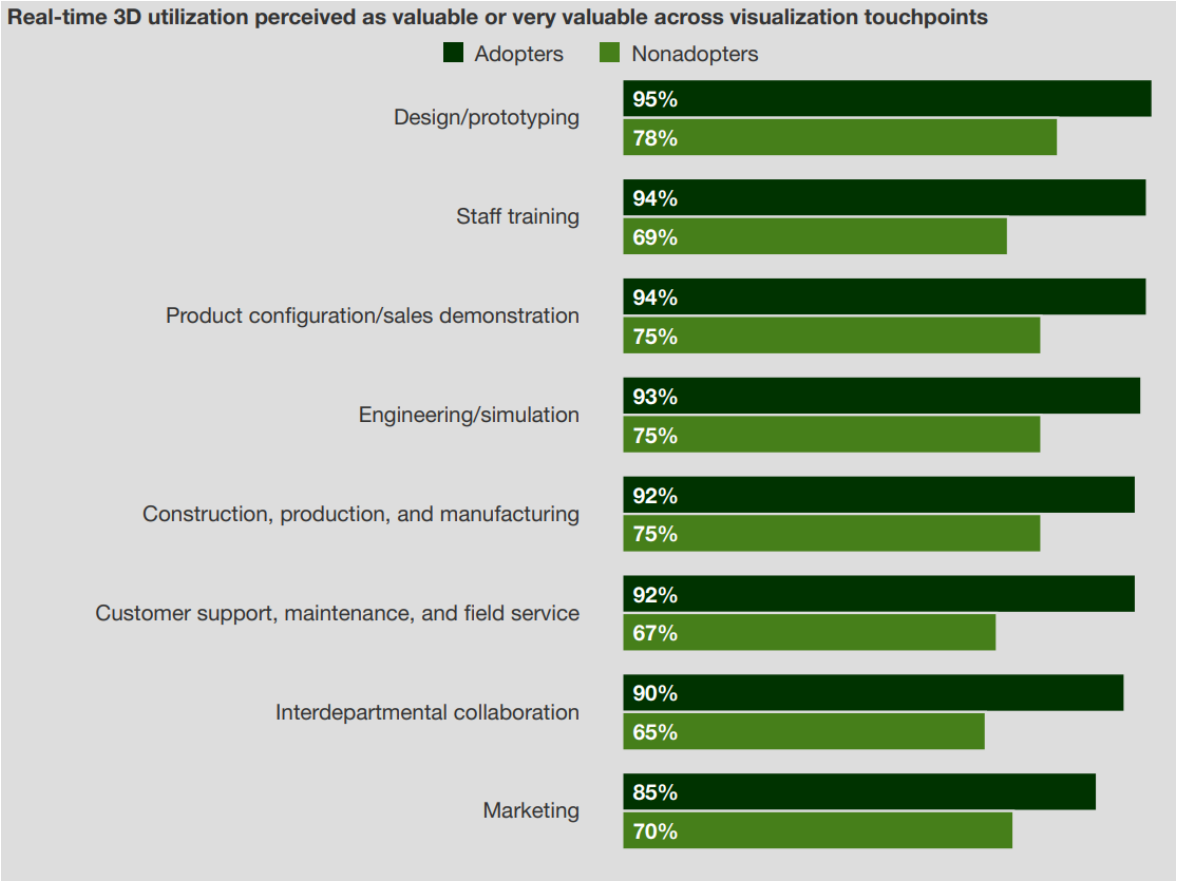


FIGURE 4.1. A commissioned study conducted by Forrester Consulting on behalf of Unity, October 2019 [2]

many tools are provided by both Unity and developers in the field of synthetic data and simulation. When these features are examined, Unity provides an ideal environment to establish the synthetic data infrastructure that we aim to develop for 3D drone recognition.

Apart from game engines, there are solutions with offline rendering features. Although programs such as Blender, 3DS Max, and Maya produce much more realistic outputs, they are not an ideal tool for synthetic data exploration, which requires a lot of experimentation and repetition due to very long render times, lack of programming infrastructure, and difficult customization. However, it is one of the options that can be used when it is desired to produce a data set consisting of completely 3D environments.

## 4.2. Decision between 3D and 2D Environments

The environments created for the production of synthetic data differ according to the characteristics of each problem domain, so each synthetic data method requires a unique solution. Therefore, the problem to be solved should be considered in detail and the environment should be designed accordingly.

When it comes to the production of synthetic data in game engines, the first thing that comes to mind is completely 3D environments similar to those in games. Although creating 3D scenes may seem an attractive solution at first, complex and rich environments created for games are created by very large teams and companies. Designing environments of this depth is a very difficult problem for small teams and requires many experts in their fields to provide the desired realism. Ready-made environments, on the other hand, do not provide a suitable environment for every problem and remain limited. In addition, while it is relatively easy to create environments with limited domains such as the detection of objects on the desktop and the detection of market products, it is quite difficult to create an environment suitable for large domains that cover all possible scenarios such as human and car detection and drone recognition.

Some methods and companies generate synthetic data with fully 3D environments. As mentioned, most of these methods aim to solve problems with limited domains and solutions that choose to create large and complex environments are usually tried by large companies. Although there are methods that aim to use open and wide world games such as Grand Theft Auto V (GTA V) instead of creating an environment it is a matter of debate how legal these methods are.

a) Full 3D Environment

a) 2D Background / 3D Model

FIGURE 4.2. Fully 3D rendered image at right, 3D drone model rendered on 2D background at left

As mentioned before, every synthetic data problem requires a specific solution, so not every solution has to be entirely composed of 3D environments. As a suggestion to this situation, it is considered to use a hybrid method using 2D backgrounds and 3D objects. 2D background images provide complexity and distribution that cannot be found in 3D environments, thus helping to reduce the difference between domains. In addition, the selected images cover a wide variety of environments, thus enabling the trained model to perform much better. The 3D models placed in front of these background images allow the object to be detected to be represented in every desired variation. In addition, the presented method can give more realistic results than 3D renders, as can be seen in Fig. 4.2.

## 4.3. Gathering Resources

Obtaining the background images and 3D drone models mentioned earlier is of great importance for the developed method and constitutes one of the most time-consuming parts. Since

the selected backgrounds and 3d models directly affect the performance of the trained model, there are factors to be considered in the selection of these materials.

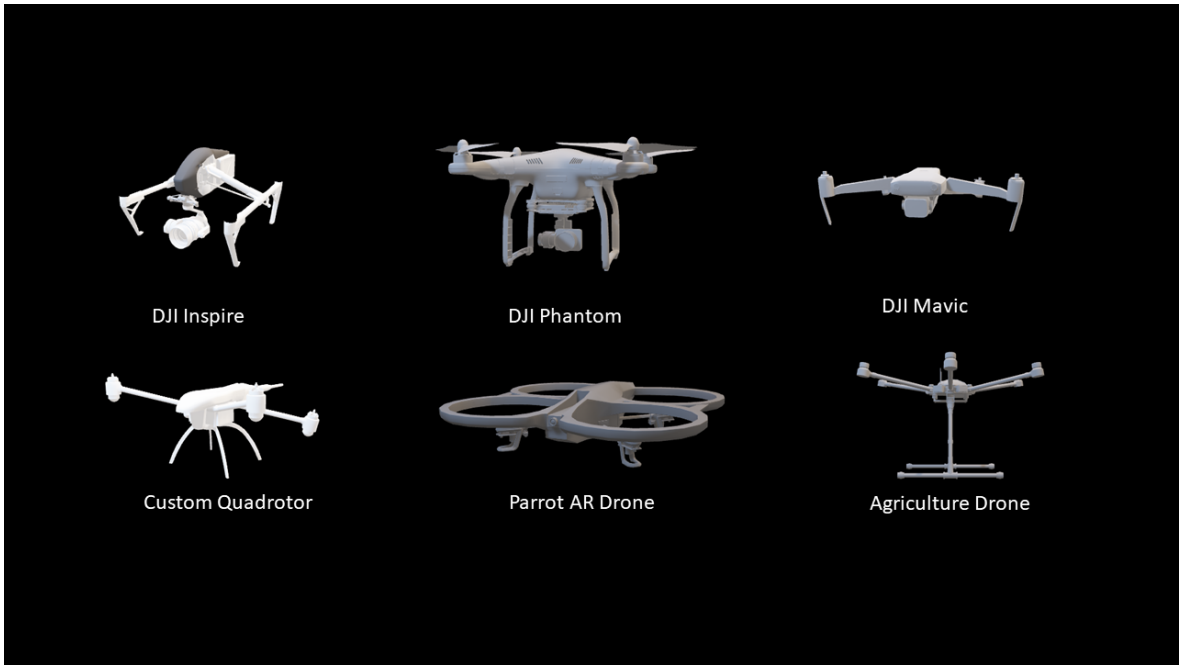


FIGURE 4.3. Six 3D drone models gathered as representation of different drone models and sizes

First, 3D models of the drones, which are the objects to be detected, were obtained. While creating these models, civilian drones such as DJI Mavic, DJI Inspire, DJI Phantom, Parrot AR Drone and DJI Matrice, which are some of the popular drones, as well as agricultural drones such as large-sized DJI Agras were taken as reference. After the meshes of these drones were obtained, realistic materials and textures were prepared for each model. Propeller texture with rotational blur has been added to the models to represent the rotation of the drones' propellers. As a result of these studies, a model pool was created to represent small, medium and large-sized drones.

Secondly, we collected 4000 background images, 1,000 from each category with different themes. These images and categories are of great importance in terms of the performance of the method we created. For this reason, background images that drones can often interfere

with have been chosen. Backgrounds with this feature are grouped under four categories according to their content;

FIGURE 4.4. Samples from urban backgrounds dataset.

- Urban : Urban backgrounds mostly contain modern city environments. Cities have many objects that can be confused with drones due to their complex and crowded structure. Especially distant objects such as cars, motorcycles and people, traffic and street lights increase false-positive and decrease model performance due to their similar features to drones.

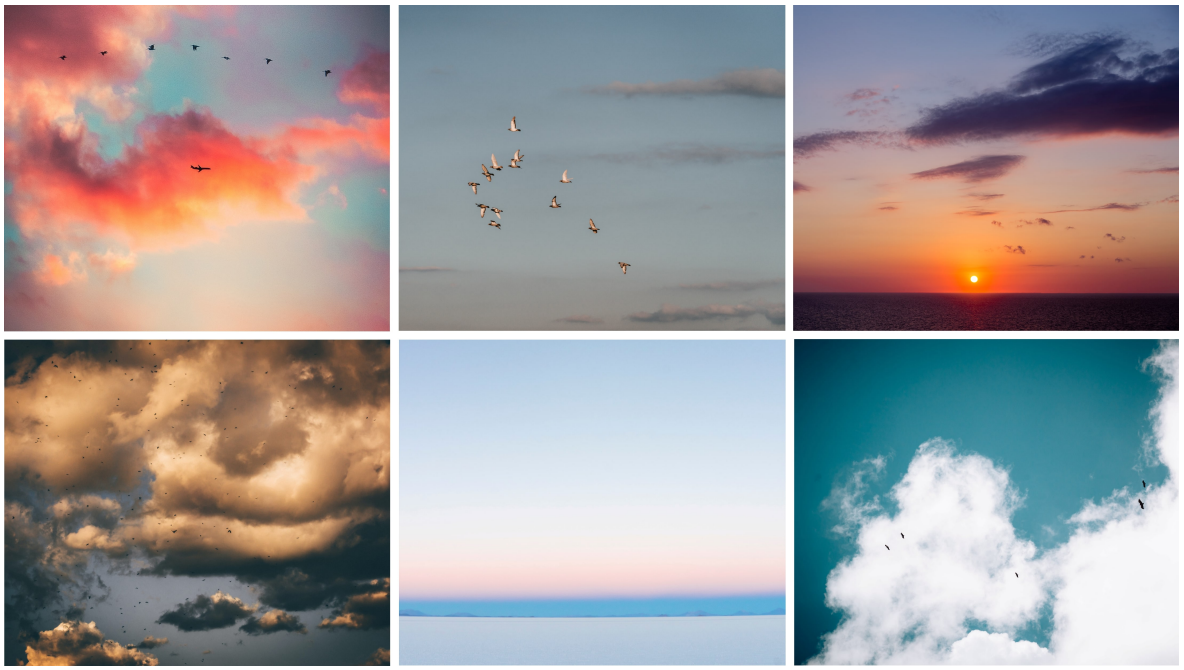

FIGURE 4.5. Samples from sky backgrounds dataset.

- Sky: Most of the drone images have sky backgrounds. The main reason for this is that these images are taken from the ground to the sky and drones are objects that can fly. Therefore, the presence of birds and planes, which are objects that can mix with the drone in the sky, as negative samples in the background increase the performance of the model. Also, some cloud structures resemble drones under different lighting conditions and shadows. To reduce this situation, it is very important to represent as many different cloud variations as possible in the background.

FIGURE 4.6. Samples from nature backgrounds dataset.

- Nature: Nature images are an important theme for drone recognition as they include different plant structures and animals. There are many different types of trees in nature, and their complex branch structures sometimes resemble a drone. The presence of trees in different colors, sizes and styles in the background contributes to the false-positive amount of the model. In addition, animals in nature can be confused with drones in distant shots.

FIGURE 4.7. Samples from misc backgrounds dataset.

- Misc: Under this category, there are images with random patterns, colors and shapes. Although images are used in general themes that can mix with the drone, it is not possible to find images for all types of backgrounds that can blend with the drone. Therefore, placing drone models on random patterns and colors increases the robustness of the model, similar to domain randomization.

In the presented method, the domain-specific environment creation part, which is the most time-consuming part in synthetic data generation methods, has been accelerated by using 2D backgrounds and 3D drone models. The creation of the above-mentioned 3D drone models and the collection of backgrounds on the themes suitable for the problem also contains some problems in themselves. Since 3D drone models have different polygon numbers and details, they had to be simplified and brought to the same level of detail. For this, the geometry decimation process is applied and detail levels are adjusted. 2D background images have different resolutions, dpi levels and formats. This creates a contrast between 3D models and backgrounds and reduces realism. The blending techniques applied to solve this problem are explained in detail in section 5.3.

Creating complex 3D environments takes much more time than the steps and problems mentioned above. In addition, it is very difficult to provide the desired detail. For these reasons, our method using 2D backgrounds and 3D drone models seems to be the most logical solution for the drone detection problem.

# 5. GENERATION METHOD

We explained the reasons and pros of using game engines in the production of synthetic data, the use of 2D backgrounds and 3D drone models in the section. In this section, we will detail how we created a controlled simulation infrastructure to generate synthetic data using these resources and the Unity 3D game engine. How a pipeline was created to create this infrastructure, the parameters used by this pipeline, the applied post-processing effects, the blending techniques that allow the models to mix with the backgrounds, and the structure of the data set produced as a result of the pipeline will be mentioned.

## 5.1. Generation Pipeline

Labeled synthetic image generation generally consists of five steps; environment randomization, segmentation calculations, image creation, post-processing and dataset file creation. Although these stages are independent of the technologies used, they are similar in most synthetic data generation approaches.

Since Unity is a 3D game engine, it generates images with real-time rendering. Real-time rendering is provided by an endless loop running in the background, a loop called a game loop. The render of the scene seen by the camera is displayed after the calculations determined in each loop are made. We will refer to each loop mentioned in our method as an ”iteration”.

As seen in Fig 5.1., our method consists of nine steps in total. Eight of these steps are repeated in each iteration, and the final step, annotation file creation, takes place after all iterations are completed. These processes repeated in each iteration are as follows;

1. Initialize simulation: This stage resets the simulation at the beginning of each cycle to reset the scene and check memory management. To achieve this, it returns all the objects in the scene to the object pool, resets the textures in the buffer, and resets the scene parameters to their default values.

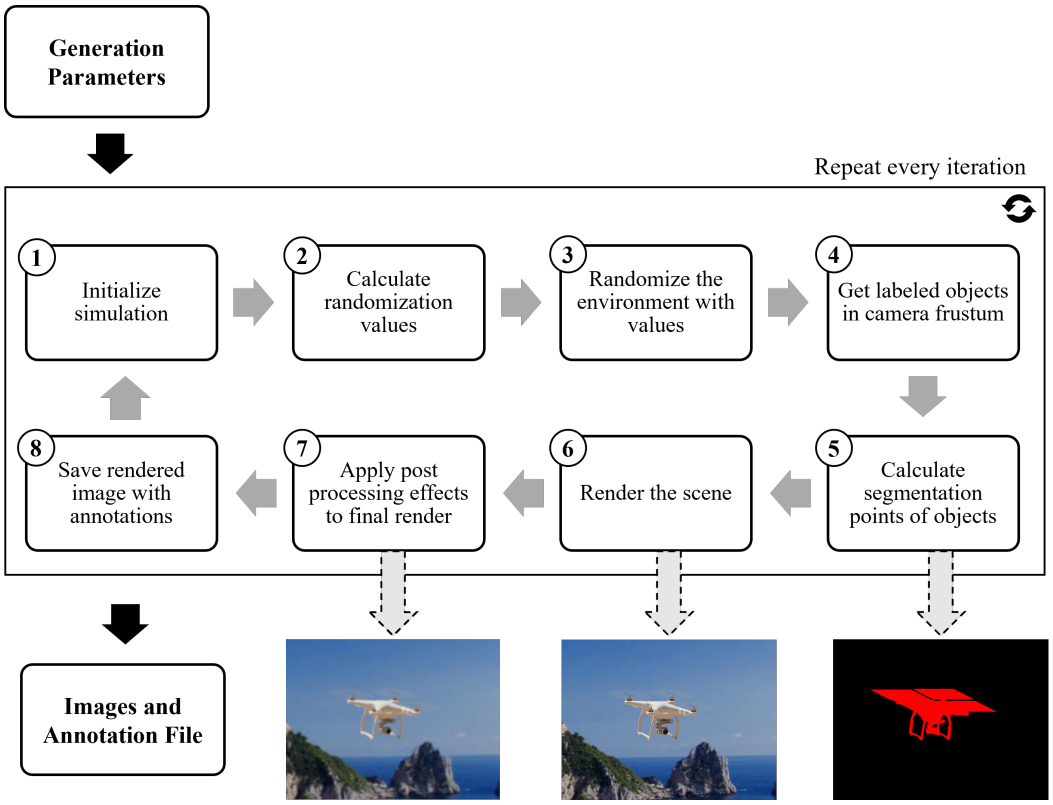


FIGURE 5.1. The pipeline of synthetic data generation. Pipeline takes generation parameters as input and gives a output file containing images and annotations.Each step in the loop is repeated at each iteration.

2. Calculate randomization values: This step calculates the values for the randomizations that will occur in each frame. These values are calculated within the limits determined for each parameter and with different distributions. We will describe the parameters, their functions and their distributions in more detail in Section 5.2.

3. Randomize the environment with values: The randomizations calculated in the previous step are used to construct the scene in this step. The background image, the position and properties of the light, the orientation and properties of the drone objects are arranged using the calculated randomization values.

4. Get labeled objects in camera frustum: As a result of randomizations, there may be a single drone or multiple drones or distractor objects on the scene. Among these

objects, the objects labeled as drones and remaining in the camera field of view are determined and their references are gathered in a list.

5. Calculate the segmentation points of objects: In this section, the boundaries of the 3D model are calculated from the mesh buffer for each object in the reference list. Then, the values of these borders in world space are converted to screen space and the location of each point in the image is calculated. Using these points belonging to each object, the object reference and the point list of the object are stored after the bounding box containing the object is calculated.

6. Render the scene: After the randomization and segmentation processes are completed, the scene is rendered and the image is created.

7. Apply post-processing effects to final render: The final image is created by adding post-processing effects selected under the randomization parameters on the rendered image.

8. Save rendered image with annotations: After all, processes are completed, the final image is saved as a file. The class, segmentation and bounding box information of the objects in the image are written to the annotation file and the loop starts again.

After the pipeline, we have developed reaches the specified number of iterations, the generation cycle ends. As a result of these processes, a ”JSON” file containing references of images and annotations of these images is created, and the data set creation is finalized.

## 5.2. Generation Parameters

The system we have created is completely parametric and the properties of the created data set are completely dependent on the specified parameters. Thanks to the parametric structure, experiments can be carried out quickly and the effect of each parameter on the trained model can be measured. While determining the generation parameters, the problem we want to solve and the features that need to be controlled are considered. The parameters we use to

randomize the scene have three categories; background, lighting and drone. Because some of the parameters are not independent of each other, the order in which the parameters are processed is important. After each parameter is randomized, the next parameter changes depending on the previous values. The functions and distributions of these parameters are as follows;

Background Parameters

- Background ID (Distribution: Uniform - Output: Integer): Randomly selects one from the background texture pool and places it on the background.
- Background Offset (Distribution: Uniform - Output: Vector2): Brings a certain part of the image to the camera frame by offsetting the placed background image vertically and horizontally.As explained in Sec. 4.3. , background images can be in different resolutions. Thanks to offset randomization, the same image can be used many times and all backgrounds are reduced to a single resolution.

Drone Parameters

- Drone Spawn Probability (Output: Boolean): Calculates the probability that the scene contains a drone in the current iteration. The fact that there is no negative sample, that is, no annotation of the object to be detected in the created data set, increases the model performance. Therefore, the creation of empty scenes with a certain percentage has been made parametric. If the drone will not be created as a result of randomization in the current iteration, other randomizations are canceled and the intermediate steps are skipped and the sixth step seen in Fig. 5.1. is started.
- Drone Count (Distribution: Uniform - Output: Float): Randomizes how many drone models will spawn in the current iteration.
- Drone Spawn ID (Distribution: Uniform - Output: Integer): Selects a random 3D model from the drone model pool. It is repeated as much as the number of drones calculated in the previous step.

- Drone Position (Distribution: Uniform - Output: Vector2): This parameter generates a random position for each drone in each scene, within the camera viewpoint.
- Drone Rotation (Distribution: Normal - Output: Quaternion): Generates a random rotation for each drone in each scene. Unlike other parameters, this parameter has normal distribution. The drones in the scene should have the rotations that real drones can have. Therefore, the roll and pitch values have a normal distribution between [-90,90] degrees, and the yaw value has a uniform distribution between [0,360] degrees.
- Drone Scale (Distribution: Normal - Output: Float): Sets the scale of the drone. In our method, there is no perspective because the camera has orthographic projection. Therefore, the distance perception in drones is obtained by changing the scale of the drone objects.

Lighting Parameters

- Lighting Rotation (Distribution: Uniform - Output: Float): Similar to the sun, the light source is randomized at an angle of 180 degrees. The parameters after this procedure change according to the randomization value at this stage. In this way, the color and brightness value at every angle is consistent with the sun.
- Lighting Color (Distribution: Uniform - Output: Color32): Randomizes the color of the light source based on the current rotation of the light source. The randomized angle value in the previous step changes the color range. As the light source gets closer to 0 and 180 degrees, randomization takes place in darker tones, while around 90 degrees it gets lighter colors.
- Lighting Brightness (Distribution: Uniform - Output: Float): Determines the brightness of the light source in a certain range based on the color value randomized

Many different datasets were generated using the above-mentioned parameters. By examining the trainings received with these datasets, it is aimed to solve the nature of the synthetic

data and the problem. Apart from these parameters, the values of the post-processing effects, which will be mentioned below, are also randomized within a certain range.

## 5.3. Post Processing and Blending

As can be seen in Fig 5.1., after the randomization, segmentation and rendering processes are completed, post-processing effects are applied to the final image in the seventh step. Post-processing effects aim to imitate the artifact and visual effects created by real camera systems.

While backgrounds consisting of images shot with real cameras have these features, 3D drone models do not have these features. Therefore, there is a contrast between the background and the 3D models. This difference is used by the model as a shortcut to detect the drone object, so a feature that does not exist in real drones is taught to the model. This situation causes object detection models trained with synthetic data to show high performance when tested on synthetic data but underperform in tests with real data.

As a solution to these problems, we use post-processing effects and blending methods. Although all the effects tried are not positive for the model, they are as follows;

- Film Grain: Film grain mimics natural noise in real cameras. Random noise added to the rendered images reduces pixel-based artifacts and provides a more natural distribution to the rendered image.
- Lens Distortion :Lens distortion is caused by camera lenses distorting the projection of the image. The effect we use creates a distortion similar to this situation. In theory, the images were taken with the real camera also have some lens distortion and this effect will increase the model performance since this ratio changes according to different cameras.
- Motion Blur: Motion blur is the blurring of objects as a result of moving objects moving in a shorter time than the shutter speed of the camera. Due to the fact that there are

moving objects in drones, different amounts of motion blur are exposed according to the recorded camera. It is very difficult to create realistic motion blur in game engines, and low-quality motion blur causes the object to become unrecognizable. That's why we experimented with using a small amount of motion blur in our method.

- Depth of Field: The Depth of Field (DoF) effect is an effect that mimics the blurring of other objects when cameras are focused at a certain distance. The drone may become blurred as a result of the cameras losing their focus, and the background may appear more blurred as a result of a camera focusing on the drone.

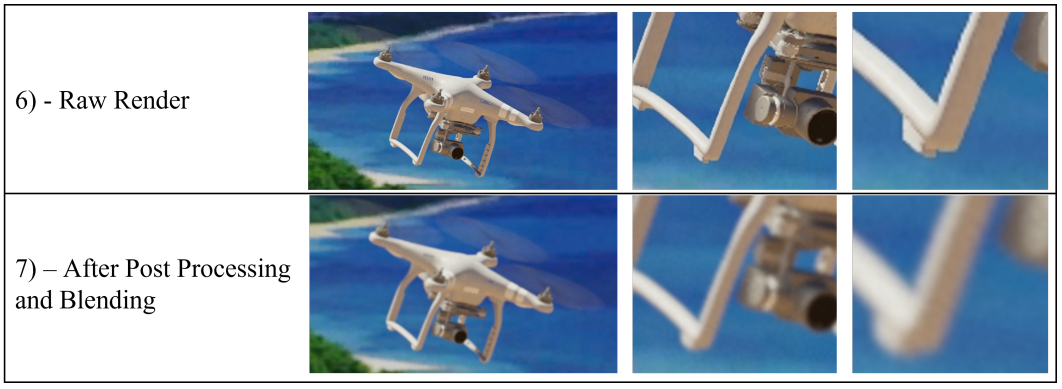


FIGURE 5.2. The raw rendering result of the scene as a result of randomizations is above, and the result after post processing and blending is below.

Although post-processing effects add realism and richness to the visuals we produce by imitating the effects created by real cameras, they are not enough on their own. Pixel-based differences between drone models and background images create a noticeable difference, especially in the border areas where the models and the background intersect, and this difference increases as the background resolution used decreases. This visible difference becomes a feature to detect a shortcut drone while the models are being trained, and the lack of this feature in real drone images reduces performance. In addition, as a result of real-time rendering and game engine optimizations, there is pixelation in the border regions of the models. While this situation is observed less in large-scale models, pixelation increases in small models and the drone model moves away from reality.

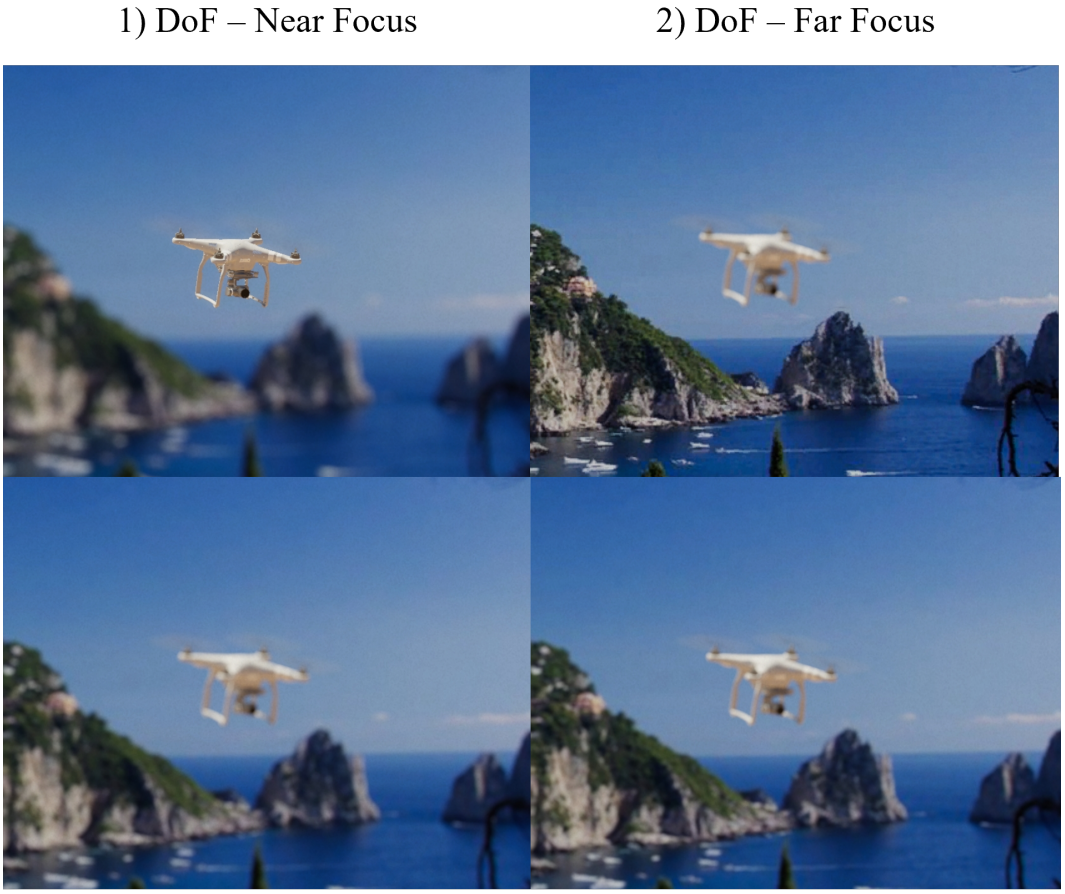


FIGURE 5.3. Comparison of four different blending technique.

Blending techniques using blur have been considered as a way to solve these problems created by 3D Models and 2D backgrounds. Using blur on the models both reduces the contrast with the background and eliminates the artifacts in the border regions. As can be seen in Fig. 5.2., the result of applying a blur to the model on the rendered render has been successfully blended into the background. This image, which is more natural to the human eye, also positively affects the model performance. Four different ways were tried to achieve the blending.

As the first method, we used the Depth of Field effect that we mentioned above. Thanks to DoF, we can adjust the blur level of distant and near objects by adjusting the focus of

the camera. While this method is logical, it involves two main problems. The first of these problems is blurring when the background loses focus, and details are lost when an already low-resolution background is used. Another problem is that DoF calculates focus and therefore blur based on distance, causing blurring of the entire background in two-dimensional backgrounds. This situation causes the loss of many details in the background depending on the first problem.

As the second method, we performed the blending process by adding a constant value Gaussian Blur on the drone model. Gaussian Blur softens the color transitions by taking the average of the colors in the neighboring pixels for each channel. Thus, a smoother and more realistic image is obtained and pixelation is prevented. While determining the amount of Gaussian Blur we added, object detection models that we trained with real data were used. When inference is performed with a model trained with real data on a theoretically produced image, the more successful the blending, the higher the confidence of the model will be. Using this logic, the confidence of the images produced with blur at different rates on the model was examined and the most optimal blur amount was selected. We call this method ”Fixed Blur”. The problem with the Fixed Blur method is that the same amount of blur is applied regardless of the resolution of the background. This leads to successful blending in some backgrounds and unsuccessful blending in others.

The third method used the object detection model trained with real data, similar to the previous method. In this method, for each different background used differently, scoring was done with inference, and Gaussian Blur was applied at a varying rate according to this scoring. In this way, a blending suitable for different resolution backgrounds was achieved. We call this method ”Inference Blur”.

As a final method, a method was developed by analyzing the backgrounds using image processing techniques and applying a blur to the drone model in an appropriate amount to the blur ratio of the background image. Our aim in developing this method is to create a fully synthetic data-driven approach by eliminating the dependency of other methods on the object detection model trained with real data. To calculate the blur in the background image, the

amount of edge in the image is calculated. Theoretically, the less edge an image has, the less sharp it is and therefore the more blurred it is. Although this theory changes according to the content of the image, it gives an average value about the amount of blur of the image. ”Canny Edge Detection Algorithm” [32] was used to find the amount of edge in the images. With this algorithm, a sharpness score was calculated for each image and blur was applied to the drone model inversely proportional to this score. This technique, which we call ”Adaptive Blur”, has been the preferred technique for blending because it gives close results when compared to the Inference Blur method.

The outputs of the four techniques developed in Fig. 5.3. are seen. Figures 1 and 2 show the problem mentioned about near and far focuses in the method using DoF. As seen in figures 3 and 4, Inference Blur and Adaptive Blur methods give similar and realistic results. In Sec. 6. , the comparison of the models trained with these methods and the performance difference they create will be examined in more detail.

## 5.4. System Output and Performance

As can be seen in the pipeline Fig. 5.4. we created, with the correct generation parameters, post-processing and blending technique, it gives results very close to real images.

FIGURE 5.4. Comparison of synthetically generated images (left) and real images from Drone vs. Bird dataset (right)

The system creates a JSON file containing the references, classes and annotations of these images, together with the produced synthetic images. The ”COCO” [33] format, popularly used in object detection, was used to create this file. The COCO format is very useful in that it can be easily integrated into many object detection frameworks. In the annotations in the COCO file, the bounding box format is [x, y, width, height] and the segmentation points are kept as a list.

Along with the capacity of the created system to produce realistic images, how fast it can produce these images is also an important factor. In this study, many experiments were conducted to understand and optimize the nature of synthetic data. Therefore, the data generation speed of the system is one of the most important factors that determine the number of experiments that can be performed. The real-time rendering capacity of game engines has helped us to create a system that generates data very quickly.

# 6. EXPERIMENTS & RESULTS

In this section, we will analyze the experiments made with the drone detection dataset produced by our system, whose design we described in Sec. 5.1. Experiments were carried out to test the datasets produced by the system and to optimize the system parameters. First, we will describe how we created a structure to carry out the experiments. Next, we'll review our ablation study to find key features for synthetic data. Finally, we will discuss the experiments and their results.

## 6.1. Experimental Setup

### 6.1.1. Object Detection Framework

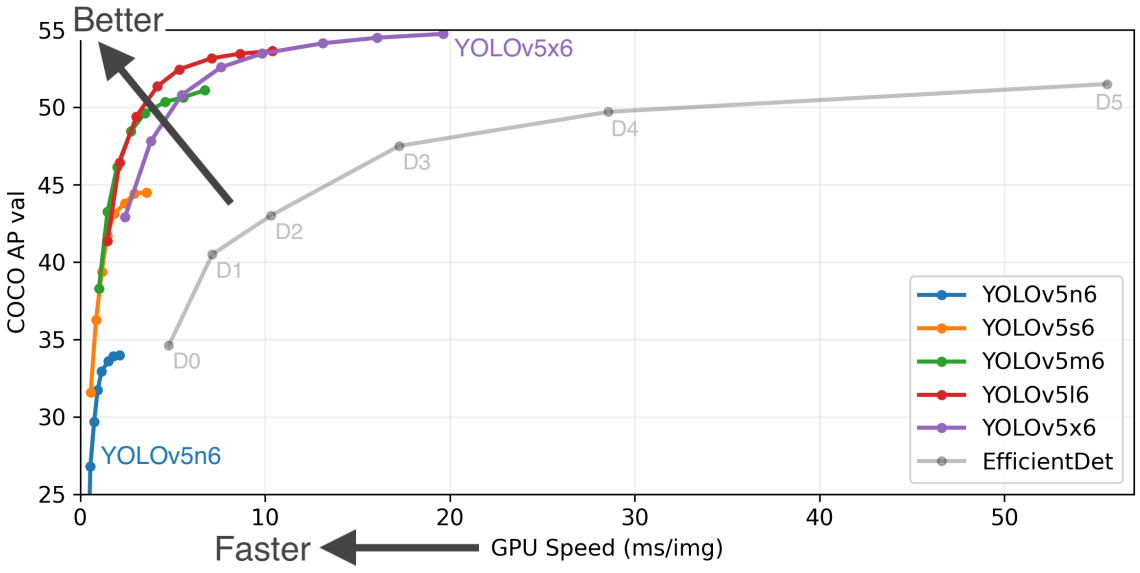


FIGURE 6.1. Comparison of EfficentDet and YOLOv5 models [3]. YOLOv5X6 slower than other models but gives best mAP results.

In this study, our main goal is to create a dataset to train an object detection model rather than design an object detection model for drone detection. However, it is necessary to take experiments with datasets to evaluate and analyze the generated datasets. For this purpose,

we chose ”YOLOv5” [3], a popular object detection framework, to be used in all trainings. Our main reason for choosing YOLOv5 is that it has the most successful model architecture in detecting small objects. In addition, thanks to its modern and easy-to-use structure, it gives successful results without making detailed adjustments.

### 6.1.2. Training Parameters

The number of training images used in all experiments varies and the remaining parameters are kept the same. Parameters were determined by considering the experimental time and hardware limits. Parameters such as the number of images, image resolution and epoch count greatly affect the training time. It is very important to keep the training period short, especially since there are too many trainings in ablation studies. In addition, the visual resolution is limited by the video card memory of the training computer. Although the RTX 3090 video card, which is the video card that we have been training with, has 24 GB of VRAM, it is not enough to train with Full HD resolution input size. Accordingly, the following parameters were used in the ablation study;

- 1344 Input Size
- 5 epoch
- 10k training image
- Yolov5m6 backbone
- Finetune Hyperparameters

Although the amount of training image and the number of epochs were changed in the final trainings, the remaining parameters were kept the same.

### 6.1.3. Metrics

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$

$$Recalls(n) = 0, Precisions(n) = 1$$

$$n = Number\ of\ thresholds.$$

FIGURE 6.2. Calculation of mean average precision (mAP)

It is necessary to use a common metric in order to compare the trainings received with different data sets. We used COCO metrics to evaluate performance in all of our experiments. ”Mean Average Precision (mAP)” is used as the performance unit in the COCO metric. mAP is a way to summarize the precision-recall curve into a single value representing the average of all precisions. The AP is calculated according to the next equation. Using a loop that goes through all precisions/recalls, the difference between the current and next recalls is calculated and then multiplied by the current precision. In other words, the AP is the weighted sum of precisions at each threshold where the weight is the increase in recall. For Precision, we get two outputs, 0.5 and 0.5-0.95. Since drones are small objects, we will use $mAP_{0.5}$ as the main metric.

### 6.1.4. Dataset Usage

Different uses for real and synthetic datasets were used in the experiments. As described in Sec 3.3., only the Drone vs Bird dataset was used as actual data. The Drone vs Bird dataset is divided into two parts, %85 for training and %15 for validation. In this way, approximately 85000 images for training and 15000 images for validation were obtained. drone vs. Since the Bird dataset consists of videos, using all frames in training leads to overfitting and reduces performance. Therefore, we perform trainings by applying different amounts of subsampling. We call these datasets ”Real Dataset”.

For synthetic datasets, we will use the dataset we obtained as a result of the ablation study we performed in the next section. We call this dataset ”Synthetic Dataset”.

## 6.2. Ablation Study

Unlike real data, all variables of the data can be controlled while generating synthetic data. Therefore, it is of great importance to find the features that will make the synthetic data most successful. In order to find these features, first by fixing certain randomization parameters, the dataset produced with these parameters was determined as the base dataset. Then, the features that may be important for the drone recognition problem were determined and the effect on the model performance was measured by changing one feature at a time. This systematic approach, called ablation study, gives us the opportunity to identify positive and negative features.

Table 6.1. shows the training results using synthetic datasets produced with selected features. For each determined feature, mAP 0.5, mAP 0.5-0.95, precision and recall scores are listed. mAP values show the average performance of the dataset, mAP 0.5 is determined as the main metric for the purpose of detecting small drones. Therefore, performance comparisons are made according to mAP 0.5 and feature selections are made.

In addition, precision and recall metrics are used to better understand the effect of the feature on the model. Precision is the ratio of how many of all predictions (true positive and false positive) are correct. The recall is the ratio of how many of the objects in the image can be found correctly. These concepts are illustrated in Fig 6.3..

The training graphics and explanations of the features we used in the ablation study are as follows;

TABLE 6.1. Comparison table of mAP, precision and recall scores of features. Features with ”*” symbols used in the ”Best” dataset.

| Feature | mAP 0.5 | mAP 0.5-0.95 | Precision | Recall |
|---|---|---|---|---|
| Base | 0.496 | 0.186 | 0.586 | 0.396 |
| Smaller Drones (*) | 0.541 | 0.198 | 0.618 | 0.489 |
| Without Propeller | 0.412 | 0.198 | 0.591 | 0.489 |
| Random Drone Color | 0.511 | 0.192 | 0.609 | 0.412 |
| Random Light Color | 0.435 | 0.174 | 0.513 | 0.332 |
| Hue Shift | 0.382 | 0.112 | 0.211 | 0.185 |
| Multiple Drones (*) | 0.52 | 0.258 | 0.612 | 0.476 |
| Distractor Objects | 0.518 | 0.213 | 0.598 | 0.489 |
| Bird Objects | 0.509 | 0.212 | 0.614 | 0.487 |
| Noise (*) | 0.56 | 0.189 | 0.64 | 0.562 |
| Lens Distortion | 0.473 | 0.122 | 0.496 | 0.318 |
| Motion Blur (*) | 0.554 | 0.151 | 0.566 | 0.491 |
| Depth of Field | 0.501 | 0.178 | 0.642 | 0.473 |
| Gaussian Blur | 0.578 | 0.227 | 0.712 | 0.574 |
| Inference Blur | 0.592 | 0.248 | 0.789 | 0.601 |
| Adaptive Blur (*) | 0.612 | 0.249 | 0.788 | 0.628 |
| **Best** | **0.713** | **0.292** | **0.811** | **0.649** |

### 6.2.1. Base Dataset

In the base dataset, the background, drone and lighting parameters mentioned in Sec. 5.2. are fixed to a certain value. No post-processing effects have been applied to this dataset. All experiments were done by adding different features to this data set, and performance comparisons were based on this data set.

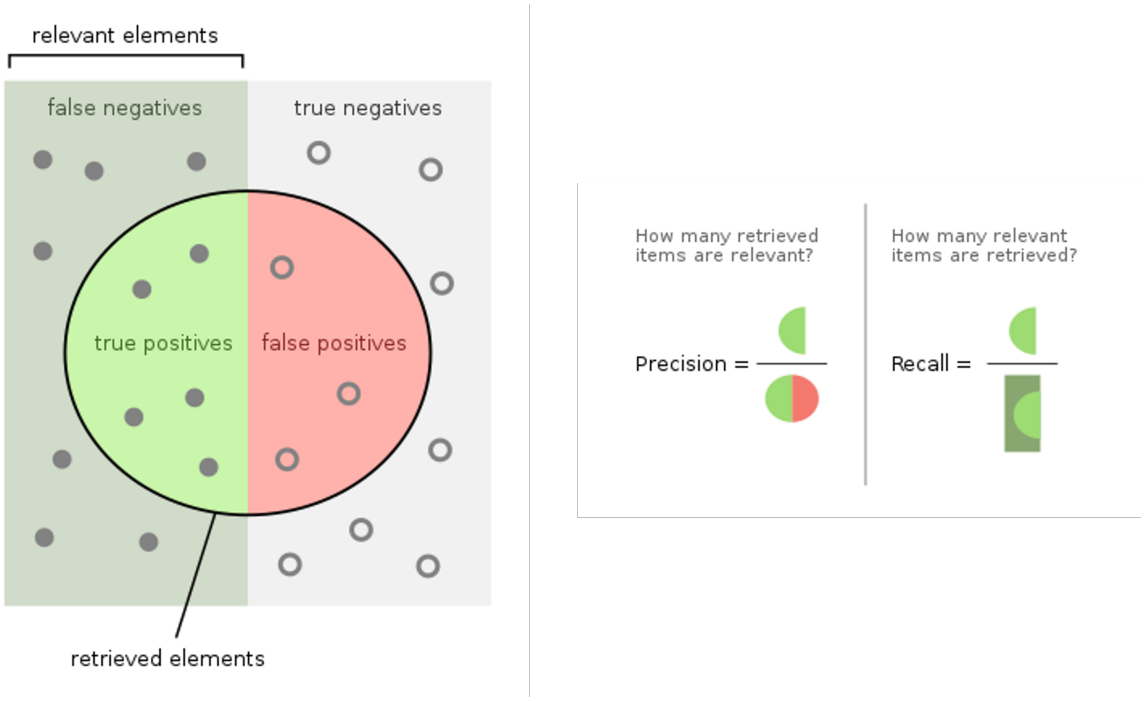


FIGURE 6.3. Illustration of precision and recall

### 6.2.2. Smaller Drones Feature

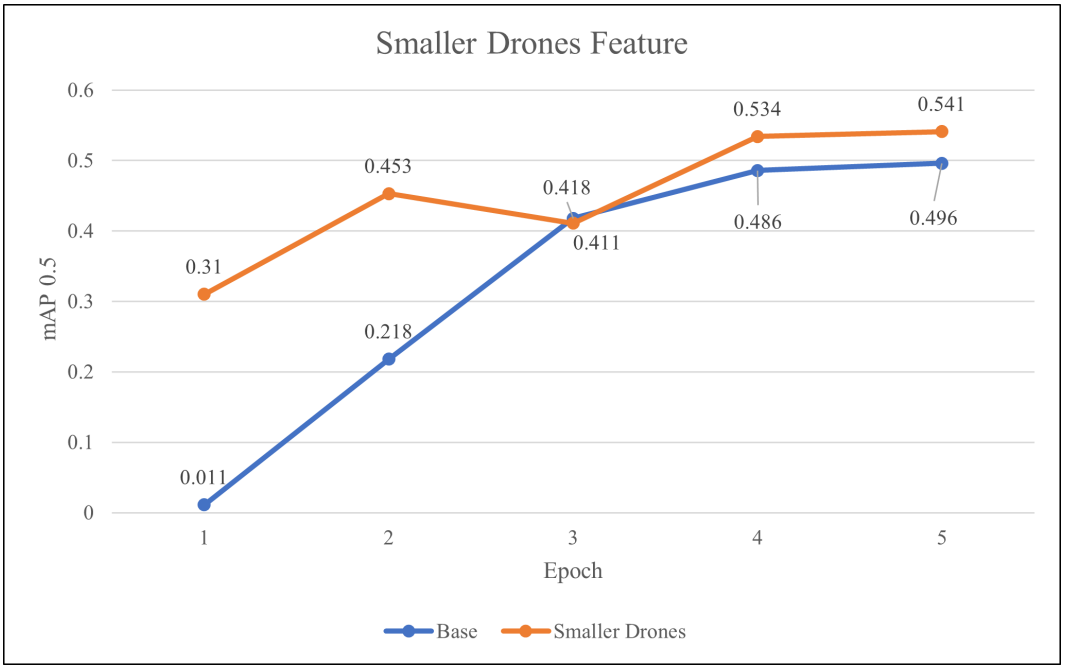


FIGURE 6.4. Training graphs of smaller drones and base dataset

In this dataset, the effect of scale randomization of drones on performance was measured. In this direction, smaller drones were obtained by reducing the scale parameter by fifty percent. As seen in Fig. 6.4., the training started with a much higher mAP score than the base data set and ended with a higher mAP score. The main reason for this is Drone vs. The Bird dataset is mostly composed of small and remote drones and this dataset is used as validation. This feature was chosen to be used in this study because our goal is to increase the detection performance of small drones.

### 6.2.3. Without Propeller Feature

FIGURE 6.5. Rendering with without propeller feature at left, base at right

In this dataset, the importance of propellers in drone models was measured. Therefore, a data set was created in which propellers were removed from all drone models. As seen in Fig. 6.6. , the mAP score remained low throughout the training and was completed with a lower mAP than the base data set. The main reason for this can be seen as the propellers being a defining feature while models detect drones. This is why the model that learned to find drones without propellers drones failed to find propellers. Therefore, this feature will not be used in the best dataset.

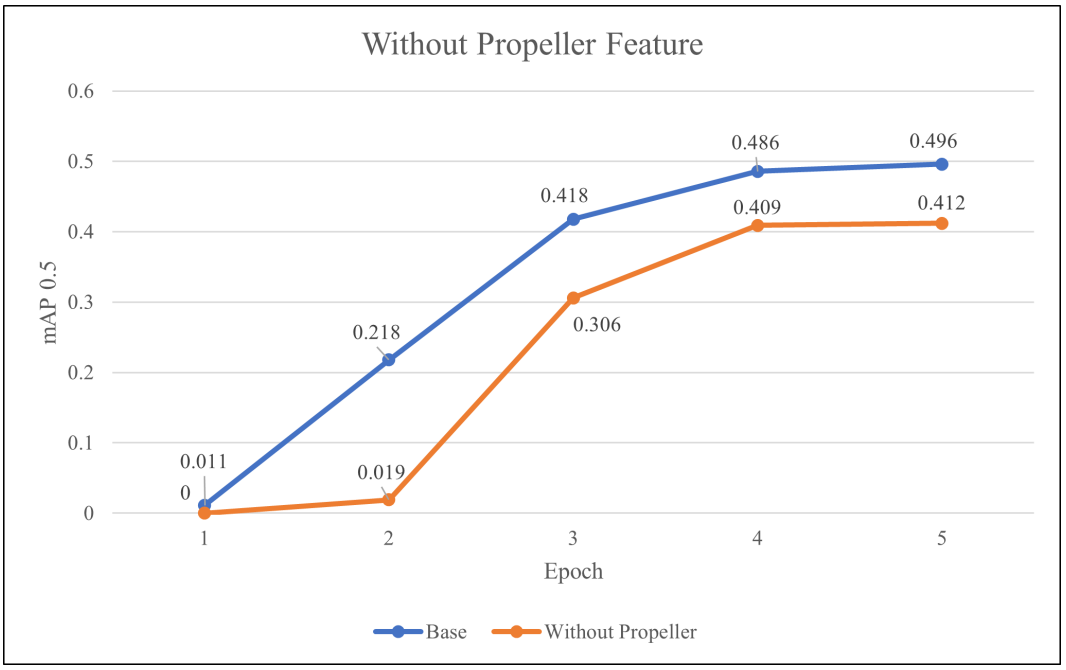


FIGURE 6.6. Training graphs of without propeller and base dataset

### 6.2.4. Random Drone Color Feature

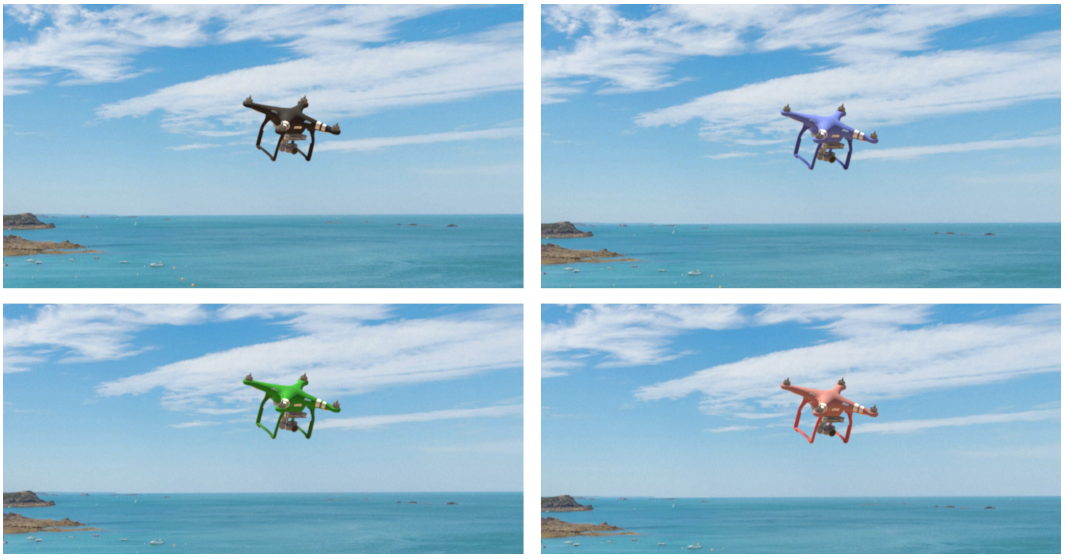

FIGURE 6.7. Rendering with randomly colored drones

In this dataset, the effect of random color drones on model performance was measured. In this experiment, random colors were given to the drones, similar to domain randomization. As seen in Fig. 6.8. , this approach did not provide an increase compared to the base data set.

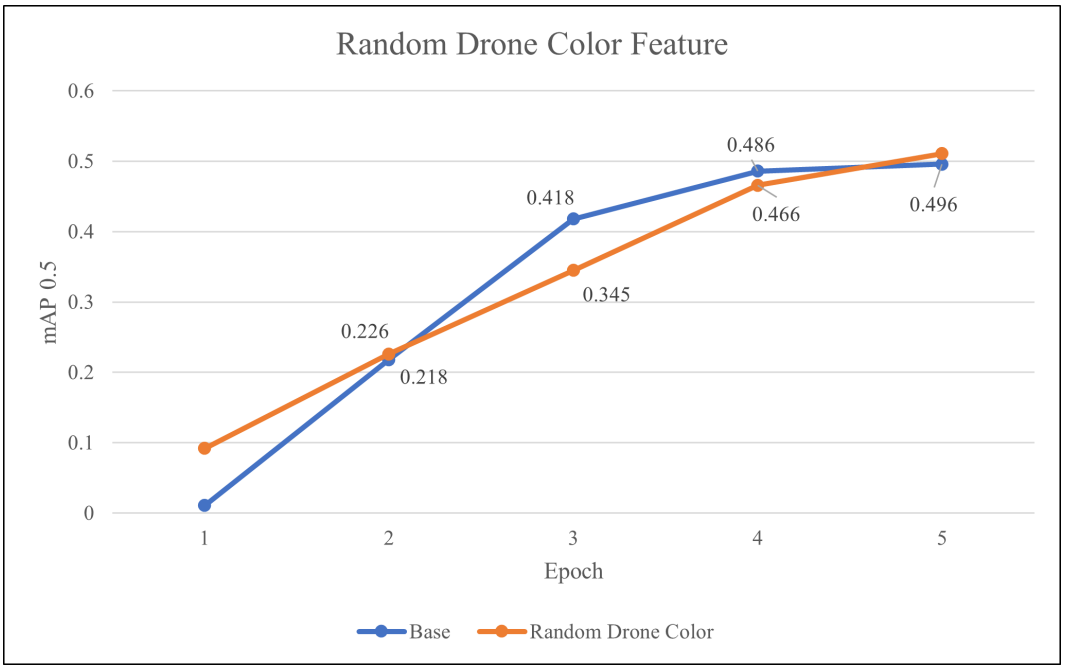


FIGURE 6.8. Training graphs of random drone color and base dataset

### 6.2.5. Random Light Color Feature

FIGURE 6.9. Rendering with randomly colored light source

In this data set, when a random color is given to the light source in each iteration, its effect on performance is measured. Similar to the random drone color experiment, the drone randomization structure was tested. As can be seen in Fig. 6.10. , this feature has reduced performance.

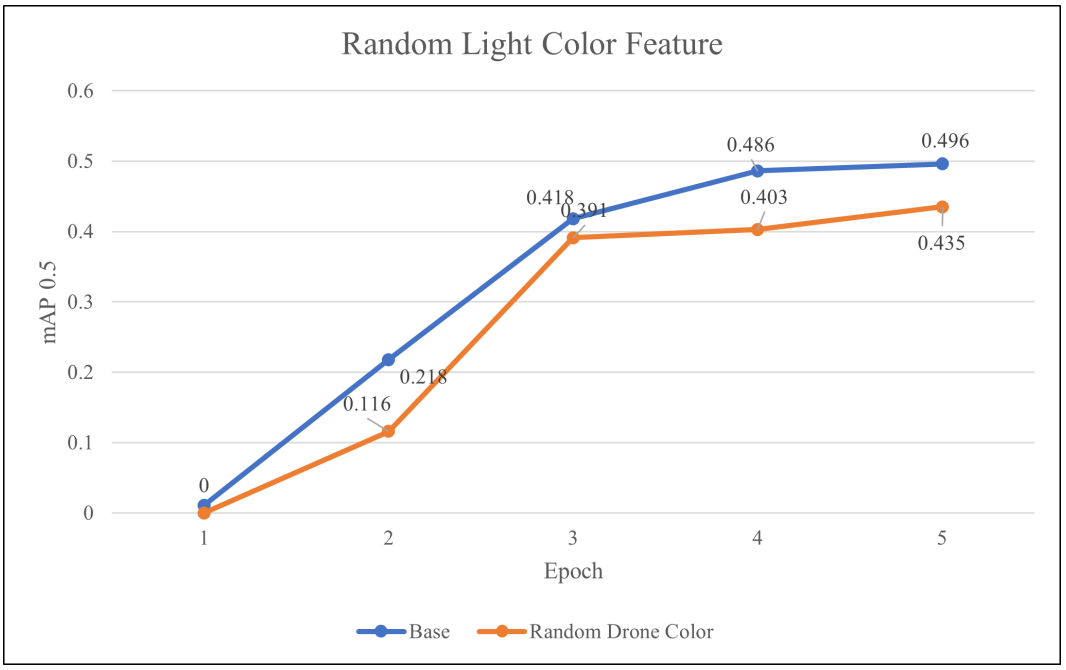


FIGURE 6.10. Training graphs of random light color and base dataset

### 6.2.6. Hue Shift Feature

FIGURE 6.11. Rendering with hue shift feature

In this experiment, the effect of hue shift, an effect that changes the color palette, on performance was measured. In this way, the colors of the entire render, not just the drone, change completely randomly. Similar to light and drone color randomization, the domain randomization approach was tested. As can be seen in Fig. 6.12., this feature has seriously reduced performance.

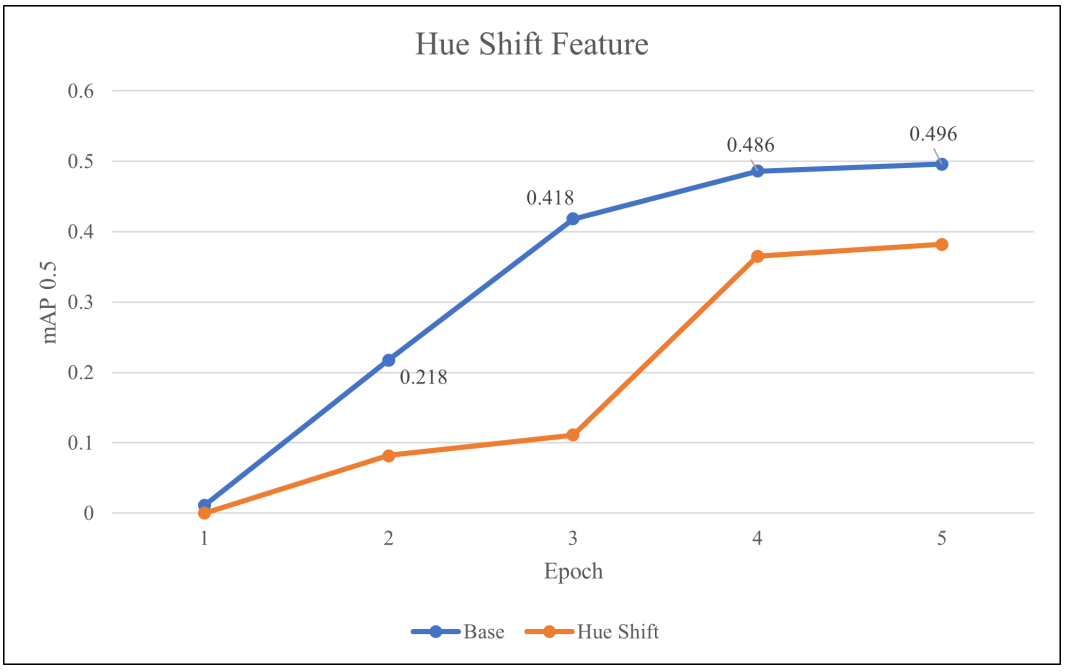


FIGURE 6.12. Training graphs of random light color and base dataset

### 6.2.7. Multiple Drones Feature

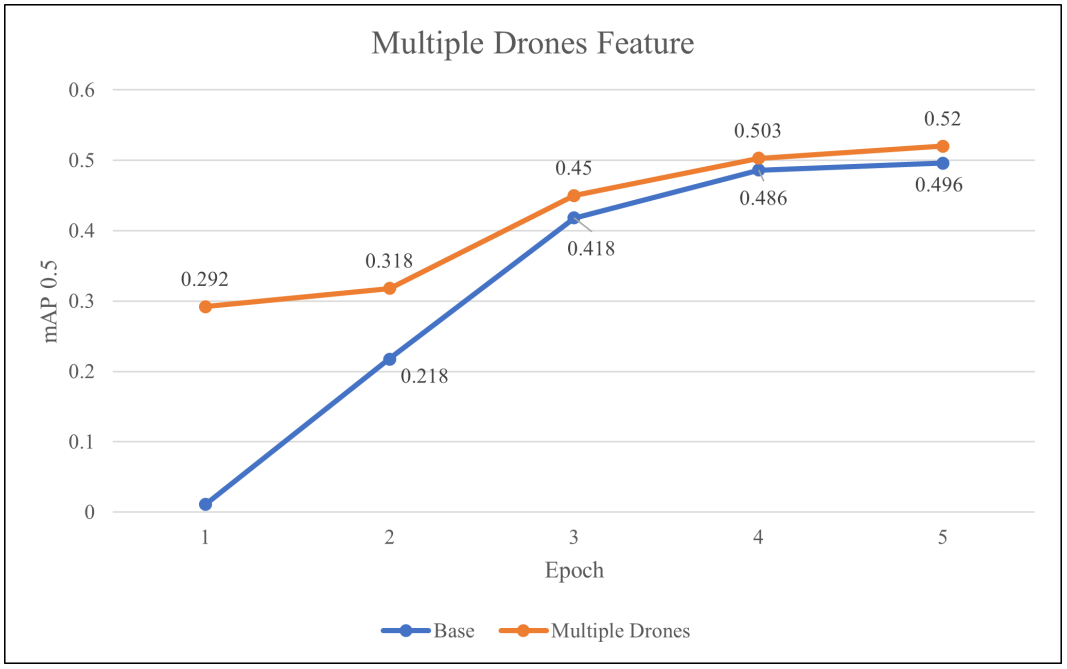


FIGURE 6.13. Training graphs of multiple drones and base dataset

In this experiment, the effect of having more than one drone in each iteration on the performance was measured. Theoretically, having more drones per iteration should improve performance by seeing more samples. As seen in Fig. 6.13. , although the training started with a high mAP score, this increase could not continue at the same rate. The reason for this may be that there is only one drone in the Drone vs Bird dataset, and as a result, the amount of false positives increases as a result of waiting for more than one drone in the frames in the training. However, it was decided to use it in the Best dataset.

### 6.2.8. Distractor Objects Feature

FIGURE 6.14. Rendering with distractor objects

This dataset aims to reduce the number of false positives by placing random geometric objects next to the drone objects. In each iteration, random geometric objects of different sizes and positions are placed together with the drones. As can be seen in Fig. 6.15. , it did not have a big impact on performance.

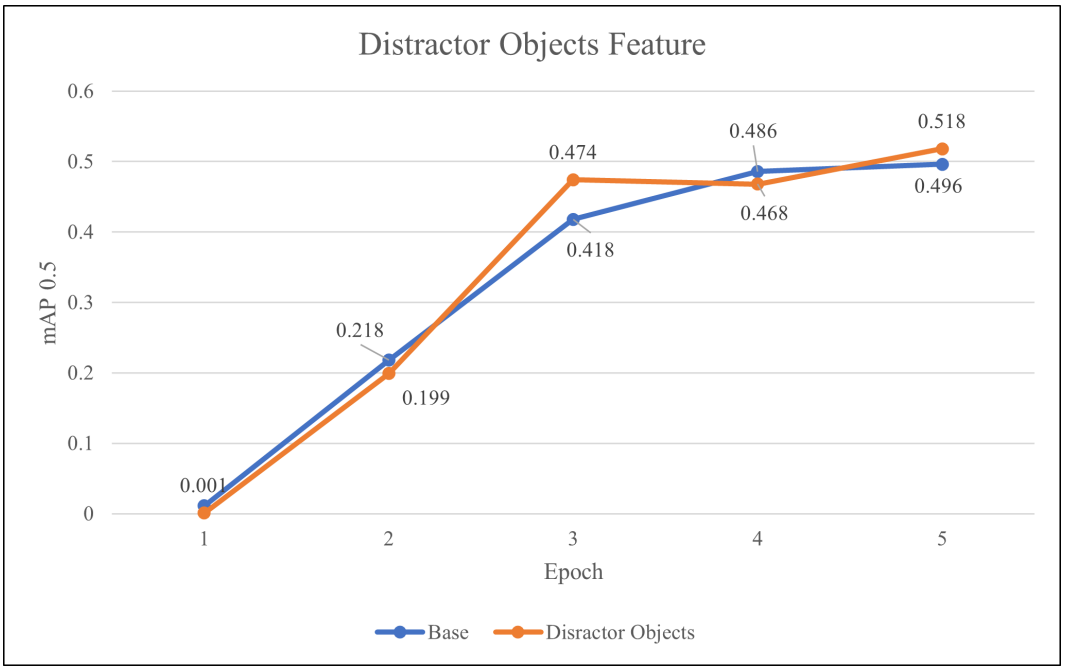


FIGURE 6.15. Training graphs of distractor and base dataset

### 6.2.9. Bird Objects Feature

FIGURE 6.16. Rendering with different 3D bird models

In this dataset, the effect of non-drone objects on performance was examined, similar to the distractor object experiment. Unlike the previous experiment, in this experiment, five different bird objects were randomly placed in each iteration. As seen in Fig. 6.17., there was no significant increase in the mAP value, similar to the distractor object experiment.

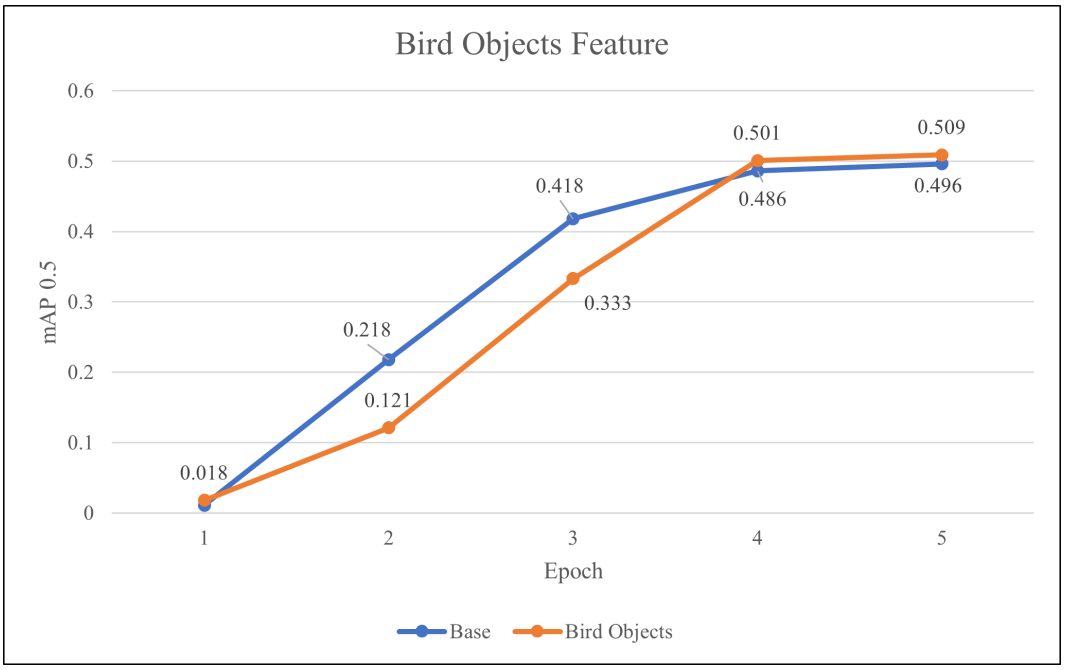


FIGURE 6.17. Training graphs of bird objects and base dataset

### 6.2.10. Noise Feature

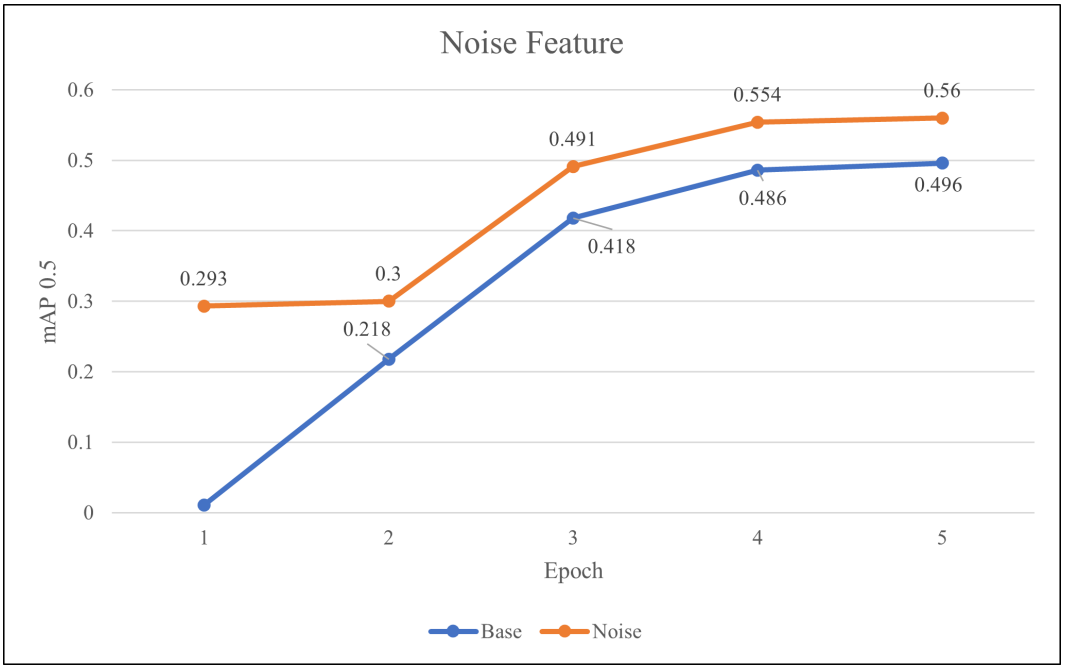


FIGURE 6.18. Training graphs of noise and base dataset

In this experiment, the effect of random noise effect on model performance is examined. To achieve this effect, a film grain effect has been applied, which imitates the noise effect in cameras. This effect greatly increased the performance as seen in Fig. 6.18. . The reason for this can be seen as the random noise approximating the distribution of the rendered image to the real image and helping the blending with the background image.

### 6.2.11. Lens Distortion Feature

FIGURE 6.19. Rendering with lens distortion effect

In this experiment, we will examine the effect of lens distortion on model performance. Lens distortion mimics projection differences caused by camera lenses. It is thought that this effect will produce samples similar to images taken with different camera lenses and increase the performance of the model. As shown in Fig.6.20. , this effect had a negative impact on performance. The reason for this can be considered as the fact that the Drone vs Bird dataset was taken with a single camera type.

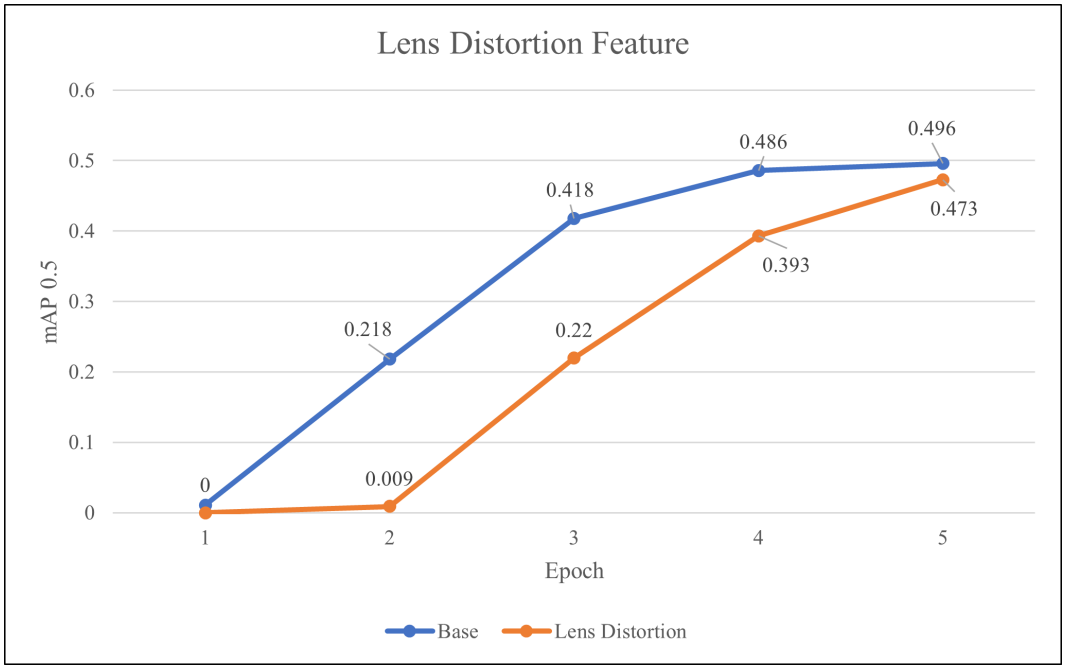


FIGURE 6.20. Training graphs of lens distortion and base dataset

### 6.2.12. Motion Blur Feature

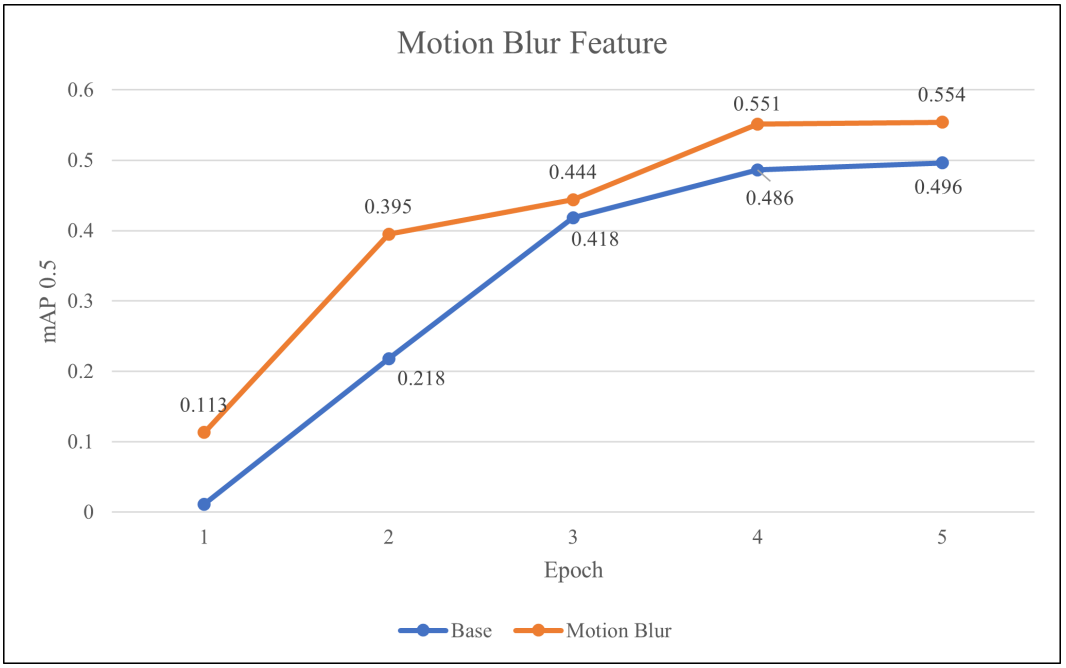


FIGURE 6.21. Training graphs of motion blur and base dataset

In this dataset, the effect of motion blur effect caused by camera shutter speed on model performance has been investigated. This effect causes blurring of drone images and distortion in images taken with real camera. It was predicted that the presence of similar and more diverse samples in the synthetic data would increase the overall performance. Performance has improved overall, as can be seen in Fig. 6.21. . This feature has led to an increase in performance, but when used with other blur methods, it has been observed that it negatively affects performance. That's why this feature is used separately from other blur features.

### 6.2.13. Depth of Field Feature

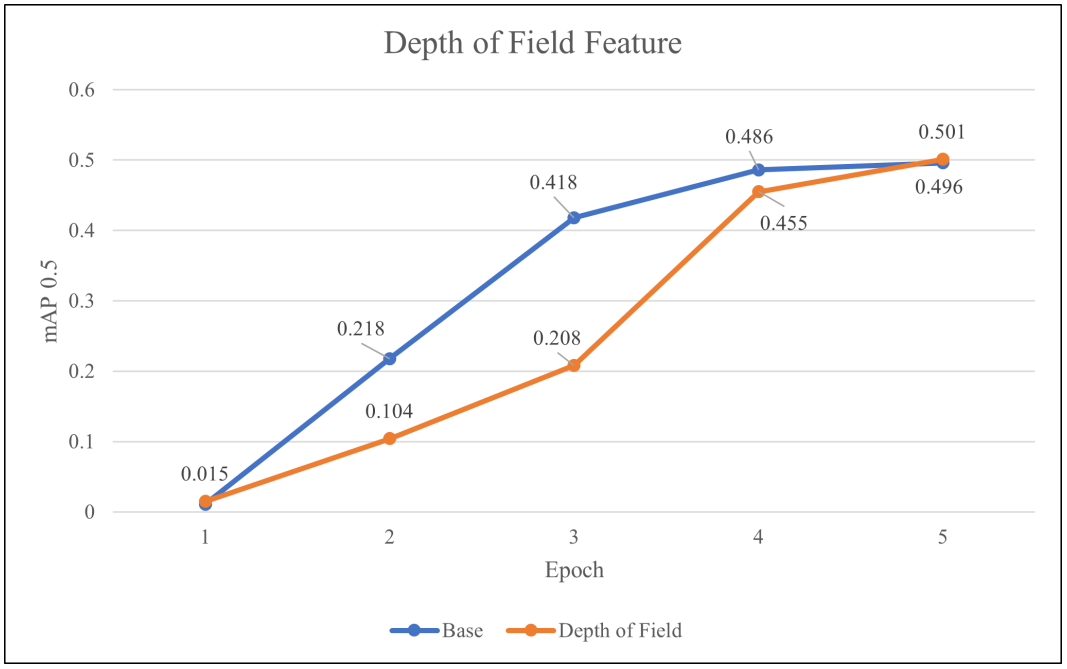


FIGURE 6.22. Training graphs of depth of field and base dataset

In this experiment, the effect of the depth of field effect, which simulates the camera focus, on the model performance was investigated. The results of the Depth of Field implementation can be seen in Fig.6.22. . This effect, which is used in cases where the focus is adjusted correctly, increases the performance because it provides a successful blending, while it decreases the performance in the focus settings independent from the background. Therefore, it did not contribute much to the performance.

### 6.2.14. Gaussian Blur Feature

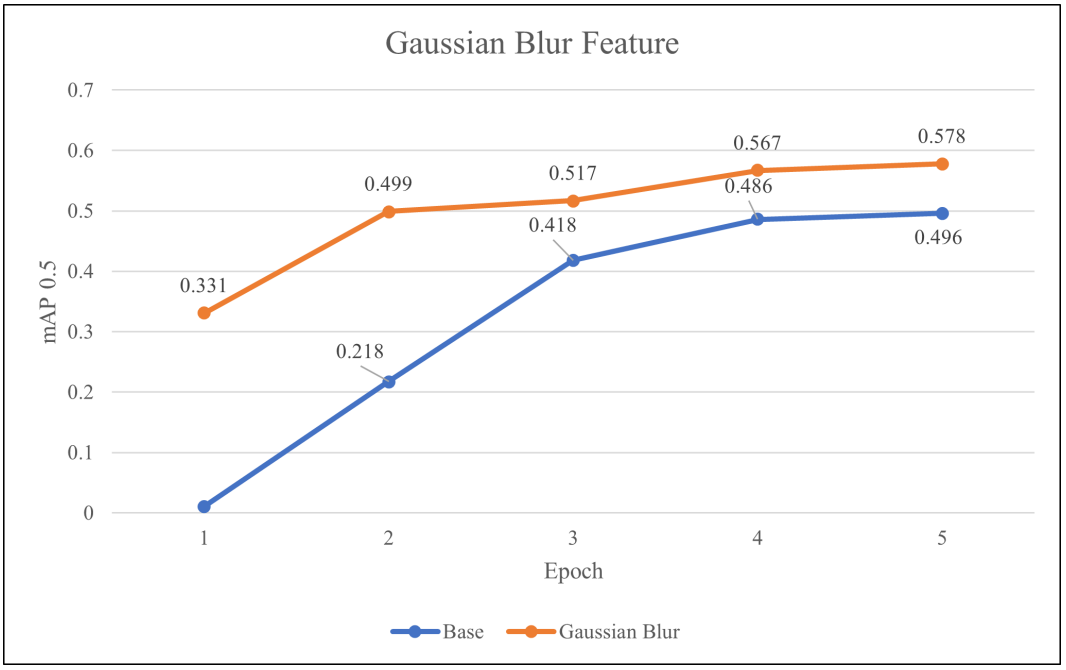


FIGURE 6.23. Training graphs of gaussian blur and base dataset

In this experiment, the effect on model performance was investigated when fixed Gaussian blur is applied to drone objects. In Sec. 5.3., it is explained how the gaussian blur ratio is determined. It has been seen that the blur effect generally increases the performance by adapting the drone models to the background. It can be seen in Fig. 6.23. that the blur effect added to the final render in this implementation leads to a great increase in performance even though it is independent of the background.

### 6.2.15. Inference Blur Feature

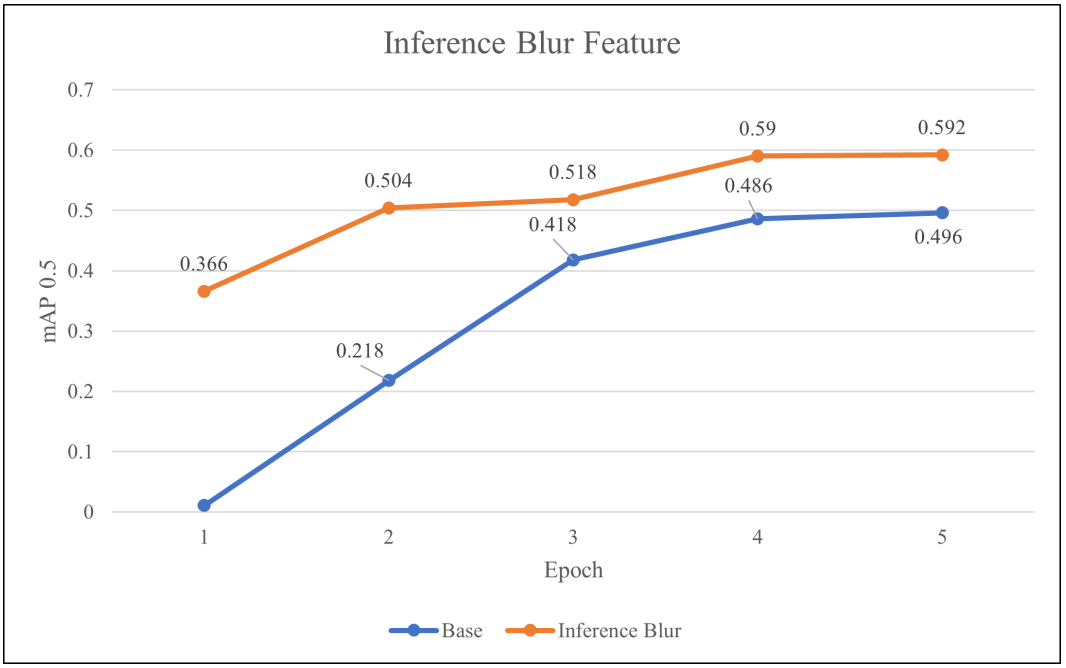


FIGURE 6.24. Training graphs of inference blur and base dataset

In this experiment, after applying the gaussian blur effect, inference was taken for each background using the model trained with real data, and the amount of blur was adjusted according to these inference scores. In this way, the blending of the drone model in accordance with every background is provided. As can be seen in Fig. 6.24. , the model performance has slightly increased compared to the fixed gaussian blur.

### 6.2.16. Adaptive Blur Feature

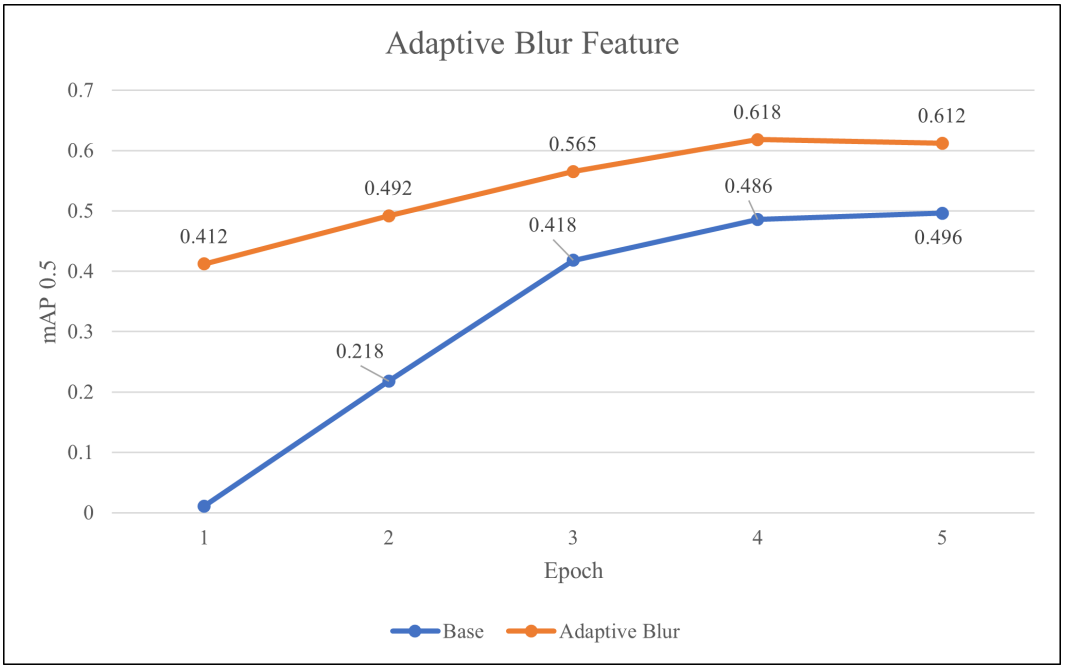


FIGURE 6.25. Training graphs of adaptive blur and base dataset

In this experiment, unlike other blur methods, we adjust the amount of gaussian blur by calculating the background blur. Thanks to the blur given to the drone object according to the sharpness of the background, blending can be done in the appropriate amount for each background. Unlike the inference blur method, this approach is more suitable for synthetic data-driven data generation, as it eliminates the need for a model trained with real data. In addition, as seen in Fig. 6.25. , it has been the feature that provides the highest performance increase in blur methods. Since this feature is the feature that affects the performance the most among all the other features examined, it is a positive indicator that the object detection model can be trained using only synthetic data.

### 6.2.17. Best Dataset

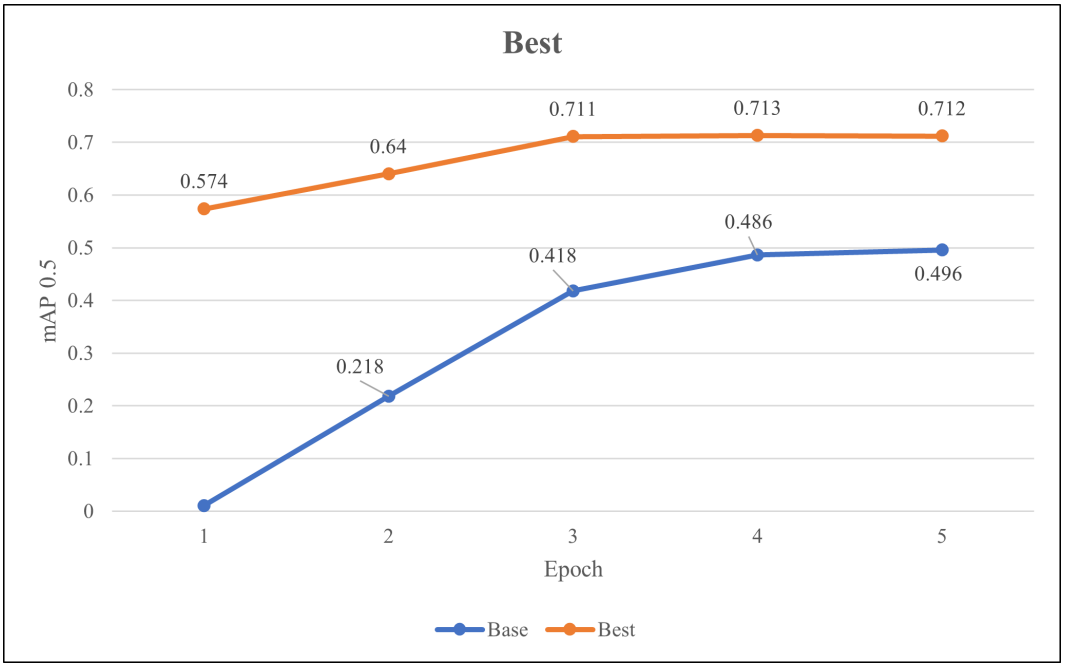


FIGURE 6.26. Training graphs of best and base dataset

In order to create the Best dataset, among all the features mentioned in the previous sections, those that positively affect the performance were used. Although the selected features provide a performance increase alone, they do not always have a positive effect when used together. Therefore, while smaller drones, multiple drones, and noise were used as fixed, two different data sets were created by adding motion blur and adaptive blur, and the ”Best” data set was created by combining these data sets. As can be seen in Fig. 6.26. as a result of the ablation study, with the right features and optimizations selected, the synthetically produced data set showed a performance close to the real data. In the next section, we will call this dataset, which we will use as a synthetic dataset, as ”Synthetic Dataset”.

## 6.3. Training Results

In this section, we will discuss the experimental results we have done in different configurations using the ”Real Dataset” and the ”Synthetic Dataset” that we created as a result of the ablation study.

First, the performance of the real data was evaluated by performing the training with real data, and the performance of the real data with different subsampling values was examined, and the number of images that gave the highest performance was searched. In the same way, the effect of the amount of synthetic data on the performance was examined and the boundary performances of both data sets were measured.

Then, experiments such as mixing real and synthetic data sets at different rates and fine-tuned training on each other were carried out, and it was investigated how to use synthetic data optimally with real data.

In all experiments, the configuration mentioned in Sec. 6.1. was used, and experiments in which synthetic and real data were used together were carried out,mAP scores of both real and synthetic models calculated on ”Validation Set” which only contains real data.

### 6.3.1. Impact of Amount of Real Data on Model Performance

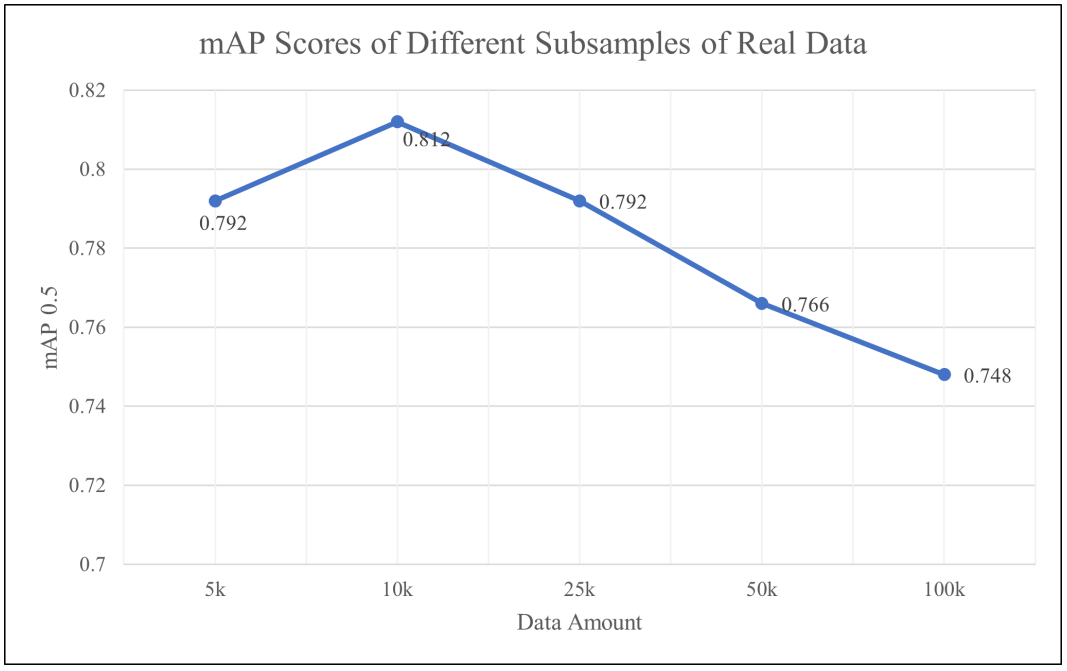


FIGURE 6.27. mAP scores of models trained with different amount of ”Real Data”

In general, as the amount of data increases in deep learning studies, model performance increases in parallel. The reason for this situation is that with the increasing amount of data, more information is provided to the model. As seen in Fig 6.27., on the contrary to this situation, the highest performance was achieved when the amount of data was reduced to ”10k”, and the performance decreased when the amount of data was reduced further. In the experiment, the data reduction process is done by the method called ”subsampling”. A certain amount (1/2, 1/4, 1/10, 1/20) was taken from the data set consisting of 100k data and new datasets with a lower number of images were created. The reason why this process increases the performance is that the Drone vs Bird dataset used consists of videos and the uniqueness rate of the frames obtained from the video is low. As can be seen in Fig. 6.28., we see that the validation loss starts to increase after a certain epoch in trainings containing more than 25k data and the model starts to overfit.

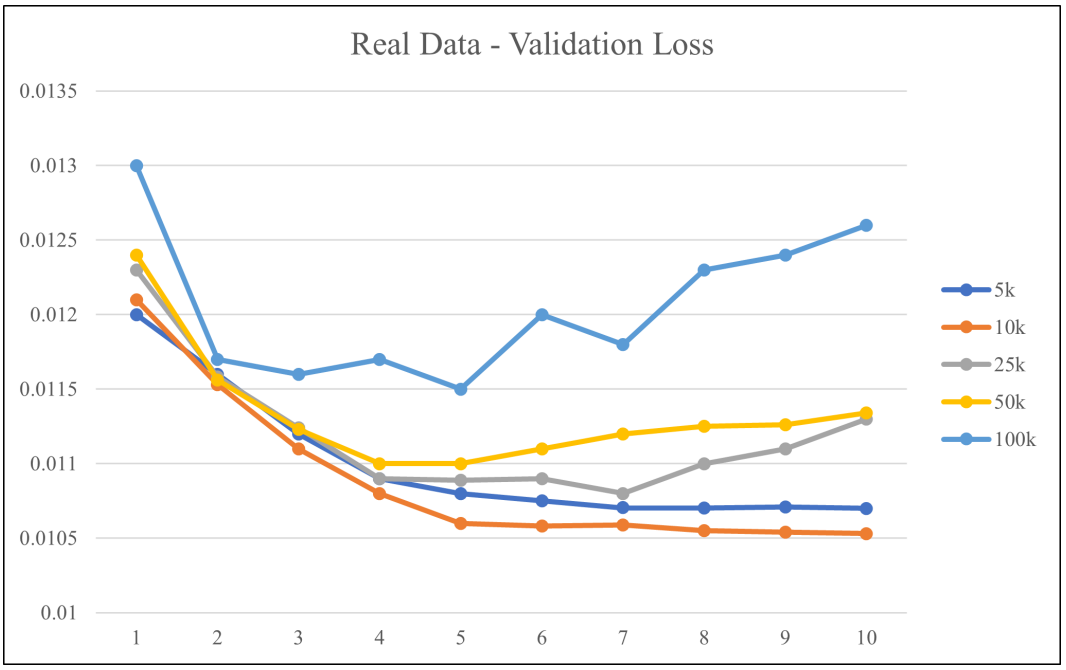


FIGURE 6.28. Validation loss graph.With examples containing more than 25k data, the validation loss in training starts to increase after a certain epoch.

### 6.3.2. Impact of Amount of Synthetic Data on Model Performance

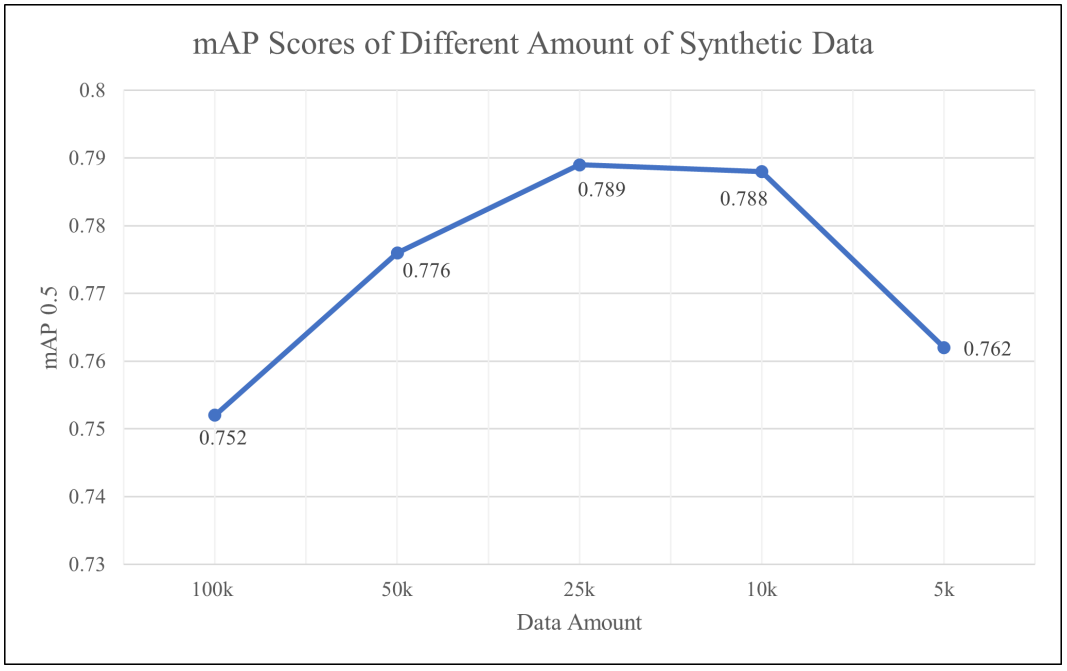


FIGURE 6.29. mAP scores of models trained with different amount of ”Synthetic Data”

All of the synthetic data experiments in the previous section were made with 10k images in order to conduct experiments quickly. In this experiment, the performance of datasets containing different amounts of images using the ”Best Dataset” features are analyzed. As can be seen in Fig 6.29. , when the amount of data is increased from 10k to 25k, there is a slight increase in performance. It has been observed that the performance decreases when the amount of data is reduced or increased further. The reason for this situation can be seen as the low amount of data does not provide enough information to the model and a large amount of data does not contain new information.

### 6.3.3. Final Trainings

After examining the performance of synthetic and real data according to the amount of data, experiments were carried out to examine the effect of using real and synthetic data together

on performance. The experiments carried out for this purpose are grouped under three headings; basic training, mixed training and finetune training. Base training is done using only synthetic or real data. The results of these trainings are used to compare other trainings. Mixed training is training done by mixing synthetic and real data at different rates. Finetune trainings were carried out by finetuning training with different amounts of real data on the synthetic model trained with different amounts of synthetic data. These experiments and score information are shown in Table 6.2. .

TABLE 6.2. Experiments with different proportions of real and synthetic data

| Experiment Name | Synthetic Data Count | Real Data Count | mAP 0.5 | mAP 0.5-0.95 |
|---|---|---|---|---|
| Real Only Training | 10k | 0 | 0.812 | 0.344 |
| Synthetic Only Training | 0 | 10k | 0.712 | 0.292 |
| Real Synth Mix 1 | 10k | 10k | 0.796 | 0.301 |
| Real Synth Mix 2 | 10k | 25k | 0.811 | 0.309 |
| Real Synth Mix 3 | 25k | 10k | 0.789 | 0.318 |
| Real Synth Mix 4 | 25k | 25k | 0.809 | 0.297 |
| Real Finetune over Synth 1 | 10k | 10k | 0.817 | 0.338 |
| Real Finetune over Synth 2 | 10k | 25k | 0.813 | 0.345 |
| Real Finetune over Synth 3 | 25k | 10k | 0.844 | 0.351 |
| Real Finetune over Synth 4 | 25k | 25k | 0.815 | 0.348 |
| **Real Finetune over Synth 5** | **50k** | **10k** | **0.865** | **0.355** |

The comparison of trainings using only real and only synthetic data are shown in Fig 6.30. . As can be seen, the real data started from a very high mAP value and increased slightly, while the synthetic data started from a low value and increased gradually and achieved a result comparable to the real data. The reason for this can be seen as the fact that the real data training started on the COCO backbone trained with real images is easier to learn at the end of the first epoch. The domain gap between the synthetic data and the real data delayed the learning of the synthetic model somewhat and caused it to finish with a lower mAP. As

a result of these experiments, our goal is to exceed the 0.812 mAP value obtained with real data.

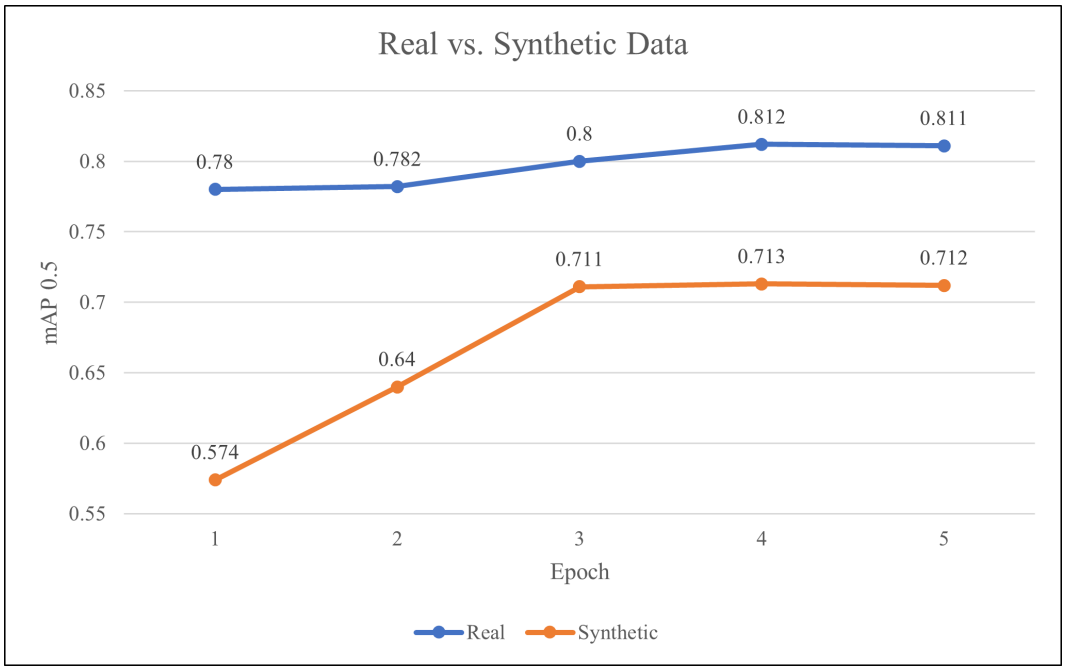


FIGURE 6.30. Comparison of trainings with only real and only synthetic data

As the first method, we trained with data sets created by mixing synthetic and real data at different rates. As seen in Fig 6.31., all of the experiments were below the real data baseline. In addition, the model performance increased as the amount of real data increased in mixing ratios and decreased as the amount of synthetic data increased. This can be explained by the fact that the model trained as a result of the domain gap between real data and synthetic data created uncertainty while learning drone features. As a result of the model's encounter with two different drone representations, it caused an oscillation in the graph during the training and reduced the performance of the training.

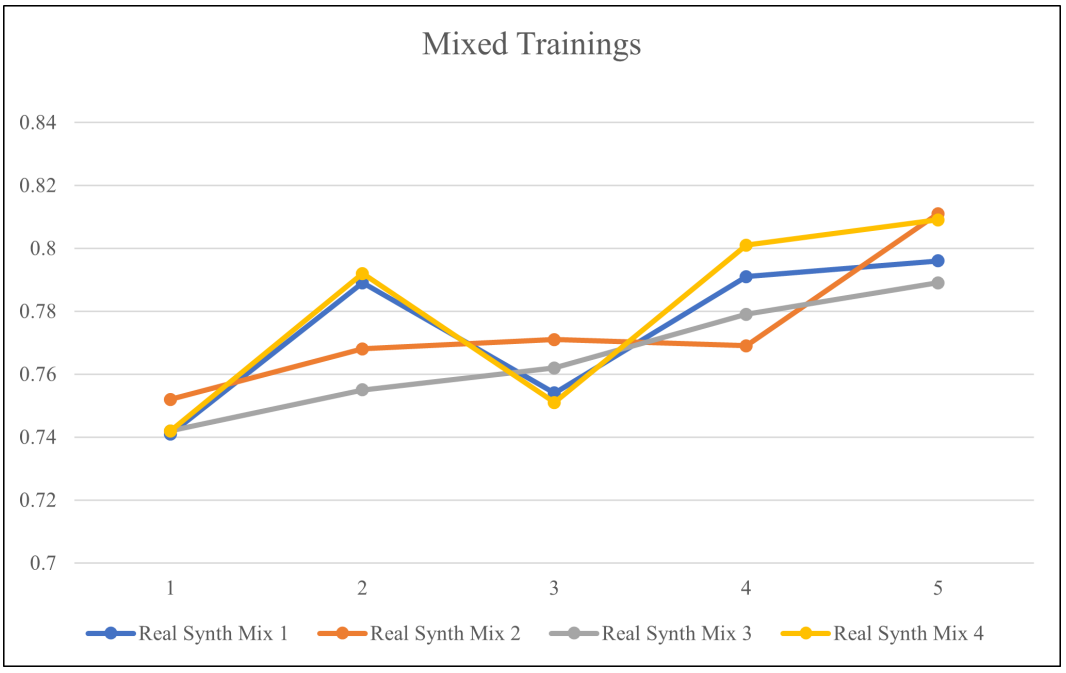


FIGURE 6.31. Comparison of trainings with mixed datasets

As a second method, experiments were carried out by fine-tuning the models that were first trained using synthetic data with different amounts of real data. Figure 6.32. shows real data fine-tuning tutorials using synthetic data as a backbone. This method, using the synthetic model, provides information about the drone features before starting the training with real data, and the training starts from higher mAP values. In addition, the training is completed with a higher mAP value, thanks to a backbone with different content from the real data. As can be seen from experiments 2 and 4, increasing the amount of real data fine-tuned in this method decreases the performance. The reason for this is that when the amount of real data in the training on the synthetic model is more than the amount of synthetic data, the features learned from the synthetic model are forgotten and over-fitted to the real data. Therefore, in experiment number 5, fine-tuning with 10k data on the synthetic data model containing 50k images gave the best results.

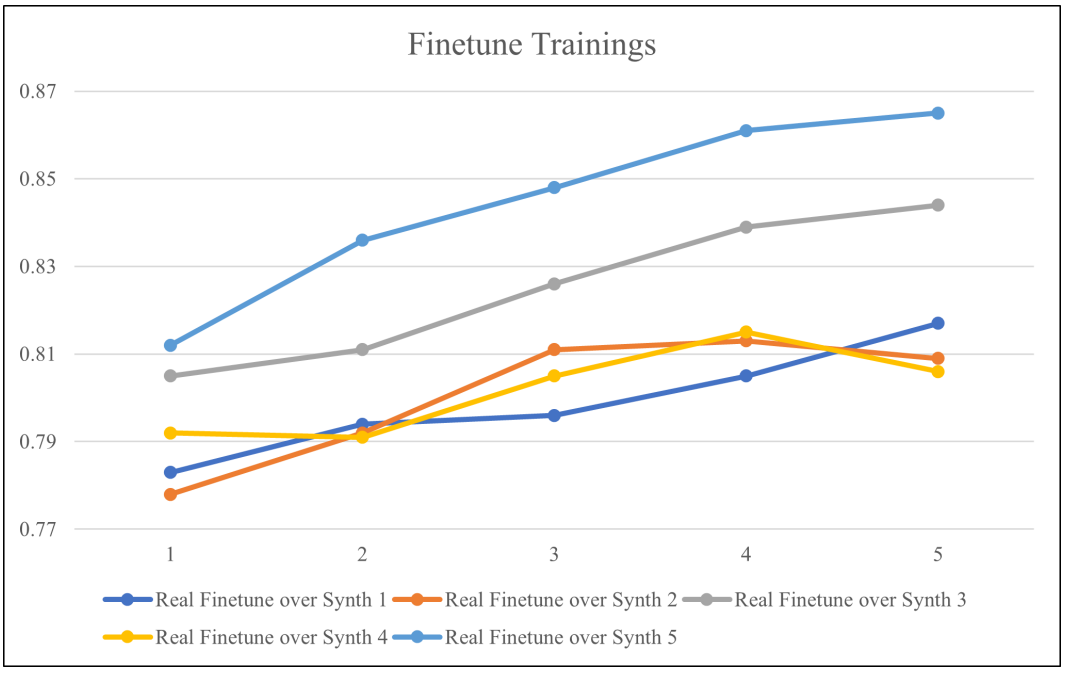


FIGURE 6.32. Comparison of trainings with synthetic models finetuned with real data

As a result of all the trainings, it has been seen that the model trained with synthetic data gives results that can be compared with the real model. This success of synthetic data also reduced the performance of real data in case of mixing with real data. This experiment has the possibility of giving better results when tested with a more realistic data set with a lower domain gap. Finally, models trained with different amounts of synthetic data were used as backbones and fine-tuned with different amounts of real data. It has been seen in these experiments that the ratio of real and synthetic is of great importance when making fine adjustments. In cases where the amount of synthetic data used in the backbone is more than the real data, the performance increased, and the most successful result was obtained by using five times the synthetic data of the real data. This result is approximately seven percent higher than the baseline value of 0.812 mAP. Although this increase seems small, it is very difficult to increase the performance of a successful model that has already been trained with real data. Therefore, as a result of the experiments, it can be said that synthetic data is a very successful method when used both alone and with real data.

# 7. CONCLUSION

The performance of deep learning methods is increasing day by day. The biggest obstacle in front of this constantly developing technology is the difficulty of accessing data. Although deep learning methods have constantly evolving architectures, the performance of the trained models does not increase at the same rate because these methods require large amounts of labeled data.

Traditionally, datasets are collected manually by humans. These collected data are then labeled by people and used to train the models. This process is expensive and very time-consuming because it is done by human hands. These manually labeled data are prone to error and human bias. In addition, the difficulty of collecting data varies according to the domain of the problem to be solved, and collecting data in some domains includes privacy problems or is impossible.

As a solution to these problems in data collection, the idea of creating these datasets synthetically has been suggested. This method has the opportunity to both solve the problems caused by the real data and create various and unlimited data belonging to the targeted domain. Drone detection, which is one of the areas where it is difficult to collect real data, is one of the most suitable problems to be solved with synthetic data. Drones can be found in any environment due to their nature, and they can be easily confused with objects such as birds and airplanes. Therefore, it has been suggested that a data set that covers a large number of domains and contains many examples can only be created with synthetic data.

In this thesis, a synthetic dataset that will be used to train a deep learning model for drone object detection and an infrastructure that can create this dataset in a controlled manner has been studied. This designed infrastructure aims to create realistic and labeled images as a result of placing 3D drone objects on 2D backgrounds. To achieve this, a controlled randomization infrastructure, randomized scenes created as a result of this structure, and a system pipeline that adds post-processing effects on the generated images have been developed.

This pipeline can create randomized environments thanks to a background dataset containing many images on certain themes and a drone set containing six different drone models. The created scenes are rendered quickly thanks to the game engine infrastructure, and the drone models are successfully blended into the background with the post-processing effects added to the rendered image.

In order to investigate the optimization of this parametric system created on the game engine, an ablation study was required. In the ablation study, the effect of the parameters of the system on the trained model performance was investigated. Thus, the features that gave the highest performance increase in synthetic data were found and the final synthetic data set was created in line with these features. As a result of the ablation study, it was noticed that the blur effect is the most affecting the blending and thus the synthetic data performance. Therefore, various methods have been tried for the blurring process, and the ”Adaptive Blur” algorithm, which applies blur depending on the sharpness value of the arc, was chosen as the implementation with the highest performance.

The performance of the system created with the experiments using the optimized data set called ”Synthetic Dataset” and the Drone vs Bird dataset consisting of real drone images was measured. In these experiments, training was carried out with synthetic and real data alone, by mixing and training on each other. As a result of all these experiments, it has been seen that the synthetic data optimized according to the domain can show a drone detection performance that can be compared with the real data. In addition, it has been observed that the performance of the real model increases to a certain extent if the synthetic model is fine-tuned with real data. The importance of the ratio of the two data sets was observed when fine-tuning, and experiments with more synthetic data than real data yielded more successful results.

As a result of the developed system, ablation study and final trainings, it has been seen that creating a synthetic data set to train the drone detection model is a fast and very successful method. The presented method can be used not only in the field of drone detection but also

in all object detection problems, and the use of synthetic data in object detection has proven to be the right decision.

# REFERENCES


[1] Brian K. S. Isaac-Medina, Matt Poyser, Daniel Organisciak, Chris G. Willcocks, Toby P. Breckon, and Hubert P. H. Shum. Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark. *CoRR*, abs/2103.13933, **2021**.

[2] Study: Digital experiences in the physical world, **2020**.

[3] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu , changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, **2020**. doi:10.5281/zenodo.4154370.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., **2012**.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, **2016**.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. **2016**.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587. **2014**.

[8] Henry S. Baird. *Document Image Defect Models*, pages 546–556. Springer Berlin Heidelberg, Berlin, Heidelberg, **1992**. ISBN 978-3-642-77281-8. doi:10.1007/978-3-642-77281-8_26.

[9] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, **2010**. ISSN 1530-888X. doi:10.1162/neco_a_00052.

[10] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images, **2016**.

[11] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects, **2018**.

[12] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *Proceedings of the British Machine Vision Conference 2017*. **2017**.

[13] Georgios Georgakis, Arsalan Mousavian, Alexander C. Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes, **2017**.

[14] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, **2017**. doi:10.1109/cvpr.2017.492.

[15] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing, **2018**.

[16] Abdulrahman Kerim, Cem Aslan, Ufuk Celikcan, Erkut Erdem, and Aykut Erdem. NOVA: Rendering Virtual Worlds with Humans for Computer Vision

Tasks. *Computer Graphics Forum*, 0(0):1–13, **2021**. ISSN 14678659. doi: 10.1111/cgf.14271.

[17] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?, **2017**.

[18] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games, **2016**.

[19] Chaitanya Mitash, Kostas E. Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation, **2017**.

[20] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning?, **2016**.

[21] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone, **2021**.

[22] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:23–30, **2017**. ISSN 21530866. doi:10.1109/IROS.2017.8202133.

[23] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data, **2020**.

[24] João Borrego, Atabak Dehban, Rui Figueiredo, Plinio Moreno, Alexandre Bernardino, and José Santos-Victor. Applying domain randomization to synthetic data for object category detection, **2018**.

[25] Diego Marez, Samuel Borden, and Lena Nans. Uav detection with a dataset augmented by domain randomization. In *Geospatial Informatics X*, volume 11398, pages 39–50. SPIE, **2020**. doi:10.1117/12.2558864.

[26] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:1310–1319, **2017**. ISSN 15505499. doi:10.1109/ICCV.2017.146.

[27] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views, **2015**.

[28] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, **2015**. ISSN 1077-3142. doi:10.1016/j.cviu.2014.12.006.

[29] Angelo Coluccia, Alessio Fascista, Arne Schumann, Lars Sommer, Marian Ghenescu, Tomas Piatrik, Geert De Cubber, Mrunalini Nalamati, Ankit Kapoor, Muhammad Saqib, et al. Drone-vs-bird detection challenge at ieee avss2019. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–7. IEEE, **2019**.

[30] Alejandro Rodriguez-Ramos, Javier Rodriguez-Vazquez, Carlos Sampedro, and Pascual Campoy. Adaptive inattentional framework for video object detection with reward-conditional training. *IEEE Access*, 8:124451–124466, **2020**.

[31] Nan Jiang, Kuiran Wang, Xiaoke Peng, Xuehui Yu, Qiang Wang, Junliang Xing, Guorong Li, Jian Zhao, Guodong Guo, and Zhenjun Han. Anti-uav: A large multi-modal benchmark for uav tracking. *arXiv preprint arXiv:2101.08466*, **2021**.

[32] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, **1986**. doi: 10.1109/TPAMI.1986.4767851.

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, **2015**.

*The presented work was used in the IEEE AVSS 2021 Drone vs. Bird competition.The model trained showed the best performance and brought first place in the competition.*

# CURRICULUM VITAE

**Credentials**


| | | |
|---|---|---|
| Name,Surname | : | Kamil Anıl Özfuttu |
| Place of Birth | : | Ankara,Turkey |
| E-mail | : | anilozfuttu@gmail.com |
| Address | : | Computer Engineering Dept., Hacettepe University<br>Beytepe-ANKARA |


**Education**

| | |
|---|---|
| BSc. : | Electrical and Electronics Engineering<br>TOBB University, Turkey |
| MSc. : | Computer Animation and Game Technologies<br>Hacettepe University, Turkey |

**Foreign Languages**

English

**Work Experience**

OBSS Technologies (2019-) - Simulation & ML Engineer
GFDS Inc. (2019-2020 ) - Unity Developer
ROBOTSAN (2016-2019) - R&D Engineer