SEPTEMBER 2021 Ph.D. in Electrical and Electronics Engineering ALİ EMRE ÖZTÜRK

REPUBLIC OF TURKEY

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

DEVELOPMENT OF RW-UAV DETECTION MODEL USING DEEP LEARNING TRAINING WITH A SYNTHETIC DATASET

Ph.D. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

ALİ EMRE ÖZTÜRK

SEPTEMBER 2021

DEVELOPMENT OF RW-UAV DETECTION MODEL USING DEEP LEARNING TRAINING WITH A SYNTHETIC DATASET

Ph.D. Thesis

in

Electrical and Electronics Engineering

Gaziantep University

Supervisor

Prof. Dr. Ergun ERÇELEBİ

by

Ali Emre ÖZTÜRK

September 2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ali Emre ÖZTÜRK

ABSTRACT

DEVELOPMENT OF RW-UAV DETECTION MODEL USING DEEP LEARNING TRAINING WITH A SYNTHETIC DATASET

ÖZTÜRK, Ali Emre
Ph.D. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ergun ERÇELEBİ
September 2021
95 pages

The use of rotary wing unmanned aerial vehicles (UAV) is increasing day by day and their accessibility is getting easier. As a disadvantage of easy accessibility and usability, UAV technology threatens security and personal areas. This technology, which is easy to access, should be detected with a cheap and easy-to-install detector. The aim of the study is to develop the UAV detection sensor using camera and machine learning based systems. Deep learning method has been used in sensor technology, which has shown successful classification results recently. Artificial intelligence trained with deep learning method shows effective results in terms of performance. On the other hand, artificial intelligence (AI) training needs a lot of data. To overcome this obstacle, the data set needed by the deep learning algorithm was produced synthetically using game engine software. The original aspect of the study is that the artificial intelligence trained with synthetically produced images was tested on real data and achieved 90% success. The success of AI, which is directly trained with synthetic data and tested with real data, is very low. The reason for this is the difference between the synthetic and real image. To minimize this difference, 7 different experiments were designed for different feature extraction algorithms. One of the experiments is the Corner Detection and Nearest Three-point Selection (CDNTS) feature extraction layer that we originally developed. The CDNTS layer classified the real data with 90% success.

Key Words: Image Classification, Deep Learning, CDNTS, Synthetic Image.

ÖZET

SENTETİK VERİSETİ İLE DERİN ÖĞRENME EĞİTİMİ KULLANARAK DÖNER KANAT İHA TESPİT MODELİNİN GELİŞTİRİLMESİ

ÖZTÜRK, Ali Emre
Doktora Tezi, Elektrik-Elektronik Mühendisliği
Danışman: Prof. Dr. Ergun ERÇELEBİ
Eylül 2021
95 sayfa

Döner kanatlı insansız hava araçlarının (İHA) kullanımı her geçen gün artmakta ve erişilebilirliği kolaylaşmaktadır. Kolay erişilebilirlik ve kullanılabilirliğin dezavantajı olarak İHA teknolojisi güvenliği ve kişisel alanları tehdit etmektedir. Erişilebilirliği kolay olan bu teknoloji ucuz ve kolay kuruluma sahip bir algılayıcı ile tespit edilmelidir. Çalışmanın amacı kamera ve makine öğrenmesi tabanlı sistemler kullanılarak İHA tespit algılayıcısının geliştirilmesidir. Algılayıcı teknolojisinde son zamanlarda başarılı sınıflandırma sonuçları gösteren derin öğrenme metodu kullanılmıştır. Derin öğrenme yöntemi ile eğitilen yapay zekalar başarım olarak etkili sonuçlar gösterir. Diğer taraftan yapay zeka (YZ) eğitiminde çok fazla veriye ihtiyaç duyar. Bu engeli aşmak için derin öğrenme algoritmasının ihtiyaç duyduğu veri setini oyun motoru yazılımları kullanarak sentetik olarak üretildi. Çalışmanın özgün yönü sentetik olarak üretilen imgeler ile eğitilen yapay zekanın gerçek veriler üzerinde test edilerek 90% başarı elde etmesidir. Sentetik veriler ile doğrudan eğitilen ve gerçek veriler ile test edilen YZ'nın başarısı oldukça düşüktür. Bunun sebebi sentetik ve gerçek imge arasındaki gerçeklik farkıdır. Bu farkı en aza indirmek farklı özellik çıkarım algoritmaları için 7 farklı deney tasarlandı. Deneylerden biri bizim özgün olarak geliştirdiğimiz CDNTS özellik çıkarım katmanıdır. CDNTS katmanı gerçek verileri 90% oranında başarılı sınıflandırmıştır.

Anahtar Kelimeler: İmge Sınıflandırma, Derin Öğrenme, CDNTS, Sentetik İmge.

"Dedicated to my family"

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Dr. Ergun ERÇELEBİ for his guidance and support throughout the study. I am thankful for his encouragement and motivation.

I also wish to thank to my father, mother and sister, Mustafa ÖZTÜRK, S. Yıldız ÖZTÜRK and Nesrin ÖZTÜRK, for believing in me.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS

e	Exponential Function
w_{ij}	weight between input and output
b_j	Bias of hidden layer
h_j	Hidden node output
Y_k	Output of hidden layer.
δ_k	Error of node
T_k	Expected output
f'	Derivative of activation function
η	Learning rate
α	Momentum coefficient
T	Template
R	Resulting image
I	Source image
TP	True positive
TN	True negative
FP	False positive
FN	False negative
$L_{cn->ct}$	Vector length
x_{cn}	Selected corner x position
x_{ct}	Target corner x position
y_{cn}	Selected corner y position
y_{ct}	Target corner y position
Vx	Vertical direction
Gy	Direction
f_r	Kernel for smooting
g_s	Spatial kernel

LIST OFABBREVIATIONS

UAV	Unmanned Aerial Vehicle
ML	Machine Learning
GPU	Graphic Process Unit
FW-UAV	Fixed-Wing Unmanned Aerial Vehicle
RW-UAV	Rotary-Wing Unmanned Aerial Vehicle
AI	Artificial Intelligence
DL	Deep Learning
CPU	Central Process Unit
CNN	Convolutional Neural Network
2D	2 Dimension
3D	3 Dimension
CDNTS	Corner Detection and Nearest three-point Selection
BP	Back Propagation
ADAM	Adaptive Moment Estimation
SGD	Stochastic Gradient Descent
RMSprop	Root Mean Square Error Probability
RAM	Random access Memory
VGG	Visual Geometry Group
CAD	Computer Aided Design
RGB	Red Green Blue
RELU	Rectified Linear Units
MD5	message-digest 5
OpenCV	Open-Source Computer Vision
AUC	Area under of Curve
LPF	Low Pass Filter

CHAPTER I

INTRODUCTION

1.1 General

Using cameras as object classification sensors has been a topic of research for many years [1]. These devices are useful sensors for the observing of large areas, thanks to their wide viewing angle and the ability to take images of distant points [1]. Cameras can gain smart abilities to perform the tasks of various sensors with image processing software, machine learning algorithms. Cameras are used as sensors in many areas such as the detection of any object, motion detection, distance detection [2]. The use of cameras in unmanned aerial vehicle (UAV) detection applications, along with image processing-based algorithms, is one of these abilities. One of the systems used in the detection of unmanned aerial vehicles is camera and image recognition [3], based on machine learning algorithms [4].

Algorithms that process camera images using mathematical functions can be examined under the headings of image processing and machine learning (ML). Image processing algorithms compare the similarity ratio of more than one image or some parts of the image by using the specified feature extension functions and work to determine the similarity ratio [5]. ML algorithms are based on determining the weight coefficients of different samples in a cluster. The diversity of the set of examples created for ML training and the number of examples directly affect the success of the ML model [6]. On the other hand, the number of samples is a parameter that increases the ML training time. The development of Graphic process unit (GPU) technology has increased the use of machine learning method deep learning algorithm. GPU hardware can increase machine learning speeds by performing parallel operations. Data diversity and number of data are important for successful machine learning [7]. For this reason, the data set generation stage for machine learning is an important stage that affects the success of learning.

1.2 Thesis Aims and Objectives

The number and types of unmanned aerial vehicles, which are classified under two main headings as Fixed-wing UAV (FW-UAV) and Rotary-wing UAV (RW-UAV), are increasing day by day [8]. Due to the nature of FW-UAVs, they need a long runway for takeoff and landing. However, RW-UAVs can take off and land from an area of $1m^2$ [8]. FW-UAVs can fly longer and carry more payloads than RW-UAVs. However, FW-UAVs are more difficult to pilot than RW-UAVs [8]. RW-UAVs can take off from small areas and are easy to control, making them easy to use anywhere and by anyone. Easy access to RW-UAVs causes the number of aircraft to increase day by day [9]. However, recently, cases such as private security areas, airports, military zones, violations of personal private areas have been experienced by RW-UAVs [10]. Violations endanger the privacy of personal life, the security of military areas and the security of restricted areas [10]. RW-UAVs, which are easily accessible and usable, can be detected by radar-based, voice recognition-based or optical imaging-based studies [11]. In addition, systems using hybrid radar, voice and optic based systems are also used. Therefore, it is important for countermeasures to detect RW-UAVs, which are easy to access and pilot, with easy-to-access and inexpensive equipment such as cameras.

Camera-based systems can be trained using image processing algorithms or machine learning algorithms, and because of this training, they can perform image classification [12]. Current technologies prefer ML algorithms for the success of the system. In systems containing ML algorithms, the variety and number of data sets are important for the success of the learning process, that is, the training of artificial intelligence (AI) [13]. AI training is to determine the weighting coefficients of the dataset set mathematically and to create a general weight function for the whole dataset [14]. As the data set grows and diversifies, this math-based operation turns into a mathematical function that becomes more difficult and time-consuming to solve. The calculation of the weight coefficient's function is to determine the function between the training data set and the expected output [14]. The general structure that determines the weight coefficients consists of deep learning (DL) networks, which is an AI method, the parameters and optimizers used in these networks [15]. Applications of DL networks are not suitable for Central Process Unit (CPU)-based systems, they are used in GPU-based systems. Due to the low number

of cores of CPU-based computer architectures, parallel processing speeds are slow [16]. Therefore, applications of ML models with image processing and simple network structure have been made in CPU-based systems. With the development of GPU-based computer systems, models with a complex network structure based on the Convolutional Neural Network (CNN), called deep learning, have been begun to be used. Thanks to the parallel processing capability of GPU hardware, more data has been started to process in a shorter time compared to classical methods [16].

As the structure of ML networks becomes more complex and deeper, they need more data for successful training [16]. This dependence on the amount and variety of data is the disadvantage of deep learning networks training. Therefore, data collection is one of the most important stages of deep learning-based studies. The use of deep learning networks is not suitable for solving problems where the number and variety of data are limited or absent. In the literature, in DL-based training studies with limited data, data is increased by using a method called data argumentation [17]. In addition, it is another method to increase the number of data set elements by adding synthetic images obtained from simulations and game engines into the data set at a certain rate [18]. Both methods are limited in allowing a small number of data to be used in deep learning algorithms [19].

The aim of this study is to detect RW-UAVs with camera-based sensors. In the study, it is aimed to transform the sky-watching cameras into sensors that detect RW-UAVs. It is aimed to produce a DL-based AI model capable of RW-UAV and bird classification since it frequently roams in the sky in birds as well as RW-UAVs. In the study, a camera-based AI model training that can classify RW-UAV and bird images using DL networks are completed. RW-UAV and bird classes are very difficult to detect and photograph because they are objects in motion in the air. Therefore, it is difficult to create a dataset of these objects by photographing enough to training in a DL-based network. The difficulty of constructing the RW-UAV and bird dataset led to the search for the answer to the following research question. “Can the artificial intelligence training-image dataset be synthetically produced as needed?” and “Can an artificial intelligence model trained with synthetic data be used to classify a real-image dataset?”.

To answer the research question, 2-dimension (2D) RW-UAV and bird images were produced using RW-UAV and bird 3-dimension (3D) models and game engine. RW-UAV and bird 2D images produced in 3D models and Game engine are defined as synthetic images. Images collected from the real world by photographing real RW-UAV and birds are called real images. The image dataset required to training the AI model was produced in a city that was completely produced and designed using the unity game engine. Real image Classification success has been measured and reported by testing CNN-based, DL models trained using synthetic data. Although the AI models trained with fully synthetic images failed on real data, 7 different experiments have been designed. These 7 different experiments are the use of various feature extraction algorithms in harmony with DL-based models. The feature extraction algorithm called corner detection and nearest three-point selection (CDNTS), used in one of the 7 experiments, is the algorithm we developed. The CDNTS layer has been designed by us and constitutes the original aspect of the study. The CDNTS layer has significantly increased the performance of DL-based AI networks.

1.3 Thesis Outline

This thesis is organized in four chapters besides this chapter and ends with references; each chapter is described as follows:

Chapter Two: covers the background required for machine learning and CNN based deep learning networks.

Chapter Three: includes the methods of generating synthetic data with the game engine and the creation of the dataset to be used in the training of DL networks, the method of determining the training strategy of DL based networks, the design of the experiments to be used in the training phase.

Chapter Four: includes a systematic review of the comparison of success rates of experiments, reporting of results of time costs.

Chapter Five: presents the conclusions, and recommendations for future work.

CHAPTER II

THEORICAL CONCEPT OF DEEP LEARNING AND IMAGE CLASSIFICATION

2.1 Machine Learning

Machine learning emerged with the aim of using computers like humans. Studies can computers learn like humans? focused around seeking an answer to the question [20]. The purpose of ML training is to reveal the rules between input and output [19]. While classical programming creates the output according to the rules and inputs defined by the developer, ML models mathematically describe the relationship between input and output. As shown in Figure 2.1, in Classical programming, the inputs are rules and data, and the output is the answers [19]. In machine learning training, inputs are data and answers, and outputs are rules. Therefore, the number and variety of data that ML models use to create rules directly affect success. A basic ML training consists of data collection, data preparation, feature extraction, and testing.

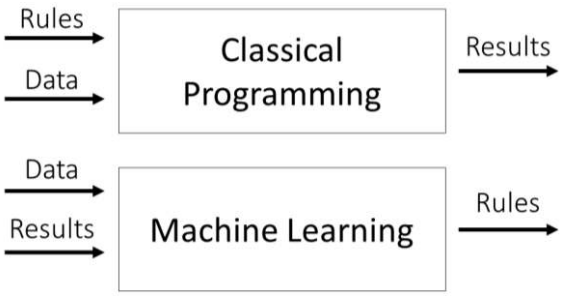

Figure 2.1 Representation of difference between classic programming and machine learning.

2.2 Artificial Neural Network

The aim of AI studies is basically to imitate human intelligence and to develop computer algorithms that can do the work that humans do. In the 1950s, Alan Turing

published a study seeking an answer to the question of whether a computer could think [96]. As shown in Figure 2.2, Artificial intelligence covers the subject of machine learning, and machine learning covers the subject of deep learning. From artificial intelligence to deep learning, the scope narrows, and the topics become specialized.

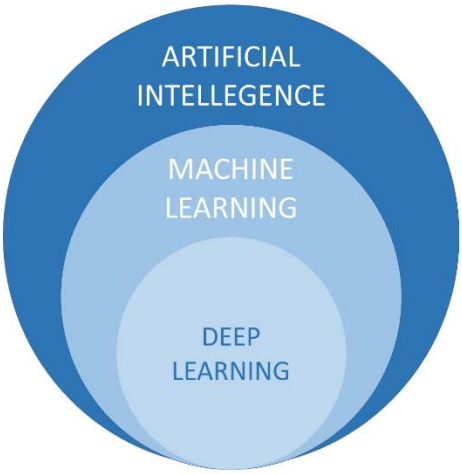

Figure 2.2 Location of Deep Learning in artificial intelligence technology.

Between the 1950s and 1980s, experts thought that the rules of a human-level artificial intelligence to process information could be programmed manually. This approach is called symbolic artificial intelligence [19]. Although there are successful programs developed using this approach, machine learning algorithms have emerged instead of the symbolic artificial intelligence approach in the following years. Machine learning approach has started to be used in solving complex problems.

Artificial intelligence is a computer-based system that tries to imitate many actions that people perform in daily life, such as classification, object detection. Structures consisting of algorithms that try to imitate the human brain are called neural networks. The basic operation of artificial neural networks is to calculate the most suitable mathematical model between input and output data with weight coefficients. For each element of the data input that has more than one element, it is converted into a score showing the input and output relationship. This score between each perceptron is called weight. This calculation is done for each cell called a perceptron. Perceptron combines to form hidden layers, and hidden layers combine to form the model. An artificial neural network cell consists of 5 parts: inputs, weights, summation function, activation function, and outputs.

Inputs: The inputs are the training data that make up the dataset. Each element of the data is processed. Considering that the data is an image, the value of each pixel value is a data element.

Weight: It is the coefficient by which each data will be multiplied before entering the perceptron. The magnitude of this coefficient determines the effect of the data on the perceptron.

Summation Function: It is the function that provides the summation process of the data that comes by multiplying with the weights before entering the perceptron.

Activation Function: It is the function responsible for the filter that will be applied on the data that will pass the total value from the previous process to the next stage. At this stage, it is checked whether the total value is greater than a certain value. Shown in Figure 2.3 are graphs of Sigmoid RELU, Hyperbolic Tangent and Leakly ReLU activation layer functions.

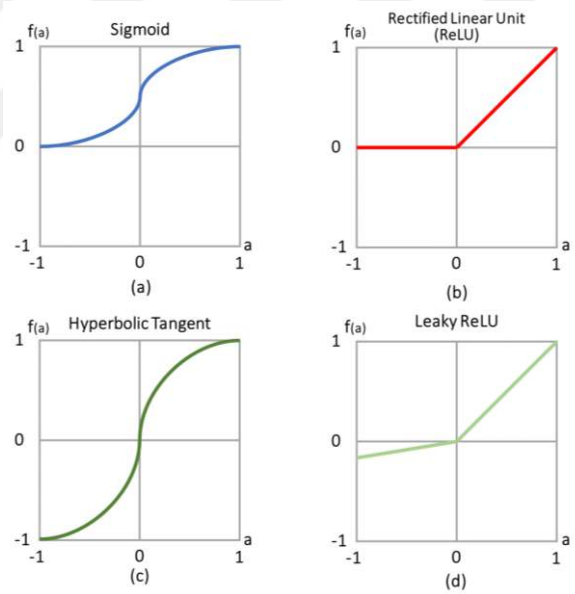

Figure 2.3 Activation function graphs, (a) Sigmoid (b) ReLU (c) Hyperbolic Tangent (d) Leakly ReLU

$$Sigmoid: a = \frac{1}{1 + e^{-z}} \tag{2.1}$$

$$ReLU: a = max(0, z) \tag{2.2}$$

$$Tanh: a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.3}$$

$$Leaky\ ReLU: a\ =\ max(0.01.z, z) \tag{2.4}$$

Output: It is the next step after the activation layer. A model with multiple data inputs can be converted to a single output cell.

2.3 Back-Propagation Algorithms

The back propagation (BP) algorithm is an algorithm defined for determining weights in training multi-layer perceptron [21]. The first step in the design process of artificial neural networks is to determine the number of weights of the artificial neural network and the initial values of these weights. The initial values of these values can also be determined by giving a completely 0 value, or by giving a random float number between 0 and 1. In studies, it is generally preferred to determine by giving random values. AI training algorithms try to find coefficients, called weights, of the mathematical function between inputs and outputs. AI algorithms use the BP algorithm with many iterations to find the most accurate weights [22]. After each iteration, the weight is calculated and then the error rate of the calculated weight is calculated, and the optimum result is tried to be found. The software flow diagram that allows the weights of a model to be updated is as in Figure 2.4. To measure the success of the ML model created using the training data, a validation dataset consisting of labeled data should be created. The BP algorithm uses the labeled validation dataset to calculate the error rate. After each iteration, it tests the model with the validation dataset and updates each weight according to the error rate.

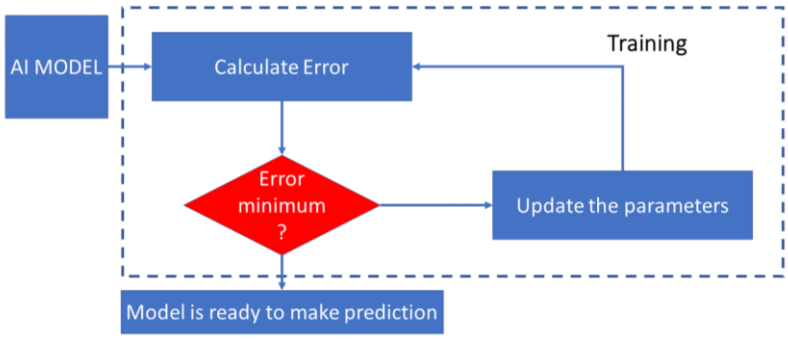

Figure 2.4 Back propagation flow diagram.

BP algorithms search for the minimum error in the weight calculation using the technique called delta rule and gradient decrease. As shown in Figure 2.5, the initial weight value is updated at the end of each iteration, trying to hold on to the minimum

error value. Some of the optimizer's algorithms trying to find the minimum value using these methods adaptive moment estimation (Adam) [23], adaptive learning rate method (AdaDelta) [24], stochastic gradient descent (SGD) [25], and root mean square error probability (RMSprop) [18] is called.

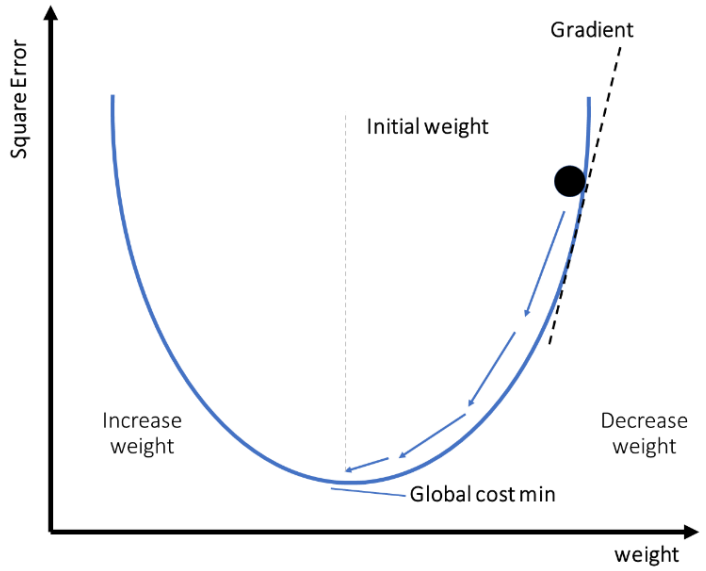

Figure 2.5 Optimizers minimum error rate search graph.

The schematic representation shown in Figure 2.6 is a simple representation of an ML model with 1 hidden layer, 3 inputs, and 2 outputs. The variables a, b, w, and y, respectively, are input, bias, weight, and output. The operation of the BP algorithm is described below in steps. Calculations can be analyzed in 3 main steps as forward propagation, back propagation, and updating the weights [26].

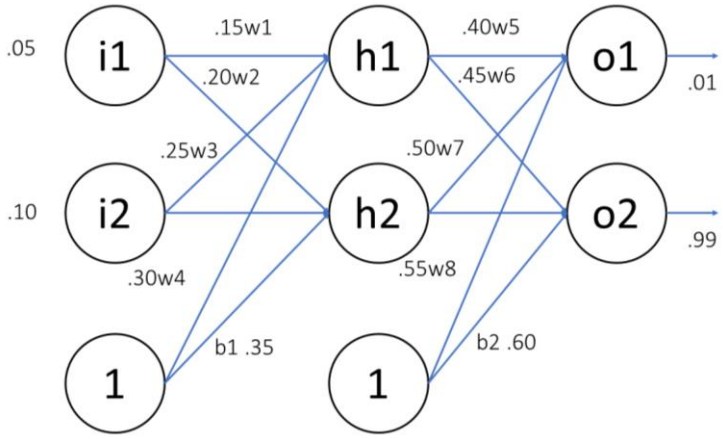

Figure 2.6 Basic neural network schematic representation.

The first step is to create the weights of an ML model. When an ML model is first started, the weights are randomly assigned, and the calculation process is started.

2.3.1 Forward Propagation

In the second step, the total value of the weights of the relevant perceptron is calculated using the Activation function. Hidden node value is calculated because of multiplying the weight of each input value connected to the node and summing the bias value [27].

$$h_j = f\left(\sum_i a_i w_{ij} + b_j\right) \tag{2.5}$$

Where:

a_i: the input training vector

w_{ij} : the weight connected between input node *i* and hidden node *j*

b_j : the bias of hidden node *j*

hj : the output of hidden node *j*

In the next step, the activation layer function is applied to the sum of the hidden layer weights to calculate the output value.

$$Y_k = f\left(\sum_j h_j w_{jk} + b_k\right) \tag{2.6}$$

Where:

w_{jk} : weight that between hidden node *k* and input node *j*

b_k: bias value of hidden node *k*

Y_k: output of hidden node *k*

In the next step, Back-propagation error will be calculated. The calculated error value is used to calculate the correction coefficient to be calculated in the next step.

$$\delta_k = (T_k - Y_k).f'\left(\sum_j h_j w_{jk}\right) \quad (2.7)$$

Where:

δ_k: error of node k

T_k: Expected output at node k

f': derivative of activation function

In the next step, the correction coefficient between the hidden node and the output is calculated using the error value. The correction coefficient is used to update the weights for the next stage, and the weight values after each update reach the ideal weight values that work with less errors.

$$\Delta w_{jk}(n) = \eta\delta_k h_j + \alpha\Delta w_{jk}(n-1) \quad (2.8)$$

η: Learning rate

α: Momentum coefficient

Δw_{jk}: weight adjustment constant for input node j and hidden node k

2.3.2 Back Propagation

The purpose of the BP algorithm is to update each weight value in the network using the error rate values. In this way, the distance of the output at the end of the network to the targeted output is reduced. For this convergence, it is aimed to operate the function with a minimum error rate by using the gradient descent algorithm [28]. In this step, BP algorithm formulas will be examined.

$$\delta_j = \sum_k \delta_k w_{jk}.f'\left(\sum_l a_l w_{lj}\right) \quad (2.9)$$

Where:

δ_j: error value of node j and a_l

w_{lj} : weight between hidden node j and input node l

Error term sum is calculated for each hidden node using the error rate calculated in the previous step.

$$\delta_l = \sum_j \delta_j w_{lj} \cdot f'\left(\sum_i a_i w_{il}\right) \tag{2.10}$$

δ_l : error of node l

In the next step, the weight correlation term between the input and the hidden node is calculated using the node l error rate value.

$$\Delta w_{ij}(n) = \eta \delta_j a_i + \alpha \Delta w_{ij}(n-1) \tag{2.11}$$

Δw_{ij} : weight correction term of input node i and hidden node j

2.3.3 Update Weights

This step is about updating the weights. The update process is done by including the correction term value calculated in the previous stage in the calculations. In the process of updating the weights, the function tries to find the optimum value by multiplying the learning rate with the correction term. The size of the learning rate value causes the function to move to more spaced points. The small learning rate value causes the function to move at more frequent points. Therefore, the learning rate has a significant effect on the training of the model.

$$\begin{aligned} w_{jk}(n+1) &= w_{jk}(n) + \Delta w_{jk}(n) \\ w_{lj}(n+1) &= w_{lj}(n) + \Delta w_{lj}(n) \end{aligned} \tag{2.12}$$

2.4 Deep Learning

Deep learning is a sub-field of machine learning that consists of successive layers and more useful representations can be obtained as each layer is processed [29]. The word deep refers to the multiplicity of layers that follow each other. The structure formed by the layers is called a model or network. The number of layers in the model determines the depth of the model. While classical machine learning algorithms usually consist of one or two layers, modern deep learning models consist of dozens of successive layers. This layered structure is called neural networks and consists of successive layers. These systems can be thought of as a multistage information

distillation filter. It consists of a basic deep learning algorithm, CNN (filter), pooling, activation layer, dropout, Flatten, fully Connected sublayers.

Deep learning, which is a sub-field of machine learning, has been used frequently after 2010. Problems that are difficult to solve with machine learning can be solved with deep learning [30-31]. It performs at human level in many applications such as image classification [32], voice recognition [33], automatic translation [34], text to speech [34]. There is still ongoing research on what can be done. The reasons that increase the use of deep learning are that it offers high performance for many problems and that the feature extraction stage, which is a critical step in machine learning application, is fully automated with deep learning [35]. The advantages of deep learning networks are that they do not use feature extraction and their high-performance values. The disadvantages of DL are the need for high performance hardware and the need for a large amount of data. With the developing technology, the increase in the parallel processing capability of GPU-based graphics cards with frameworks such as CUDA has made DL models available. In addition, the data accumulation of many applications on the internet and the labeling of this accumulated data by users has paved the way for new AI applications.

2.4.1. Deep Learning Networks

Deep learning networks are CNN-based networks consisting of more than one hidden layer [36]. There are different DL networks created in the literature. These networks have strengths and weaknesses according to the speed of training and prediction [37]. In addition, some networks can be better trained with complex images, while some networks can be better trained with simple images. Therefore, these networks differ from each other in terms of speed and performance. In DL networks, after each Convolutional layer, the weight value is formed according to the number of inputs. Each weight value takes a place on the random-access memory (RAM). Therefore, the larger the network, the greater the capacity of the computer system must be. In the DL image classification studies in the literature, it is generally tried to reach the most successful result by trying all the networks. A few of the known of these mentioned networks are listed below.

1. LeNet Network - 1998

The CNN-based network known as Lenet-5 was developed by Yann Le Cunn in 1998 [38]. Le Cun found this network while looking for an answer to the question of how handwriting images could be defined with computer-based software. The architecture of the LeNet network is an input dataset consisting of 32x32 pixel images. With the first process 5x5 filter and convolution layer, the image size is reduced to 28x28 pixels. At this stage, while some features of the image are eliminated, some features are highlighted. In the next stage, the data size was reduced to 14x14 pixels by applying pooling. In the last stage, it is connected to 10 outputs using the fully connected layer. The 10 outputs in this study represent numbers 0 to 9. The LeNet network developed for the Handwriting classification application has transformed a dataset of handwritten letters into a model with 10 outputs, that is, 10 classes.

2. AlexNet Network - 2012

AlexNet was developed by Alex Krichevsky in 2012 as a CNN-based DL network [36]. AlexNet and LeNet are similar in structure. However, AlexNet has both more filters and more convolutional layers. It is considered deeper because of the septum. The main difference that separates AlexNet from LeNet is that it is deeper and uses RELU layers [39].

3. VGGNET Network - 2014

The VGGNET network, published in 2014, consists of 11 to 19 layers and is a deeper network than AlexNet, LeNet. VGG networks are named VGG11 – VGG19 according to the number of layers [37]. One of the most effective networks among VGG networks is VGG16. Oxford Visual geometry group (VGG) developed VGG16 network for training ImageNet dataset and object classification [37]. VGG16 has a structure that uses 3x3 filter structure and uses more layers than AlexNet. The ML model trained using VGG16 network and ImageNet dataset classified 1000 different objects, namely classes, with a success rate of 92.7%.

4. SqueezeNet Network - 2017

SqueezeNet is a CNN network consisting of 18 layers. Tested on ImageNet [40] image database, which consists of more than 1 million and 1000 different object images. SqueezeNet is a network built on top of the AlexNet network and is 50 times faster than AlexNet [40]. The feature that distinguishes Squeezenet from other networks is that it has bypass and fire functions. Fire module's expand layer has a predefined number of filters partitioned between 1x1 and 3x3, and here we turn the knob on these filters from mostly 1x1 to mostly 3x3 [41].

2.4.2 Deep Learning Tools

Thanks to the frameworks specialized for DL training and testing, GPU-based hardware can be used, and software can be developed for this hardware. For DL training, TensorFlow [42] developed by GOOGLE, which is open source, and the use of python programming language together with Keras libraries developed by Chollet, F. are frequently used methods for DL training [43].

The correct software usage from GPU hardware to the user are CUDA, TensorFlow, and Keras respectively. CUDA is a framework between hardware and software that parallelizes operations. It is a framework that contains image reading and ARRAY operations, parallelization libraries in TensorFlow. It is the framework where operations such as the installation of the image processing DL model such as Keras TensorFlow are made [43]. CUDA, TensorFlow, and Keras are similar libraries in terms of functions. DL applications can be made using only CUDA or CUDA + TensorFlow. TensorFlow and Keras libraries are designed to use more user-friendly functions only to facilitate the user's work [43]. The aim is to enable the user to develop DL applications faster by writing fewer pieces of code.

2.4.3 Deep Learning Training with Synthetic Images

Many studies are using DL models. Image classification [44-46], Natural language processing [47-48], Object detection [49], signal processing [33] etc. A standard DL training consists of data collection, data preprocessing, training, and testing. Many images are used in DL-based AI training. The number of images in the studies varies between 10K and 1000K [40][50]. To create the image dataset required for image classification, it is necessary to photograph objects or find object photos from the

internet [51]. This is labor-intensive work. Facilitating dataset collection or creation is one of the areas of study in the literature [52]. In some studies, in the literature, photos from the scenes of the games are used for AI education and it is tried to overcome the data constraint. These data are classified as synthetic data. The focus of these studies is to ensure that AI trained using the unreal image dataset can work successfully on real data [53]. Some of these studies are listed below.

1. Semantic Segmentation for Autonomous Car

Vision-based semantic segmentation plays a key role for the driving control algorithms of autonomous vehicles in urban motion scenarios [54]. The authors developed a city using the game development engine and created a dataset called SYNTHIA with 213400 photos they collected from this city. They tried to create a base for AI-based systems that would perform tasks such as pedestrian recognition of vehicles. They try to increase the performance of AI prediction by using it together with synthetically produced real data [54]. The dataset created for training was mixed with real images, providing a success rate of up to 94%.

2. Smoke Detection

Autonomous systems can be used to monitor forest areas against fire using remote monitoring technology. The researchers created their first dataset by recording real smoke images and combining them with real forest images in a real green box studio. They created the second dataset by taking synthetic smoke images from simulation programs and combining them with real forest photos. They created the third dataset by collecting real forest smoke images. They created the training dataset from these 3 datasets. The test results of the trained artificial intelligence were successful with an accuracy rate of 99.71% [55].

3. Joint Detection

Researchers used 3D model files to extract similarities between 3D models and 2D images. 3D model files are synthetic models created using computer aided design (CAD) based software. In this study, the dataset was created from 3D model files and 2D image files. The AI model trained using the training dataset, which consists of combining synthetic and real data, has been successful with 80% accuracy [56].

4. Dental Occlusion Classification

Researchers tried to classify dental occlusion diseases by using 3D scanner models in STL format collected from patient dental records. The dataset consisting of 3D models was converted into 2D histogram images and AI training was carried out. The trained model was successful with an accuracy rate of 78.95% [57].

5. Anomaly Detection in Crowd Scenes

Anomaly detection in crowded cities is a hot topic for security applications [58]. The researchers developed a system to generate synthetic data due to the scarcity of data to be used in training the AI model. They trained a CNN-based model with 2149 videos they collected from this game engine-based system. The trained model classified events such as fight, scatter and fire with an accuracy of 92% [59].

2.4.4 Convolutional Neural Networks (CNNs) and Sub Process

CNN is used as a sub-function of deep learning architectures for data filtering [60-61]. CNN is generally used for image or video [60]. A standard image file consists of a combination of data called pixels and containing an 8-bit color code. 3 pixels defined as red, green, blue (RGB) can overlap and form any color on the color scale. This is the digital version of an artist's ability to mix colors to create different colors. Images are classified as RGB and are colored images created using 3 primary colors. Each primary color is represented by an 8-bit data from 0 to 255. Therefore, a color image consisting of 1 pixel is a 24-bit, that is, 3-byte data. In the image shown in Figure 2.7, there is a schematic representation of a 4x4 pixel RGB image. A video file consists of a sequence of image files, one after the other. The image formed by displaying image files at 24 frames per second is called video.

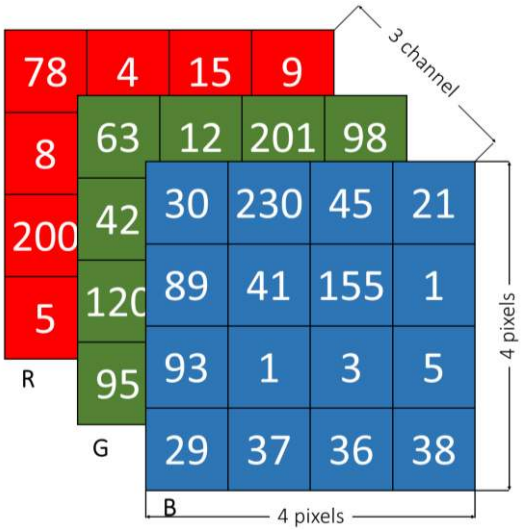

Figure 2.7 Basic RGB image matrix representation.

CNN architectures try to classify images by extracting features from images [60]. Completing the process of learning the remaining features, the neural network tries to detect similar features on a new image [62]. Detected features form a pattern. The weight matrix, which is a combination of patterns, is learned knowledge.

CNN networks transform the pixel values of image files by processing them with the mathematical operator convolution. CNN networks consist of many convolutional operations, this structure is called network. The number of filters used in the Convolution mathematical operator is one of the parameters that affect the success of the feature extractor of CNN networks.

1. Filters (Convolutional Kernels)

All values of the filter to be applied to the two-dimensional information are multiplied by the elements of the input matrix and the sum of all values is recorded as the relevant element of the output matrix. This is also called a cross-correlation relationship [63]. In this process, the filter feature detector output matrix is a feature map. As a result of the convolution process, some features of the input data are revealed while some data are eliminated. Which data to extract depends on the properties of the filter.

In the schematic representation shown in Figure 2.8, the result matrix is obtained by subjecting the image data called input to the convolution process with the filter. The filter selected in 3x3 pixels transforms the 4x4 pixel data of the image into 2x2 pixel

data. The filter is applied to the whole image by shifting it one by one by the stripe value on the image. The stripe value is a parameter that affects feature extraction [64]. This operation changes the size of the input data. For this size change not to be a problem when moving to the next convolution layer, the new matrix can be subjected to padding.

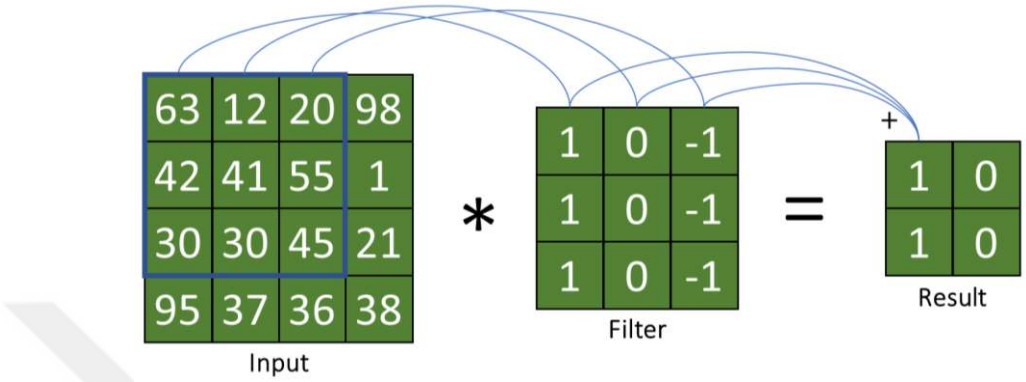

Figure 2.8 Convolution math operators representation.

2. Padding Process

In CNN-based networks, the size of the data whose dimensions change after each convolution layer can be determined again. This process is applied by adding new elements to the matrix. If the input matrix is 4x4 and the output matrix is 6x6, the dimensions of the output matrix are desired to be equal to the dimensions of the input matrix. The matrix dimensions are enlarged by adding the value 0 instead of the elements that are not as shown in Figure 2.9.

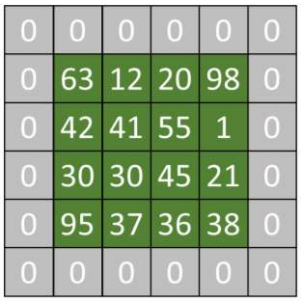

Figure 2.9 4x4 image matrix padding model.

3. Max Pooling Process

The pooling layer is a layer used after the convolutional operator. This layer saves by selecting the largest value of the values found in the determined pooling measures [64]. In the image shown in Figure 2.10, 2x2 max pool operation is applied on the

4x4 input matrix. The 2x2 matrix, which is the result of the application, is shown in the figure. The largest value of each 2x2 piece of the 4x4 input matrix is determined and written into the relevant cell of the new matrix.

Figure 2.10 Max pooling process and result.

4. Activation Function Process

The activation function is a non-linear function located between the input neuron and the next neuron. It has an activation function with different features. Rectified Linear Units (RELU) pass positive values, eliminating negative values. As shown in Figure 2.11 it accepts negative values as 0 and does not change positive values.

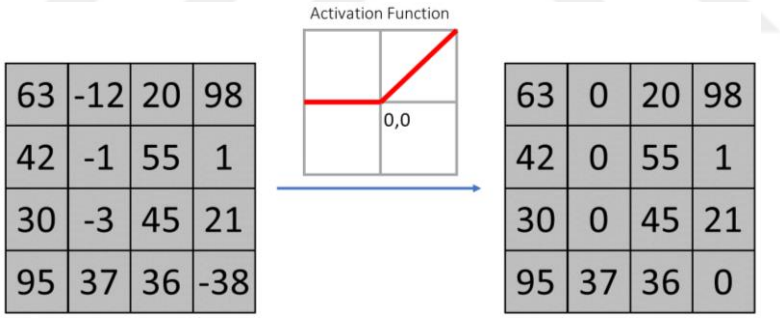

Figure 2.11 Activation layer process and result representation.

5. Dropout Process

The dropout process is the process of deleting the perceptron in the hidden layers and the weights to which this perceptron is attached [65]. In the display shown in Figure 2.12, some perceptron among the hidden layers is inactive. While creating the artificial intelligence model, the dropout rate can be adjusted after each hidden layer [65]. Training takes place according to the determined dropout values. According to the determined dropout rates, the perceptron is selected from the relevant layer and the rest is deleted. Which perceptron will be selected is decided randomly?

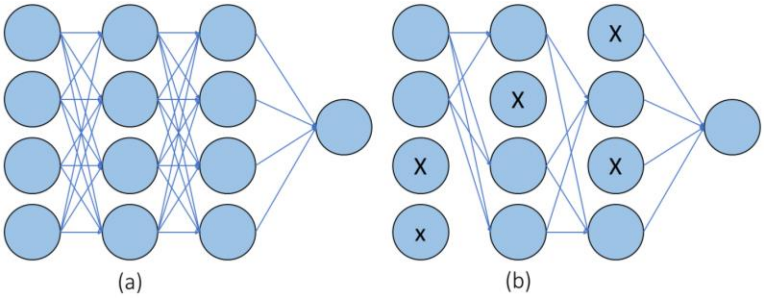

Figure 2.12 Dropout process and result representation.

The high or low amount of perceptron in AI models is a parameter that affects the success of the AI model [66]. But there is no general rule between the number of perceptron and success [66]. In some cases, the high number of perceptron causes the network to overfitting [66]. This results in the network guessing incorrectly. In these cases, the method of deleting some perceptron is used to prevent the network from memorizing. The name of this method is dropout [66].

6. Flatten Process

Flatten layer is a layer that transforms the image matrix consisting of rows and columns into a column. The purpose is that we want to later input this into an artificial neural network for further processing.

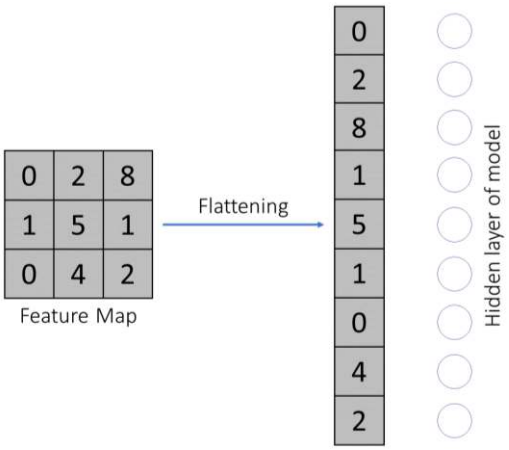

Figure 2.13 Flattening process and result representation.

As shown in Figure 2.13, a 3x3 matrix consisting of rows and columns was converted into a 1x9 column matrix. This transformation is done to ensure compatibility with the hidden layer consisting of neurons [67]. In the schematic representation shown in Figure 2.13, the input data consists of rows and columns, while the hidden layer consists of only columns. To ensure that the row and column

numbers are the same, the dimension between the input data and the hidden layer is harmonized by applying the flatten operation.

7. Overfitting Detection

Overfitting is the overfitting of the AI model [66]. Memorizing the learned data and if it memorizes every prediction data it encounters. In the first evaluation, although the training of the model is measured as successful, it produces unsuccessful results because it makes every data look like the data it has learned. The best way to understand that a model is overfitting is to interpret the training and validation accuracy values graph.

The image shown in Figure 2.14 shows the accuracy and validation graph of an AI network trained with 100 epoch iterations. The accuracy value indicates the success of the network in each epoch. The line shown as the training data is the success value measured during the training. The line indicated by the validation label shows the successful result of testing the AI with new data after completing its training. It is seen that as the number of epochs increases, the training accuracy value increases, but the test accuracy value does not change much. This graph indicates overfitting. At the end of the training, while the training accuracy value of the AI model was over 90%, the test accuracy value remained at 70%. This situation shows that the AI model memorizes and cannot do the learning process correctly.

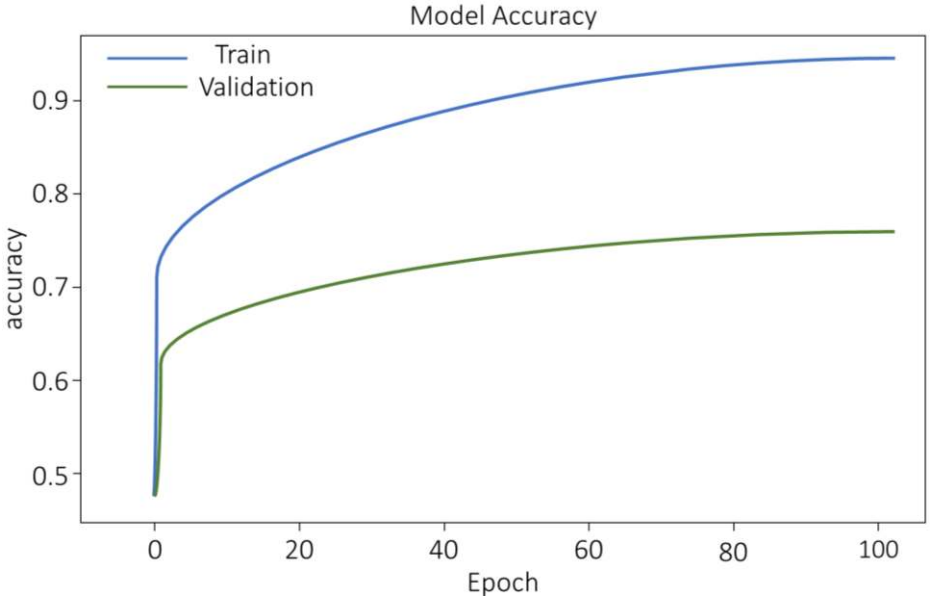

Figure 2.14 Neural network overfitting graph.

2.4.5. Central Processing Unit and Graphical Processing Unit (CPU and GPU)

Machine learning studies have been examined since the development of processors. Prediction algorithms, curve fitting algorithms, and similar applications could be done using CPU-based computers. However, CPUs are processors that can perform only one operation at a time [68]. Processors that can perform a single operation at a time and perform sequential operations take a long time to perform a transaction-intensive job. For these reasons, it takes a long time for complex machine learning algorithms to be developed and run on the CPU. Systems have been developed that allow CPUs to be connected to each other in parallel and to collect different workpieces of a job at the same time [68]. However, these systems are expensive in terms of volume and cost. With the development of GPUs, hardware capable of parallel processing has been produced. GPUs are the hardware that can perform many mathematical operations at the same time. With the development of GPUs, the hardware constraint, which is a problem for the development of parallel processing software, has disappeared [69]. Machine learning libraries that perform parallel processing with GPUs have begun to be produced [69]. Deep learning method emerged with the development of GPU and parallel processing libraries. GPU-based systems run the deep learning algorithm faster. The graph shown in Figure 2.15 shows a comparison of GPU and CPU speeds for some DL networks. The graph shows that Intel CPUs need much longer training times than GPUs.

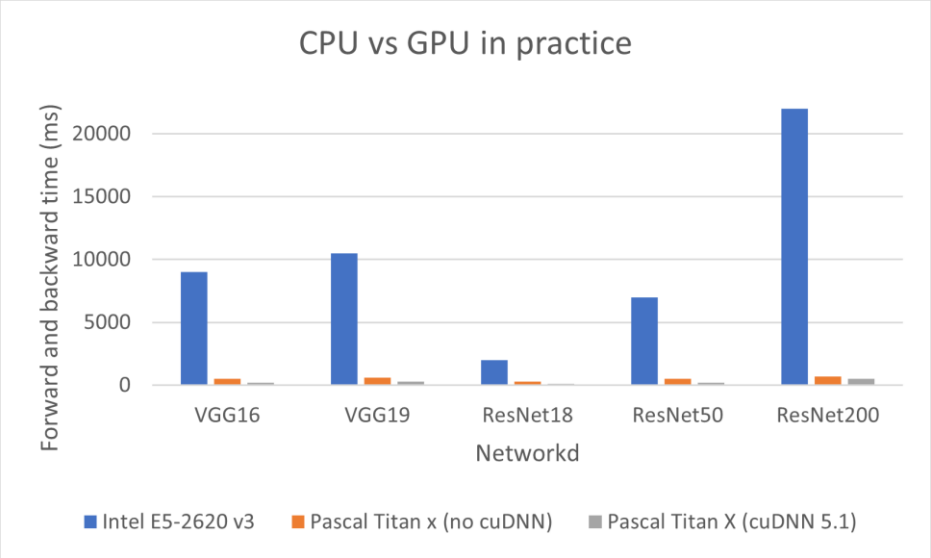

Figure 2.15 CPU and GPU hardware speed test comparation graph.

Today, the two major GPU manufacturers are NVIDIA and AMD. With the CUDA parallel processing library developed by the NVDIA company, software that performs parallel processing on GPUs can be developed [70]. The DL libraries used are widely developed on CUDA.

2.5 Data Preparation Process for Image Classification with Deep Learning

Data collection is the process of collecting data that will be used in training the AI model to be trained for classification. The number of data to be used for the DL networks training process is a parameter that directly affects the success of AI training. Therefore, the more the number and variety of data collected for the images to be classified, the more successful models can be trained. The images to be collected for data collection can be obtained using ready-made data sets [71]. Some of these datasets are CIFAR10[50], IMAGENET. However, the type of image to be classified may not be found in the ready datasets and it is a common situation that the desired type of data is not found. In this case, the researcher should create the image dataset. Photographs of the object can be taken using the camera to create the dataset. Searching and downloading can be done over the internet using search engines. For DL networks training we need to do this for about 10000-20000 images [72]. Considering these large numbers of data sets, this process requires time and effort.

In the next step, the collected data needs to be prepared for training. This preparation process is the labeling of the data one by one. The labeling process may involve different stages depending on the resolution quality of the images. Data collection and data preparation are labor-intensive and demanding process. The meticulous work during the preparation of the training dataset directly affects the accuracy values of the model, namely its success.

CHAPTER III

METHOD of SYNTHETIC IMAGE GENERATION and TRAINING

3.1 Method

Machine learning (ML) methods are among the most frequently used methods to classify UAVs with sensor fusion data. Image classification methods with artificial intelligence (AI) are used in UAV classification studies [12]. With the increase in the number of fixed-wing (FW) and rotary-wing (RW) unmanned aerial vehicles (UAV) and the ease of accessibility, serious security gaps have occurred. In studies on FW-UAV and RW-UAV in the literature, it has been tried to detect UAV by using radar-based [12], sound-based [73], optical-based [74] sensors. In addition, there are UAV detection studies using sensor fusion, that is, using several different sensors together [12]. Artificial intelligence (AI) and image classification methods are used in UAV classification studies. Autonomous systems capable of AI-based image classification can be developed. A series of processes are required for the AI-based detection capabilities of autonomous systems. Image classification methods with AI consist of a series of processes such as data collection, data preparation and labeling, training, and testing [13]. The data type used to training ML models in image classification problems is an image. Therefore, an ML model with image classification capability is expected to be trained and tested with an image dataset. The stages of the image classification method, which is the subject of this study, are explained below.

Image classification ML models used for the solution of image classification problems need a dataset consisting of the images of the objects to be classified. Several different methods can be used to create a dataset. One of them is to create using ready-made datasets such as IMAGENET [40], CIFAR10 [50]. However, ready-made datasets consist of limited data [40][50]. Therefore, these datasets are useless in studies for detecting objects that are not in the ready datasets. In cases

where ready datasets cannot be used, datasets can be created by taking pictures of the object with the camera. The diversity of the created dataset plays an important role in the success of the ML model. For this reason, when manually created datasets, the object should be photographed from different angles and from different distances. In manually created datasets, diversity is tried to be created by changing parameters such as different angles and different distances [75]. The dataset, which is created manually, needs to be prepared in the next stage of data preparation. The preparation process is the systematic stacking of photographs by examining them one by one. Collecting data and organizing these collected data according to the ML model are called data collection and preparation, which is a preliminary process for education. Data collection and data preparation processes are labor-intensive and create a significant iterative workload for each new class when training the machine learning model [75]. Recently, the success of image classification research has increased with the development of deep learning ML networks. Classical ML networks involve feature extraction and feature selection. Feature extraction and feature selection layers are manual process in classic ML. In deep learning networks, feature selection and feature extraction operations are automated. [35]. Automation of feature extraction and feature selection layers by DL can be considered as an advantage of DL networks [76]; however, the disadvantage of this advantage is that DL networks needs many data [7]. In deep learning and machine learning-based image classification applications, the convolutional neural network (CNN) is the most widely used method. CNN with deep learning-based networks is a method that is constantly evolving and is becoming more popular for image data classification studies [60].

In this study, RW-UAV detection has been studied utilizing camera-based and DL-based networks. Since birds are the most natural objects that can be confused with RW-UAV in the air, bird and RW-UAV classification have been made. In the experiments, we focused on progressing the accuracy of the results obtained with the deep learning-based ML models trained with the synthetic image dataset generated entirely using the game engine and tested them with the originally generated real image dataset. A virtual studio has been created using the Unity game engine to create a synthetic dataset. 69120 synthetic images have been created using three-dimensional (3D) models of the bird and RW-UAV. In total, 34560 synthetic RW-

UAV images and 34560 synthetic bird images have been generated. The produced images have been trained in different combinations by changing the CNN-based AlexNet, Visual Geometry Group Network (VGGNet), and SqueezeNet, LeNet networks and the different optimizer, dropout, batch size, and learning rate parameters of the models. DL models with a total of 288 different parameters have been trained with these synthetic images. Models have been individually tested with 3385 real images and 5000 synthetic images.

3.2 Green Box Studio Design and Synthetic Image Rendering

To create synthetic images, a virtual studio with a green background has been prepared in the UNITY game engine environment. Green box is one of the techniques used in the studio cinema industry. It is a studio consisting of a green tone color, whose hue and code are known and rarely found in nature. The basic principle is to add the desired backgrounds by removing the background from the photograph of the object taken in a studio with a green background. The photographs shown in Figure 3.1 (a) and (c) are photographs of the object with a green background. The photograph shown in (b) and (d) is the photograph in which the green tone has been removed from the photograph and a new background has been added. Since the green tone is a known color code, it becomes easier to extract the green color from the pixels of the image software.

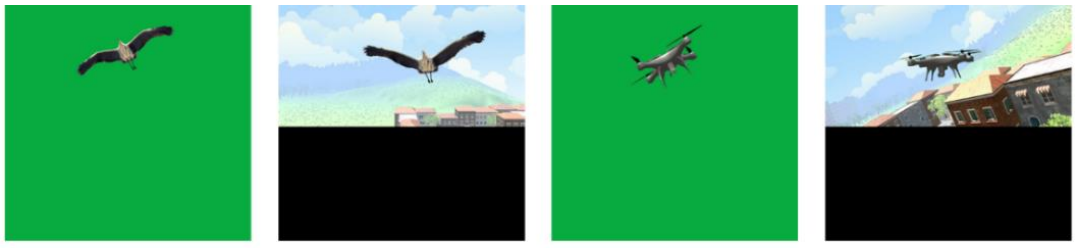

Figure 3.1 Green background and city background synthetic images of FW-UAV and bird.

The synthetic image dataset required for training the DL-based AI model has been created entirely using green box studio. 3D RW-UAV and 3D bird models have been photographed from different angles in the green box studio and converted into 2D images. The 3D models have been rotated around the X Y Z axes with a determined angle, photographed from every angle and the dataset has been created. This process

has been applied separately for all RW-UAV and Bird models. As a result of the process, a total of 34560 RW-UAV and 34560 bird images have been obtained. The picture shown in Figure 3.2 is the visual of the virtual studio prepared in the UNITY game engine environment. Figure 3.2 is (a) camera (b) RW-UAV 3D model, (c) light sources, (d) green screen, (e) sample camera view. As shown in Figure 3.2 (e), a photograph has been taken and recorded after each rotation of the 3D model, which rotates gradually from the front of the camera.

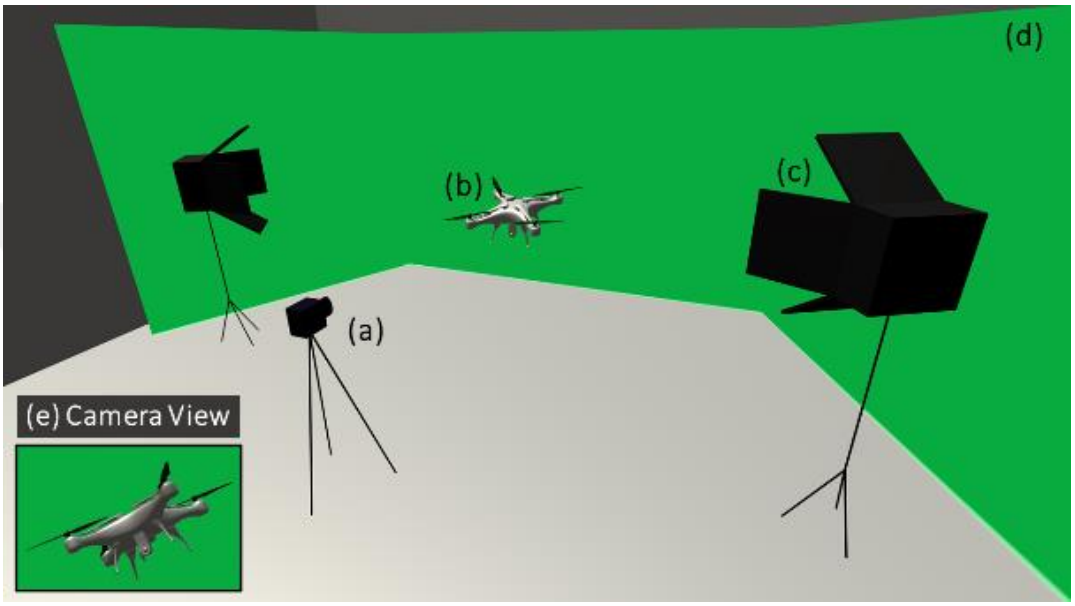

Figure 3.2 Green box studio designed in the unity game engine environment: (a) Virtual camera used to take a picture. (b) RW-UAV 3D model. (c) Light sources to generate realistic pictures. (d) Green-colored background, wall view. (e) Example camera view.

Since the models have a symmetrical shape, the same shape is obtained when rotated 90 degrees on the X-axis, 30 degrees on the y-axis, and more than 30 degrees on the z-axis. This causes the formation of a symmetrical picture of the same picture. To avoid this repetition, the models have been rotated 90, 30, 30 degrees on the X Y Z axes, respectively. In this way, the dataset consists of unique images.

```
Set angles of UAV model to 0,0,0, respectively, as x, y, and z axis;
for x_a ← 0 to 90 do
    Rotate UAV model on X axis by 1 degree;
    for y_a ← 0 to 15 do
        Rotate UAV model on Y axis by 2 degree;
        for z_a ← 0 to 15 do
            Rotate UAV model on Z axis by 2 degree;
            Take a picture from the camera that views the UAV model;
        end
        Set Z axis angle of UAV model to 0 degrees;
    end
    Set Y axis angle of UAV model to 0 degrees;
end
Set X axis angle of UAV model to 0 degrees;
```

Figure 3.3 Algorithm 1: RW-UAV model rotation and synthetic image generator algorithm in the unity game engine.

In Figure 3.3, algorithm 1, the scripting algorithm that performs the rotation and photographing of 3D models around the X, Y, Z axes in the green box virtual studio is shown. Basically, the algorithm, which consists of 3 nested loops, photographs the model by rotating it by 2,1.1 degrees around the X, Y, and Z-axis, respectively. At the end of each cycle, it starts again by making the angle of the relevant axis 0 degrees. Thanks to this process, it is possible to photograph different angles of the model.

All data images must be unique as the generated dataset is used in training, validation, and testing processes. Having the same data in the training validation and test dataset leads to an incorrect evaluation of the training of the AI model. The success of the AI model may turn out to be more successful than it should be. To avoid this error, each element of the generated dataset was compared with other elements. The comparison was done using the message-digest algorithm (MD5). Each image was encrypted with the MD5 algorithm and images with the same password were deleted. This method is a method used in the literature [77].

The AI model trained with RW-UAV and bird synthetic image data has been tested with real RW-UAV and bird photos and synthetic RW-UAV and bird photos, and the prediction success have been measured. Two different cities have been designed using the UNITY game engine to test with synthetic data. Cities have been designed

considering the environments that can be encountered in daily life. Objects such as buildings, traffic lights, streetlamps, tables, chairs, trees, and lakes required for city design were purchased from the UNITY asset store [78] and combined. These cities have been used as a simulation where we could measure the prediction success of AI. The view of the two cities from different angles is shown in Figure 3.4.

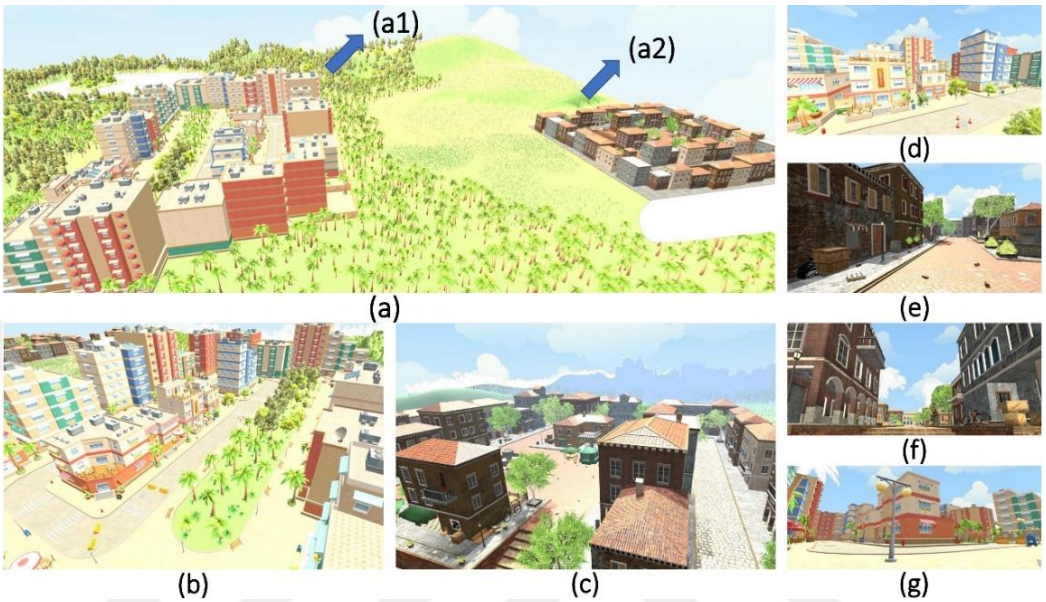

Figure 3.4 Modern and historical city views designed in the unity game engine virtual environments: (a) Game engine terrain of all design. (a1) Modern city design. (a2) Old city design. (b–g) City views from different angles.

One of the two cities has been created with modern design lines and the other with old design lines. The modern city is equipped with objects that we encounter in our daily lives, such as high-rise buildings, air conditioners on roofs, restaurant signs, trees. Images from the modern city area are shown in Figure 3.5.

Figure 3.5 Sample modern city views.

The old city has been equipped with historical buildings, trees, and objects such as roof chimneys. The purpose of designing cities is to measure the success rate of RW-UAV and bird classification within the city image of AI. Images from the old city area are shown in Figure 3.6.

Figure 3.6 Sample old city views.

3.3 Data Organizing for Training

The synthetically produced image dataset has been divided into three groups as training, validation, and testing. DL models measure how successful the validation image data set can be predicted with the weight values created according to the input data during the optimization process. It updates the weight files according to the

measurement result. This process is repeated as many as the number of iterations of the DL model. As a result of the iteration process, the network is trained in its most successful state. For this reason, the training and validation dataset are used during AI training. After the training process is completed, how successfully the network is trained is measured with the test dataset. The validation dataset is used in the training at the end of each iteration, while the test dataset is used after the completion of the training. For a correct evaluation, the data in the training, validation, and test version should be different from each other. To enable controlled experimentation, our synthetic dataset has been organized as shown in Figure 3.7. The visual shown in Figure 3.7 has been created to show the preparation of the RW-UAV dataset. The same process has been done for bird synthetic images.

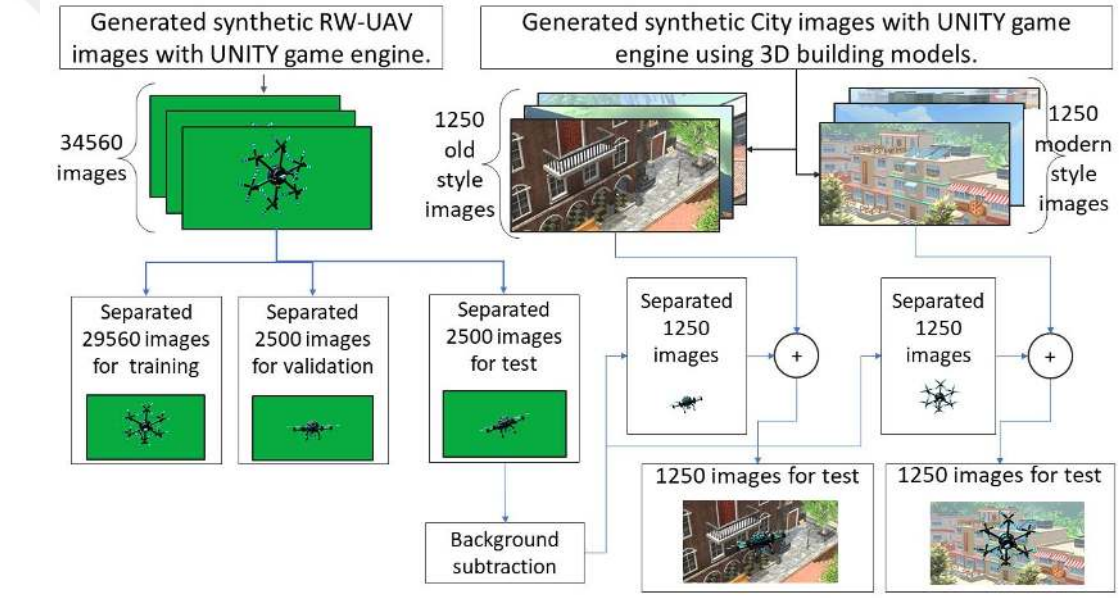

Figure 3.7 The schematic representation related to the creation of a dataset of the RW-UAV class.

The RW-UAV dataset, consisting of 34560 synthetic image data, has been divided into 29560, 2500, and 2500 for the training, validation, and test dataset, respectively. The background color of 2500 images reserved for the test has been removed and modern and old-style city images have been added. The created dataset was prepared considering the compatibility with AI models and the compatibility of the datasets within themselves. Therefore, all images created are 224x224 pixel RGB images.

RW-UAV and bird synthetic images with city backgrounds have been used for the test dataset. The predictive success of the trained AI model on synthetic data has been measured using these data. The realism of the test simulation scenario was tried to be increased by using environments created with the old city and modern city designs.

3.4 Real-Test Dataset Collection and Preparation

The research purpose of this study is to create a model that can classify real images with the AI model trained using synthetic images. A dataset of real RW-UAV and bird photos is needed to see the results of the research. The model, which was trained with fully synthetic images, was tested with real images as well as in the simulation environment created with synthetic images. Three steps were applied to collect real RW-UAV and bird images, identify similarities of collected images, and delete and edit similar images.

In the first stage, a script has been developed that search for real RW-UAV and bird images using Internet search engines and downloads results from images. In addition, a script has been developed that downloads RW-UAV and bird videos from internet video sharing sites and saves the images of these videos at 5-second intervals. Using these two scripts, 8560 RW-UAV and 12569 bird images have been downloaded from the internet.

In the second stage, many duplicate and similar photographs have been detected in real object images downloaded from the Internet. Similar photos and duplicate photos lead to inaccurate assessment of test results. If the photo that the AI model classified correctly during the test phase has duplicates, this will cause the model's success to be overestimated. On the contrary, it causes the prediction success score of the model to be lower than it is. To avoid these errors, duplicate and similar photos should be deleted from the test dataset. A similarity detection script has been developed for deleting similar and identical images. Developed using the template match modes coefficients normed (TM_CCOEFF_NORMED) [86] function of the open-source computer vision library (OpenCV) library, the script looked for similarity in the real image dataset. The similarity extraction method is a method that has been used in the literature, and we extracted all images with a similarity rate of

more than 80% [97]. The formula of the TM_CCOEFF_NORMED function used for similarity detection is given in formula x.

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y').I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2.\sum_{x',y'}I'(x+x',y+y')^2}} \quad (3.1)$$

where T, R, I, and (x, y) values are the template, resulting image, source image, and position of pixel values, respectively. Using these methods, a dissimilar dataset consisting of 1859 birds and 1526 RW-UAV real images was created. The number of elements of the dataset, which consists of real RW-UAV and bird images collected from the Internet, has been greatly reduced. However, a real image test dataset consisting of 3385 images in total has been obtained. Some examples of real datasets are presented in Figure 3.8

Figure 3.8 Some examples of real bird and RW-UAV images. (a–c) Bird images; (d-f) RW-UAV images.

In the third stage, the collected and extracted images have been modified according to the appropriate aspect ratio for the testing process. The size of the real image's dataset collected were between 227 × 227 pixels and 1280 × 1280 pixels. Image sizes were converted and scaled to 224 × 224 pixels for the trained AI to make classification.

3.5 Combinations of Network's Parameters

Synthetic images dataset has been used to training deep learning-based classification networks specifically AlexNet, SqueezeNet, LeNet and VGG16—in the setups for experiments. All combinations of AlexNet, SqueezeNet, LeNet, VGG16 DL networks, and adaptive moment estimation (Adam) [23], adaptive learning rate method (AdaDelta) [24], stochastic gradient descent (SGD) [25], and root mean square error probability (RMSprop) [18] optimizers have been applied for these

experiments. Networks and optimizers have been trained with combinations of the different hyperparameters listed in Table 2.1.

Table 2.1 The combinations values of hyperparameters.

Hyperparameters	**Values**
Optimizers	Adam / AdaDelta / RMSprop / SGD
Batch Size	32 / 128 / 256
Learning Rate	0.1 / 0.01 / 0.001
Dropout	0.5 / 0.8

AI training has been completed using synthetic image data. It has been tested using synthetic and real data during the testing process. 7 different experiments have been trained with all combinations of 4 different DL networks, 4 optimizers, 3 batch sizes, 3 learning rates, 2 dropout values. Therefore, 288 network training have been performed for each experiment. The networks that make up the combination, optimizers, dropout values, batch size values , and learning rate values , and the relationship between the variables are shown in Figure 3.9. By trying all combinations, the best result can be learned by doing grid research. Deep learning-based classification networks should be chosen considering their speed and accuracy [79]; therefore, in this study, we tested four different networks and compared the speeds and accuracy of the networks to provide a comprehensive comparison.

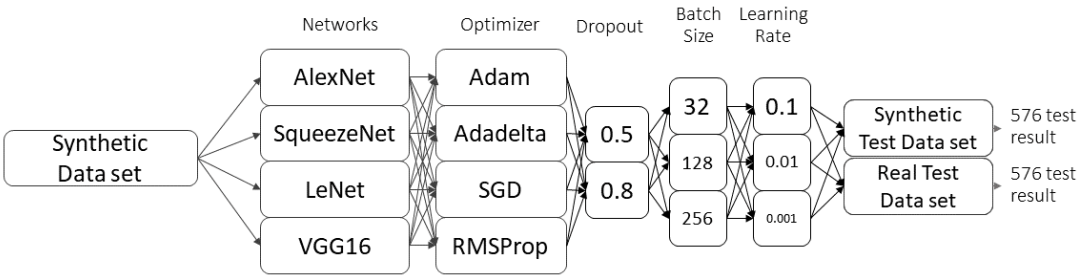

Figure 3.9 Schematic representation of combinations of networks.

3.6 Setup of Experiments

Dataset creation, consisting of synthetic and real RW-UAV, bird's images, training, validation, and test data has been completed. The dataset distributions are shown in the schematic image shown in Figure 3.7. With the AI-based model, necessary images have been prepared for RW-UAV and bird classification experiments. A total

of 7 main experiments have been designed to enable the synthetic data to be trained with an AI model to predict real images.

Each network has been trained using each optimizer, dropout, batch size and learning value. Due to the number of variables and the number of networks, a total of 288 networks were trained. The 288 trained networks were tested with both synthetic data and real data. This process has been repeated for 7 experiments. The schematic representation showing the application of the DL model training process together with the experiments is given in Figure 3.10. A total of 2016 training procedures and 4032 tests were carried out for 7 experiments. A script has been developed to automate this process. The script performs the training and testing process by changing the values of all variables cyclically and reports the test results. All training and testing stages were carried out by using the GPU computing system (GeForce GTX 1080ti, Nvidia Corp., Santa Clara, CA, ABD), 128 gigabyte (GB) random access memory (RAM), I7 processors and the TensorFlow-based [42] Keras [43] library with python programming language.

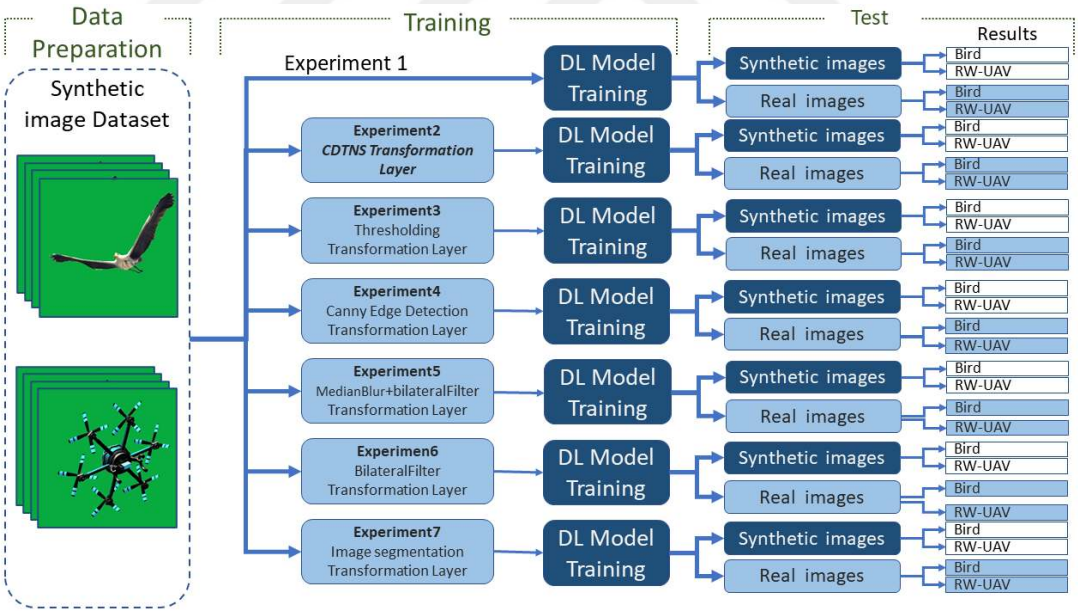

Figure 3.10 Training and testing process for all experiments.

7 different experiments using image processing methods have been designed. There is a noticeable difference between synthetic data and real data. Synthetic data are unnatural images produced using 3D models and a game engine. Realism cannot be fully achieved in synthetic data. Real data are images created by taking a camera

from real objects such as RW-UAV and birds. Real data are data with more truth than synthetic data [80]. This difference negatively affects the success of classifying real data of the model trained with synthetic data [80]. The subject of this study is that AI trained with synthetic data can successfully classify real data. Due to the real difference between the two data types, the AI model trained with synthetic data is unlikely to recognize real data. Therefore, a search has been made for a method that will make the two data types look like each other and approximate them as reality. Seven different experiments, named below, have been designed. The first of these experiments is to test AI, which is directly trained with synthetic data, with real data. In other words, there is no image processing technique between the AI training model and the input consisting of synthetic data in experiment 1. The other 7 experiments are about adding image processing methods to the input of DL-based networks. These methods are a feature extraction layer before DL based network. 6 of 7 experiments are feature extraction methods used in literature. Experiment 2 is the method of feature extraction we recommend. By using feature extraction methods, it is aimed to reduce the features of synthetic and real images and increase the performance of the DL network [81].

- Experiment 1: DL based network training
- Experiment 2: CDNTS layer and DL based network training
- Experiment 3: Thresholding and Dl based network training
- Experiment 4: Canny edge detection and DL based network training
- Experiment 5: Median Blur, Bilateral Filter and DL based network training
- Experiment 6: Bilateral Filter and DL based network training
- Experiment 7: Image segmentation and DL based network training

The performance, computation times, validations, and test accuracies of the methods were measured. The classification results measured from experimental setups were verified with the validation methods of the F-score [82] and the area under the curve (AUC). AUC and test accuracy parameters were computed to analyze the performance and success of the network. The F-score value is a parameter that calculates test accuracy using by recall and precision values. The F-score calculation has been made in all the experiments and it was measured how successful the trained network could classify. The F-score calculation method is applied using True

positive, true negative, and False-negative values. For this reason, it not only determines the true and false ratio, but also the performance of the classes themselves. F-score is an acceptable verification method in scientific studies [82].

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F_{score} = 2.\frac{Precision\,.Recall}{Precision + Recall} \tag{3.4}$$

where *TP*, *TN*, *FP*, and *FN* are true positives, true negatives, false positives, and false negatives, respectively.

3.6.1 Experiment 1: DL based network Training

In Experiment 1, DL-based networks have been trained with synthetic data, and the success of classifying synthetic test data and real test data has been measured. During the training process, all DL-based networks shown in AI Figure 3.10 have been trained using combinations. The aim of the experiment is to use synthetic images in DL based network without any image processing. Figure 3.11 shows examples of synthetic training, synthetic testing, and real test images used in the experiment.

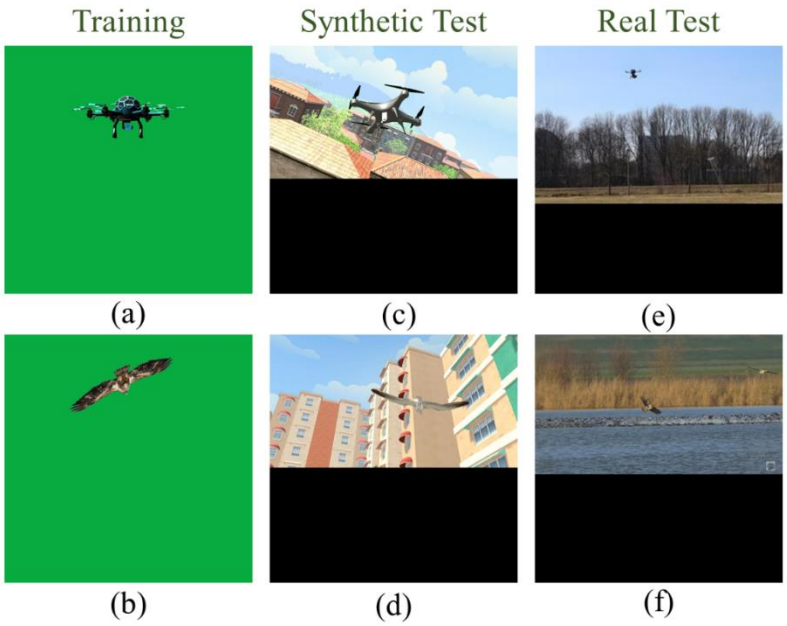

Figure 3.11 Train, synthetic, and real test dataset sample images of Experiment 1

3.6.2 Experiment 2: CDNTS Layer and DL Based Network Training

We propose a new approach to increase the accuracy of the DL models trained with synthetic images and tested with real images. This approach, as in other experiments, is designed to reduce the difference between synthetic images and real images. The proposed approach, corner detection, and nearest three-point selection (CDNTS) is a hybrid method of corner detection methods and the nearest three-point search (NTS) method for feature extraction that can be utilize with DL models [83]. It has been obtained by using the image processing method used in the literature with a new method. Corner detection-based methods are commonly utilized in visual tracking [44], image matching and photography [45], robotic path finder's [49], etc. This method is a method that is generally used to detect corners in image-based studies. The NTS part of the hybrid CDNTS layer, which makes up the other half, is a function that looks for the closest points. NTS issues, which essentially try to measure the distance between two or more points, are a popular topic in literature research investigations. This method is used to group pixels in two-dimensional (2D) images and vertices in three-dimensional (3D) point cloud data and find the closest pixels and vertexes [84]. It's utilized in decision tree problems to create a decision tree by determining the distances between the weights of classes [85]. The CDNTS layer proposed in this work is a feature extraction strategy that employs the "corner detection" and "nearest point search" methods. The three major actions of the CDNTS layer are corner point coordinate detection, corner distance computation, and drawing lines between the closest three corners. The original image is changed into a set of corner points and lines because of these procedures. The corner detection-based CDNTS is a hybrid layer that increases the accuracy of DL networks trained on synthetic pictures in real-image classification.

The choice of method for image feature extraction has been shown in some studies to be a successful way to improve the performance of a CNN-based algorithm [76]. Simple and lower-dimensional RW-UAV images made from lines drawn between the image's corner points could help DL algorithms perform better [81]. Therefore, a layer called the corner detection, and nearest three-point selection (CDNTS) layer has been joined to the front of the CNN-based networks. This layer, developed for the input of CNN-based networks, is an algorithm that finds corner points in the images and connects the three closest corner points with lines. To detect corners, the

OPENCV framework's "good features to track" library was used. The corner detector library has been developed by Shi-Tomasi with the authors in [86-87]. The position data from the corner detector was used to draw lines between the closest three corner points. In the literature, there are studies that draw continuous lines between points [88]. The continuity of the lines drawn between the points was ignored in our research. A line was drawn between the three closest points to each point. Algorithm 2 has shown CDNTS layer flow diagram. The OPENCV library was used to create the corner detection algorithm, which is part of the CDNTS layer. Another part of the CDNTS layer, the nearest three-point selection part, has been coded by the authors. A general flow diagram of the algorithm is shown in Figure 3.12.

image = Read image data from dataset;
corners = Find corner using Opencv goodFeatureToTrack(image);
foreach $corner \in corners$ **do**
  foreach $corner_{target} \in corners$ **do**
    $vector_{\text{len}} = \sqrt{(corner_{\text{xPos}} - corner_{\text{targetXPos}})^2 + (corner_{\text{yPos}} - corner_{\text{targetYPos}})^2}$;
    $min1st, min2nd, min3th \leftarrow 99999;$;
    if $vector_{len} \leq min1st_{vector}$ **then**
      $min3th \leftarrow min2nd;$
      $min2nd \leftarrow min1st;$
      $min1st \leftarrow vector_{\text{len}};$
    else if $vector_{len} \leq min2nd$ **then**
      $min3th \leftarrow min2nd;$
      $min2nd \leftarrow vector_{\text{len}};$
    else if $vector_{len} \leq min3th_{vector}$ **then**
      $min3th \leftarrow vector_{\text{len}};$
  end
  DrawLine (from $\text{corner}_{\text{targetXYPos}}$ to $\text{min1st}_{\text{XY Position}}$) ;
  DrawLine (from $\text{corner}_{\text{targetXYPos}}$ to $\text{min2nd}_{\text{XY Position}}$) ;
  DrawLine (from $\text{corner}_{\text{targetXYPos}}$ to $\text{min3th}_{\text{XY Position}}$) ;
end
Corner detection and nearest three-point selection is completed.

Figure 3.12 Corner detection and nearest three-point selection layer.

Algorithm 2 has been utilized to find the corner points of the images and to plot a new image with lines between the three closest corner points. This new image consisting of only lines and corners was then given to the CNN-based network. Sample output images of Algorithm 2 have been shown in Figure 3.12.

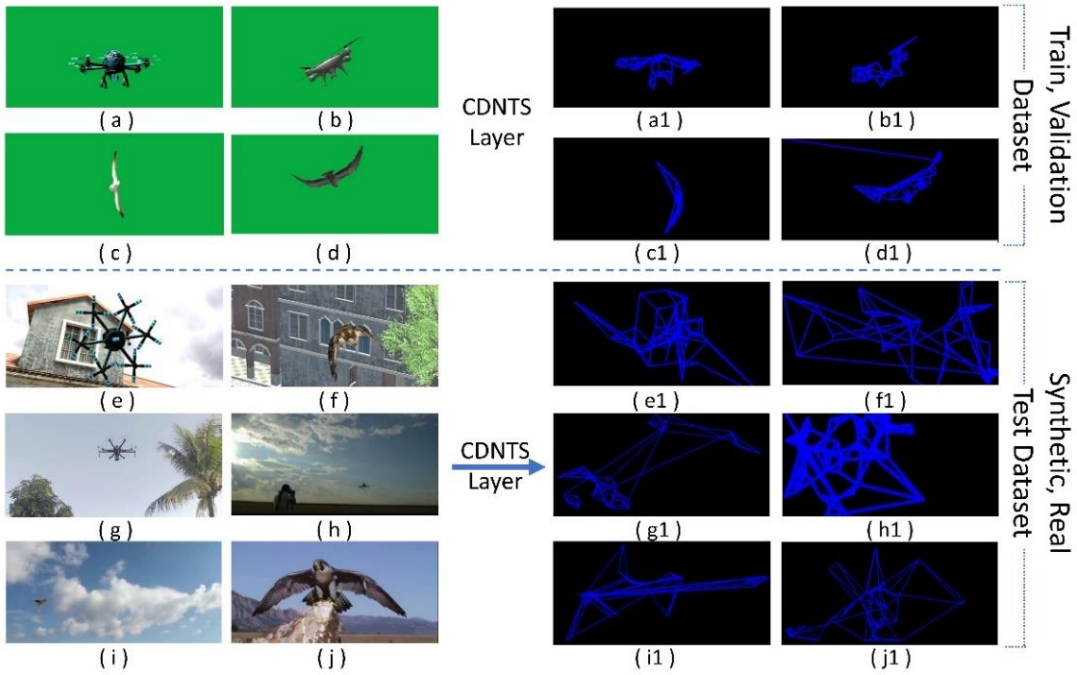

Figure 3.13 Sample output images of the CDNTS algorithm. (a–j) training, synthetic test, and real test images. (a1-j1) transformed images.

A schematic representation of the CDNTS layer nearest three-point selection operation is shown in Figure 3.13. The letters y, x, c, and L determined on the graph refer to the image pixel horizontal coordinate, image pixel vertical coordinate, detected corner, and distance between vertices, respectively. The formula for calculating the variable L is given by (3.5).

$$L_{c_n->c_m} = \sqrt{(x_{Cn} - x_{Cm})^2 + (y_{Cn} - y_{Cm})^2} \tag{3.5}$$

where $L_{cn->cm}$, x_{cn}, x_{cm}, y_{cn}, and y_{cm} values are the vector length between selected corner and target corner, selected corner x axis position, target corner x axis position, selected corner y axis position, and target corner y axis position, respectively.

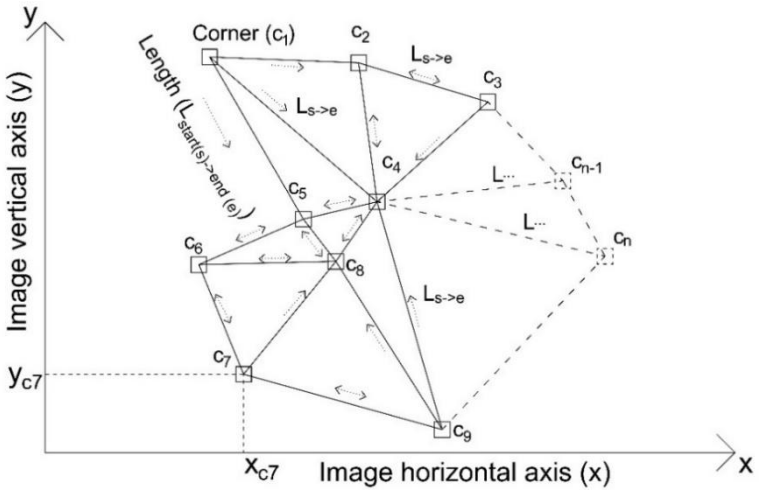

Figure 3.14 CDNTS layer, nearest three-point selection process schematic representation.

The distance calculation formula has been repeated in a loop to define the distances of all vertices to other corners. In the loop, the closest point has been searched for, and a line was drawn to the nearest corner. A previously drawn line could be re-drawn in case two vertices were the nearest corners to each other. This overlap process is shown in the schematic representation with a double-sided arrow. The one-sided arrows show the starting and ending points of the lines between the corners.

3.6.3 Experiment 3: Thresholding and DL Based Network Training

In the experiment, the pixel values of the images forming the dataset have been changed according to the determined threshold value. Images have been transformed using the THRESH_TOZERO [86] function in the OpenCV library. The experiment has been designed to reduce the difference between synthetic images and real images. The THRESH_TOZERO mathematical formula is shown in the eq 3.6. The function checks the threshold for each pixel of the image, and if it is greater than the threshold, it leaves the pixel value as it is. If the pixel value of the image is less than the threshold value, it will set the value to 0.

$$dst(x,y) = \begin{cases} src(x,y) \; if \; src(x,y) > thresh \\ 0 \qquad\qquad otherwise \end{cases} \tag{3.6}$$

The DST function represents the new image, the x,y values represent the width and height of the pixel in the image, and the src source image. The graph shown in Figure

3.15 is the output graph of the threshold zero function. All pixel values below the Threshold line are 0, others are at their original values.

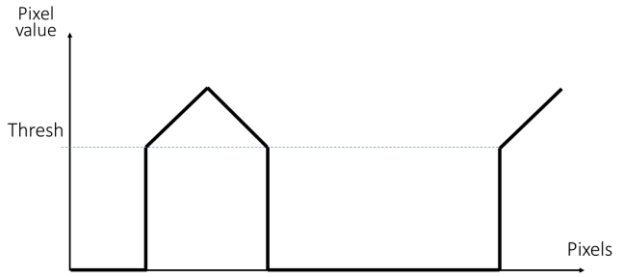

Figure 3.15 Output graph of threshold zero function.

The image shown in Figure 3.16 contains samples from the training, validation, test synthetic, and test real datasets. The images in the training, validation, test synthetic, and test real datasets are original and after the threshold function is applied.

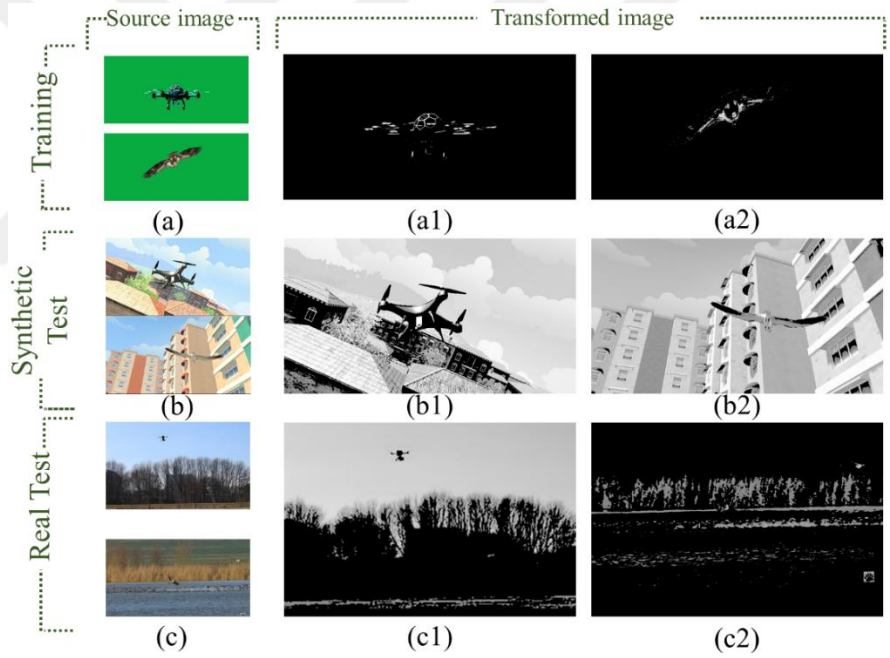

Figure 3.16 Sample output images of the threshold zero function. (a–c) training, synthetic test, and real test images. (a1-c12) transformed images.

3.6.4 Experiment 4: Canny Edge Detection and DL Based Network Training

In the experiment, synthetic images have been transformed into other images consisting only of edges by subjecting them to the edge detection algorithm.

AI training is completed using these transformed images. Synthetic and real test images in the test dataset were also transformed into images consisting of edges by

using an edge detection algorithm. The AI model trained using edge images was expected to predict test images consisting of edges. The estimation results have been noted. As in other experiments, the aim of this experiment is to eliminate the differences between synthetic images and real images and try to convert them into images that are like each other. The edge detection algorithm used in the experiment is the Canny edge detection [89] algorithm. John F. Canny. The Canny edge detection algorithm, developed by Canny, consists of noise reduction, finding intensity gradient of image, non-maximum suppression, and hysteresis thresholding stages.

1. Noise reduction

The noise reduction stage consists of a 5x5 matrix gaussian filter. It is obtained by averaging the 5x5 matrix in the example shown in figure 3.17. The sum of the values of the 5x5 part of the whole matrix is found and each pixel value is divided by the total value. This process reduces the sharpness between the pixels of the image, allowing us to obtain a smoother image [90].

$$\frac{1}{159}\begin{bmatrix} 4 & 12 & 8 & 20 & 10 \\ 5 & 5 & 7 & 2 & 1 \\ 1 & 2 & 7 & 5 & 10 \\ 6 & 4 & 12 & 8 & 0 \\ 8 & 6 & 6 & 8 & 2 \end{bmatrix}$$

Figure 3.17 Noise reduction matrix of image.

2. Finding Intensity Gradient of the image

The smoothed image is then filtered in both the horizontal and vertical directions with a Sobel kernel to obtain the first derivative in both the horizontal (Gx) and vertical (Vx) directions (Gy). We can calculate the edge gradient and direction for each pixel using these two images. The direction of the gradient is always perpendicular to the edges. It's rounded to one of four angles: vertical, horizontal, and two diagonals.

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2} \tag{3.7}$$

$$Angle\ (\Theta)\ = tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{3.8}$$

3. Non-maximum Suppression

After obtaining the gradient magnitude and direction, the image is fully scanned to remove any unwanted pixels that do not form the edge. For this, each pixel is checked to see if it is a local maximum in its neighborhood in the gradient direction. Point A is perched on the cliff's edge (in vertical direction). The gradient's direction is perpendicular to the edge. Process has been shown in Figure 3.18. Points B and C are oriented in a gradient. To see if point A forms a local maximum, it is compared to points B and C. If this is the case, it will be considered for the next stage; otherwise, it will be suppressed (put to zero) [86].

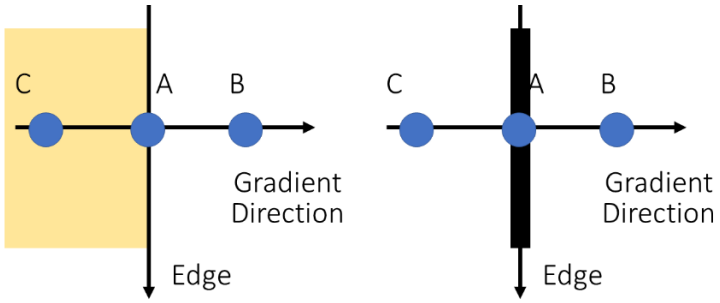

Figure 3.18 Gradient magnitude and direction.

4. Hysteresis Thresholding

This stage determines which edges are genuine and which are not. We'll need two threshold values, minVal and maxVal, for this. Any edges with an intensity gradient greater than maxVal are certain to be edges, while those with an intensity gradient less than minVal are certain to be non-edges and should be discarded. Based on their connectivity, those who fall between these two thresholds are classified as edges or non-edges. They are part of edges if they are connected to "sure-edge" pixels. Otherwise, they are discarded as well. Because edge A is greater than maxVal, it is referred to as a "sure-edge." Even though edge C is below maxVal, it is connected to edge A, so it is considered a valid edge, and we get the full curve. Edge B, on the other hand, is connected to no "sure-edge" and thus is discarded, even though it is above minVal and in the same region as edge C. As a result, it is critical that we select minVal and maxVal appropriately to obtain the desired result.

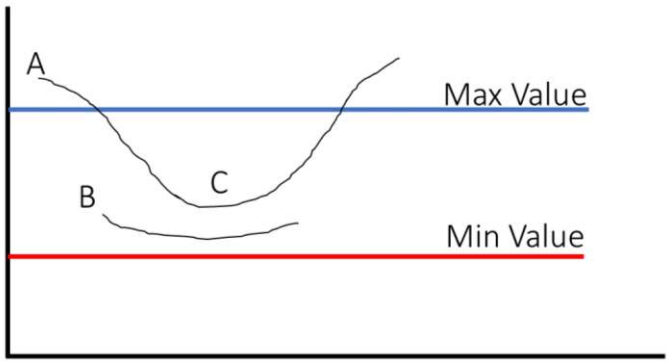

Figure 3.19 Hysteresis thresholding graph

The image shown in Figure 3.20 contains samples from training, validation, test synthetic, and test real datasets. The images in the training, validation, test synthetic, and test real datasets are original and after the threshold function is applied.

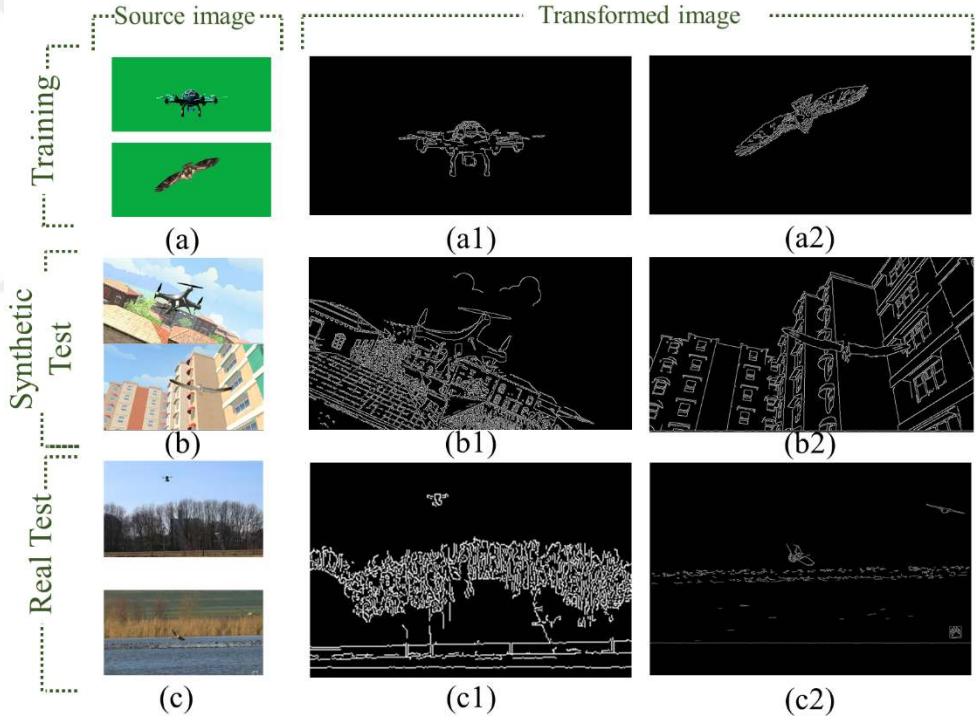

Figure 3.20 Sample output images of canny edge detection (a–c) training, synthetic test, and real test images. (a1-c12) transformed images.

3.6.5 Experiment 5: Cartoon Effect and DL Based Network Training

In the experiment, the cartoon effect was applied to the images forming the dataset. The cartoon effect is a method that converts image images into cartoon images [91]. Thanks to this method, the characteristic lines of the objects in the image are brought

to the fore, and most studies use the total variation (isotropic), defined as the total magnitude of vertical and horizontal discrete gradients of images [92].

The Cartoon effect occurs when three functions are applied consecutively to the image. Median blur, adaptive threshold, and bilateral filtering functions are applied on the image respectively. The purpose of applying the Cartoon effect is to try to eliminate the differences between synthetic images and real images in terms of appearance and realism. It is thought that the prediction success of the AI-based model will increase when the reality differences between synthetic and real images decrease.

1. Median Blur

As in one-dimensional signals, images also can be filtered with various low-pass filters (LPF). LPF helps in removing noise, bluring images. Image blurring is achieved by using a low-pass filter. This method is useful for removing noise. It removes high-frequency content from the image, making the border lines a bit blurry.

2. Adaptive Threshold

The adaptive threshold is a method that adaptively eliminates the light value of the pixels in the image. The simple threshold method works based on a specified threshold value. If the pixel value of the image is less than the threshold value, it takes its own value if it is greater than zero. This method has been described in previous experiments. At the adaptive threshold value, the threshold value varies for each pixel. The adaptive threshold method gives better results under locally varying light values of the image. In this method, the image is divided into local regions within itself, and a separate threshold value is determined for each local region. For this process, a threshold value is determined by subtracting the average of the pixel values of the local region. Or the threshold value is determined by subtracting the average of the maximum and minimum pixel values [93].

3. Bilateral Filtering

The bilateral filter is used as a non-linear, edge-preserving, and noise-reducing filter [94]. The filter replaces the intensity of each pixel with the weighted average intensity value from nearby pixels. The bilateral filter is defined in eq 3.9.

$$I^{filtered}(x) = \frac{1}{w_p}\sum_{x_i \in \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{3.9}$$

and normalization term W_p, is defined as

$$W_p = \sum_{x_i \in \Omega} f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{3.10}$$

Where:

I^{filtered} is the filtered image.

I is the original input image to be filtered;

x are the coordinates of the current pixel to be filtered.

f_r is the kernel for smoothing difference intensities.

g_s is the spatial kernel for smoothing difference in coordinates.

W_p is assigned weight

In the image shown in Figure 3.21, there are samples taken from training, validation, test synthetic, and test real datasets. The images in the training, validation, test synthetic, and test real datasets are original and after the threshold function is applied.

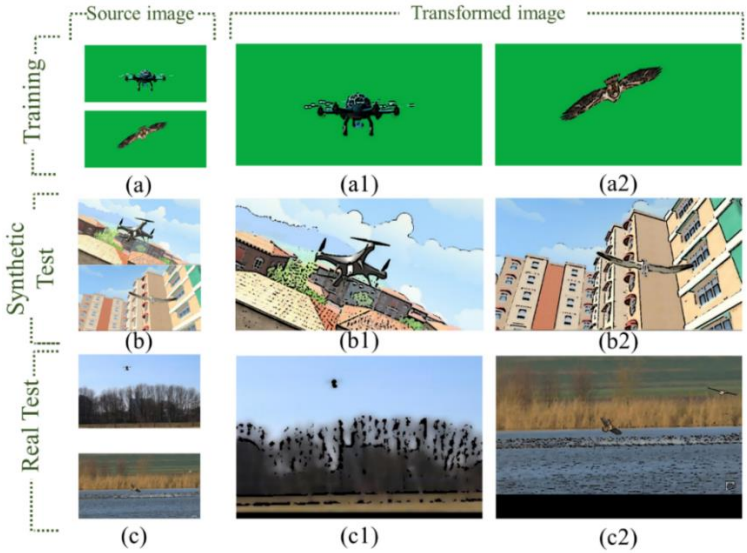

Figure 3.21 Sample output images of the cartoon effect (a–c) training, synthetic test, and real test images. (a1-c12) transformed images.

3.6.6 Experiment 6: Bilateral Filter And DL Based Network Training

In the experiment, all the images were transformed by applying the bilateral filter function, which is one of the sub-functions of the previous experiment. The binaural filter function adds appearance blur to the image. The purpose of applying the filter is to reduce the difference between synthetic and real images. In the image shown in Figure 3.22, there are samples taken from training, validation, test synthetic, and test real datasets. The images in the training, validation, test synthetic, and test real datasets are original and after the threshold function is applied.

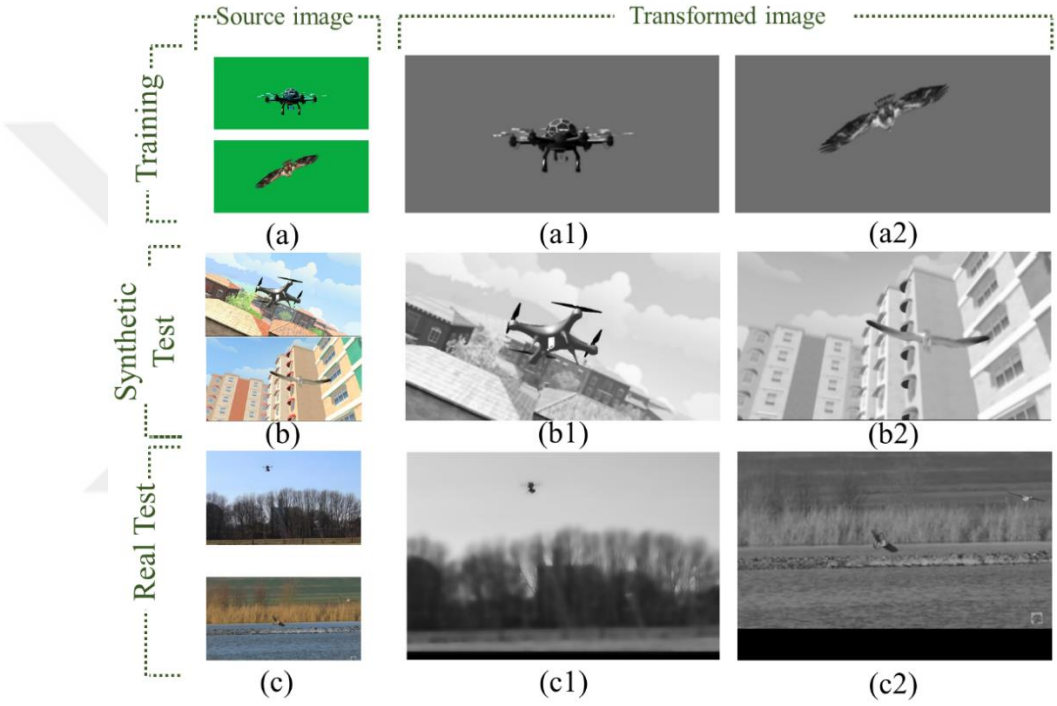

Figure 3.22 Sample output images of the bilateral filter. (a–c) training, synthetic test, and real test images. (a1-c12) transformed images.

3.6.7 Experiment 7: Image Segmentation and DL Based Network Training

In the experiment, new images were created by automatic segmentation of the objects in the images. The segmentation method used is Felzenszwalb's [95] segmentation. All synthetic and real images were segmented and converted into new images using the segmentation method. When RW-UAV and bird objects in synthetic and real images were segmented, it was expected that the difference between synthetic and real images would disappear. Felzenszwalb segmentation is a graph-based method used to define boundary regions of an image. The method creates a new picture by determining the boundaries of the objects in the images. The

function creates segmented RGB images on the image using fast minimum spanning tree-based clustering. In the image shown in Figure 3.23, there are samples taken from training, validation, test synthetic, and test real datasets. The images in the training, validation, test synthetic, and test real datasets are original and after the threshold function is applied.

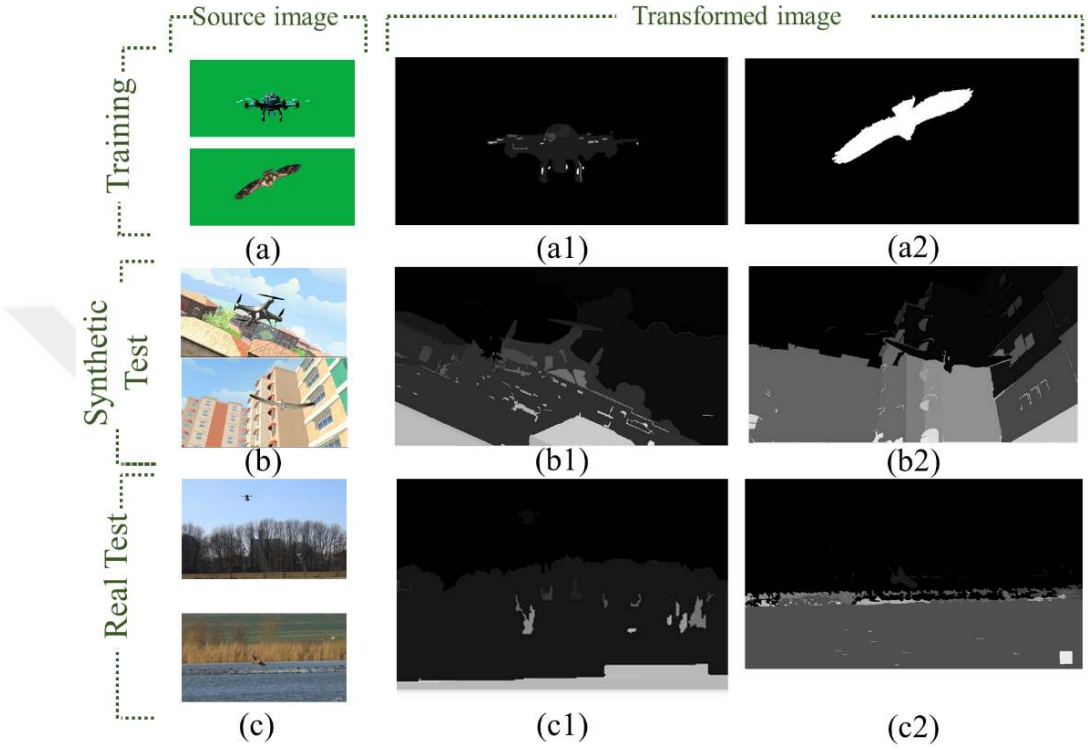

Figure 3.23 Sample output images of auto segmentation (a–c) training, synthetic test, and real test images. (a1-c12) transformed images.

CHAPTER IV

TRAINING AND TEST EXPERIMENTAL RESULTS

4.1 Experimental Verification

We examined the AI training using synthetic data under seven experiments. In methods, four different networks were trained to determine the most successful network and optimizer with a combination of four optimizers, two dropout values, three batch size values, and three learning rate values. Therefore, a total of 2016 trained combinations of models were obtained, including 288 different trained models in each experiment. Training has been performed using synthetic RW-UAV and bird images for each combination of each experiment. The aim of the study, which consisted of 7 experiments in total, was to develop a deep learning-based AI model that classifies real images using synthetic images. The characteristics of the experiments listed below and their differences from each other are explained in the methodology section.

- Experiment 1: DL based network training
- Experiment 2: CDNTS layer and DL based network training
- Experiment 3: Thresholding and Dl based network training
- Experiment 4: Canny edge detection and DL based network training
- Experiment 5: Median Blur, Bilateral Filter and DL based network training
- Experiment 6: Bilateral Filter and DL based network training
- Experiment 7: Image segmentation and DL based network training

In this section, the measured results for each experiment and each model and the time cost of the feature extraction layer, test results on real images, and test results on synthetic images are explained.

4.1.1 Test Result Verification

As mentioned in the methodology section, AI models trained using synthetic images were put into the classification test separately from real and synthetic test groups. The images used in the training and the images used in the testing process are different from each other. How the images are produced and how the similarities are eliminated are explained in the method section. F-score, accuracy (Acc) Area under of Curve (AUC) values after synthetic image test and real image test of the experiments are listed separately. The AUC value is the area under the precision and recall graph used in the F-score calculation. Acc value is the accuracy value calculated with the arithmetic mean. The accuracy of the models during the testing phase has been found by calculating the F-score value. F-score is a frequently used verification method in ML studies in the literature [46][75][83]. An experiment result with an F-score of 50% indicates that artificial intelligence has failed. In the image classification process, if the AI model is 50% correctly classified, the same result can be obtained by flipping a coin instead of the AI model. Therefore, when creating tables, training combinations with F-score values below 55% are shown as unsuccessful results by grouping them.

4.1.2 Computational Times

The training times of all networks used in the training and testing phase were tracked, and the durations of all networks were calculated for 10 epochs to make an accurate comparison in terms of time consumption. All training and testing stages' time costs were calculated on a system using the GPU computing system (GeForce GTX 1080ti, Nvidia Corp., Santa Clara, CA, USA), 128 GB RAM, I7 processors, and the TensorFlow-based and Keras library.

4.1.3 Precision Recall Graph

Precision and recall plots are created using the F-score value calculated during the testing process of the trained DL model. The ideal graph shape is shown in Figure 4.1. The graph is the graph of the F-score value with a 100% success rate. The meaning of this graph is that the DL model has correctly classified all the elements in all the classes studied.

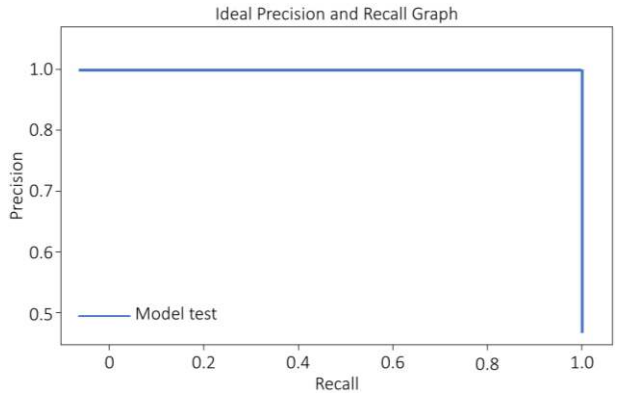

Figure 4.1 Ideal precision and recall graph.

4.2 Experimental Results

4.2.1. Experiment 1 Results

In the experiment, the performance of synthetic data training of standard deep learning-based AI models on real data was measured. Experiment 1 consists of training and testing results of known deep learning-based AI models. An extra layer of feature extraction is not used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.2, the methodology of experiment 1 is shown schematically. Synthetic images are introduced directly into AI models training. And the Trained AI model is directly tested with synthetic and real images.

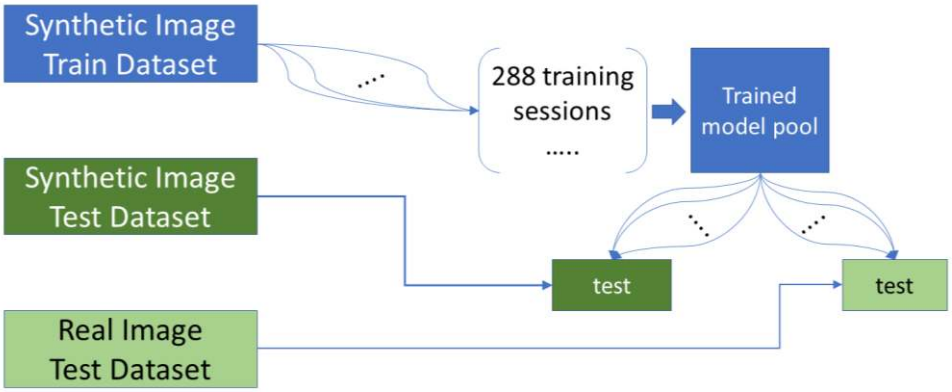

Figure 4.2 Experiment 1 train and test scenario schematic representation.

The experimental results trained and tested in accordance with the schematic representation are listed in Table 4.1. Testing was carried out with each combination of multiple DL-based AI models, optimizers, dropout values, batch size values , and learning rate values. The table contains values greater than 55% of the accuracy test

results. The accuracy of the test results of the combinations not in the table is less than 55%.

Table 4.1 Exp 1: F-score, accuracy (Acc), and AUC values of the test results.

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
1	LeNet, AdaDelta, 128, 0.001	69	37	58.4	58.4	<55			
2	LeNet, SGD, 128, 0.001	52	69	62	62.4	34	67	56	53
3	SqueezeNet, AdaDelta, 128, 0.01	38	68	57	57.4	**48**	**66**	**59**	**57**
4	SqueezeNet, AdaDelta, 256, 0.1	69	71	70	70	<55			
5	SqueezeNet, Adam, 128, 0.1	49	64	58	57.5	51	63	58	56.8
6	SqueezeNet, Adam, 32, 0.01	**66**	**80**	**74**	**74.4**	30	69	57	53.4
7	SqueezeNet, RMSprop, 32, 0.01	24	70	57	56.8	0.9	71	56	51
8	SqueezeNet, RMSprop, 32, 0.1	69	69	69	69.3	44	63	56	54.3
9	VGG16, SGD, 32, 0.001	70	35	59	58.5	**58**	**69**	**64**	**63.2**
10	VGG16, SGD, 32, 0.01	**76**	**59**	**70**	**69.7**	56	59	57	57.6

As can be seen from the results, the combination with the highest performance on synthetic test data was obtained using the squeezeNet network with the Adam optimizer. In combination 6, which provides 74% success, the batch size is 32 and the learning rate is 0.01. Network number 7, trained with the same network, batch size value and learning rate value, and different optimizers, achieved 56.8% success. It has been observed that the optimizer affects the success of the training. On the

other hand, the performance is different in combinations 5 and 6 using the same network, optimizer, and different batch sizes. This shows that the batch size value is effective in the performance. In addition, the only different parameter in the combination of 7 and 8 is the learning rate. This again affected my performance. In Experiment 1, the highest performance value in the classification of synthetic images is combination 6 with 74% accuracy. The highest success rate in classifying real images is combination number 9 with 64% accuracy. Direct AI model training has not achieved remarkable success in classifying real images. In fact, it has not achieved remarkable success in the classification of synthetic images.

Table 4.2 Experiment 1, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	5.22	18	22.7
LeNet	4.63	21.5	45
SquuezeNet	5.46	17.5	38.5
VGG16	8.14	23.5	52

The time costs of each experiment for each combination in Experiment 1 have been measured. In the training phase of the networks, the elapsed time value of all combinations has been measured separately. The mean value of the measured values is shown in table 4.2. All measurements were made on a GPU-based system with the computer hardware specifications specified in the computational time section. The time cost of an image for AI training was measured at least 5.22 ms and at most 8.14 ms. While the trained AI model is performing synthetic image classification, the minimum time cost for an image is 17.5 ms, and the maximum time cost is 23.5 ms. While the trained AI model is performing real image classification, the minimum time cost for an image is 22.7 ms, and the maximum time cost is 52 ms. The reason for these time differences is the difference in the structural complexity of the networks.

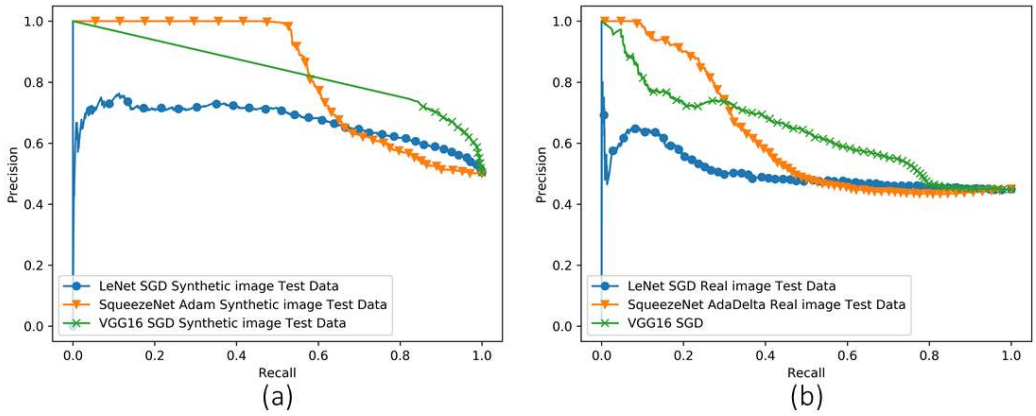

Figure 4.3 Precision and recall graph of experiment 1 synthetic and real image test results.

The results of the tests performed with the synthetic and real image test data of Experiment1 are given in the table. The precision and recall graphs of the most successful combinations of the nets shown in Table 4.1 is shown in Figure 4.3. The graph labeled as (a) in the figure is the precision and recall graph of the test data set consisting of synthetic images. The graph labeled as (b) in the figure is the precision and recall graph of the test data set consisting of real images. The precision and recall graph of a classification process with 100% success in the test process is explained in the method section. Considering the ideal precision and recall graph, experiment 1 test result precision, and recall graph show the failure of the training of the network. The squeezeNet network classification process shown in figure 4.3 is the most successful. This graph is the graph of the number 6 combination listed in Table 4.1.

4.2.2. Experiment 2 Results

In the experiment, the effect of synthetic data training of our proposed CDNTS layer on real data has been measured. Experiment 2 consists of the training and test results of the network model created by adding a layer called CDNTS to the entry point of known deep learning-based AI models. A feature extraction layer called CDNTS is used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.4, the methodology of experiment 2 is shown schematically. Before entering the AI training model, the synthetic images have been transformed by passing through a layer called CDNTS. In addition, the Trained AI model was directly tested with synthetic and real images.

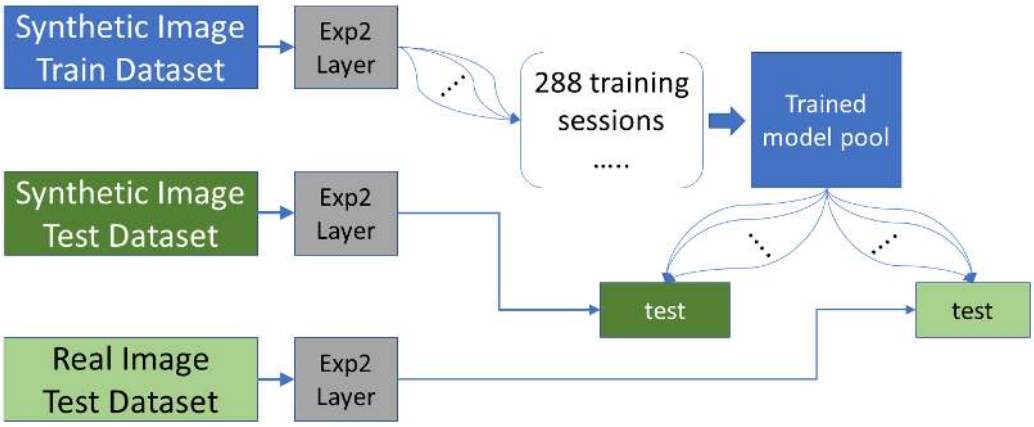

Figure 4.4 Experiment 2 train and test scenario schematic representation.

The experimental results trained and tested in accordance with the schematic representation are listed in Table 4.3 Since testing is performed with each combination of multiple Deep learning-based AI models, optimizers, dropout values, batch size values , and learning rate values, the table is filled for all combination values.

Table 4.3 Exp 2: F-score, accuracy (Acc), and AUC values of the test results.

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
1	AlexNet, AdaDelta, 128, 0.001	72	40	62	62	67	56	62	64.7
2	AlexNet, AdaDelta, 128, 0.01	81	72	77	77.3	65	59	62	63.7
3	AlexNet, AdaDelta, 128, 0. 1	85	82	84	83.7	64	65	65	65.4
4	AlexNet, AdaDelta, 256, 0.01	82	77	80	79.8	<55			
5	AlexNet, AdaDelta, 256, 0.1	86	84	85	85.3	60	68	65	64.2
6	AlexNet, AdaDelta, 32, 0.001	81	74	78	78.1	<55			

Table 4.3(cont.) Exp 2:F-score, accuracy (Acc), and AUC values of the test results

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
7	AlexNet, AdaDelta, 32, 0.01	84	79	82	81.5	62	62	62	62.5
8	AlexNet, AdaDelta, 32, 0.1	88	87	88	87.5	<55			
9	AlexNet, Adam, 32, 0.1	83	80	81	81.4				
10	AlexNet, RMSprop, 32, 0.001	87	86	86	86				
11	AlexNet, SGD, 128, 0.001	76	60	70	70	72	72	72	72.9
12	AlexNet, SGD, 128, 0.01	81	73	78	77.6	66	62	64	65.3
13	AlexNet, SGD, 128, 0.1	86	87	87	86.7	55	73	67	64.5
14	AlexNet, SGD, 256, 0.001	81	74	78	77.8	**73**	**77**	**75**	**74.8**
15	AlexNet, SGD, 256, 0.01	83	77	81	80	65	62	64	64.8
16	AlexNet, SGD, 256, 0.1	**90**	**90**	**90**	**89.8**	52	73	65	63.3
17	AlexNet, SGD, 32, 0.001	81	73	78	77.6	<55			
18	AlexNet, SGD, 32, 0.01	84	80	82	82				
19	AlexNet, SGD, 32, 0.1	86	84	85	85				
20	LeNet, AdaDelta, 128, 0.001	83	77	80	80	64	56	61	62.1

Table 4.3(cont.) Exp 2:F-score, accuracy (Acc), and AUC values of the test results

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
21	LeNet, AdaDelta, 128, 0.01	**86**	**85**	**85**	**85**	<55			
22	LeNet, AdaDelta, 128, 0.1	81	71	77	76.9	64	54	60	61.3
23	LeNet, AdaDelta, 32, 0.001	85	83	84	84	<55			
24	LeNet, RMSprop, 256, 0.1	<55				59	71	66	65.1
25	LeNet, SGD, 128, 0.001	86	82	84	84.3	70	75	73	72.7
26	LeNet, SGD, 128, 0.01	87	84	86	85.7	70	78	74	73.6
27	LeNet, SGD, 32, 0.001	88	86	87	86.7	**72**	**81**	**77**	**76.2**
28	SqueezeNet, AdaDelta, 256, 0.1	76	56	69	68.5	70	72	71	71.2
29	SqueezeNet, AdaDelta, 32, 0.1	80	77	79	78.9	82	89	86	84.8
30	SqueezeNet, Adam, 128, 0.001	69	74	72	71.8	78	87	84	81.8
31	SqueezeNet, Adam, 128, 0.01	81	73	78	77.8	85	90	88	87.1
32	SqueezeNet, Adam, 256, 0.001	74	60	69	68.6	82	87	85	83.9
33	SqueezeNet, Adam, 256, 0.01	**84**	**83**	**84**	**83.7**	82	89	86	84.5
34	SqueezeNet, Adam, 32, 0.001	75	64	70	70.2	78	82	80	80

Table 4.3(cont.) Exp 2:F-score, accuracy (Acc), and AUC values of the test results

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
35	SqueezeNet, Adam, 32, 0.01	74	63	69	69.2	86	91	89	87.8
36	SqueezeNet, RMSprop,128, 0.001	71	79	76	75.6	79	88	85	82.8
37	SqueezeNet, RMSprop, 256, 0.01	71	30	59	58.5	**88**	**91**	**90**	**88.9**
38	SqueezeNet, SGD, 128, 0.01	<55				77	87	83	81.3
39	SqueezeNet, SGD, 128, 0.1	81	74	78	78	84	90	88	86.4
40	SqueezeNet, SGD, 32, 0.01	80	79	80	79.8	83	89	86	85
41	SqueezeNet, SGD, 32, 0.1					76	72	74	75.8
42	VGG16, SGD, 32, 0.001	82	82	82	81.8	<55			
43	VGG16, SGD, 32, 0.01	**87**	**88**	**88**	**87.4**				

The results of Experiment 2 are listed in Table 4.3 As a result of the experiment, the models subjected to the classification test process with synthetic and real images gave successful results. 43 successful results are listed in Table 4.3. In the classification test process of synthetic images, the number 16 combination provided a 90% success rate. SGD optimizer, 256 batch size values, and 0.01 learning rate values are used in the AlexNet network, which is 90% successful on synthetic images. In the real image test process, test 37 was the most successful combination with a rate of 88.9%. The Adam optimizer, which makes up the number 33 combination, achieved a success rate of over 84% in the classification of both

synthetic and real images in the SqueezeNet network with its 256-batch size value. Over 74% success has been achieved in 4 different networks. The proposed model has been successful in recognizing both synthetic images and real images.

The CDNTS layer is explained in detail in the method section. This layer is a feature extraction layer that we recommend. The purpose of using the layer is to minimize the difference between real and synthetic images. In this way, it is aimed that the AI model can be trained with synthetic images to recognize real images in the real world. The test results of Experiment 2 are a feature extraction method that increases the success of the CDNTS layer up to 90%. In addition to the advantage of increasing the performance of the layer, it has a disadvantage such as extending the training and testing time. The CDNTS layer is applied to every image in the training, validation synthetic test, and real test image dataset. The time taken to transform an average image is 36.7 milliseconds.

Table 4.4 Experiment 2, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	4.88	15	38.5
LeNet	5.03	18	45
SquuezeNet	4.98	14	38.5
VGG16	8.02	19.5	46.5

In Table 4.4, the average time cost for an image of each network is listed. The fastest learning belongs to AlexNet with 4.88 milliseconds. The slowest learning time belongs to the VGG16 network with 8.02 milliseconds. The same sequence is seen in the test process, which is the classification process of the trained network. Classification of synthetic images is faster than the classification of real images. The reason for this speed difference is because real images have more color gamut.

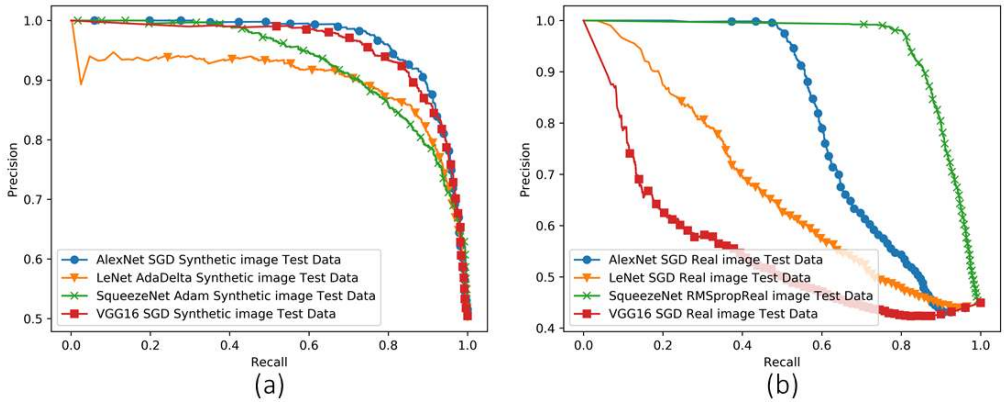

Figure 4.5 Precision and recall graph of experiment 2 synthetic and real image test results.

The results of the tests performed with the synthetic and real image test data of Experiment2 are given in the table. The precision and recall graphs of the most successful combinations of the nets shown in Table 4.3 is shown in Figure 4.5. The graph labeled as (a) in the figure is the precision and recall graph of the test data set consisting of synthetic images. The graph labeled as (b) in the figure is the precision and recall graph of the test data set consisting of real images. Considering the ideal precision and recall graph, the experiment 2 test result precision, and recall graphs show the success of the training of the network. All the networks shown in Figure 4.5 (a) presented a near-ideal graph in the classification process. The graph shown in Figure 4.5 (b) shows the successful results of AlexNet and squeezeNet networks.

4.2.3. Experiment 3 Results

In the experiment, the effect of synthetic data training of a layer applying a threshold filter on the image on real data has been measured. Experiment 3 consists of the training and test results of the network model created by adding a layer that converts the image pixel values to 1 or 0 according to the threshold value at the entry point of known deep learning-based AI models. The feature extraction layer, which uses the thresholding method, is used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.6, the methodology of experiment 3 is shown schematically. Synthetic images were transformed by layering before entering the AI training model. In addition, the Trained AI model is directly tested with synthetic and real images.

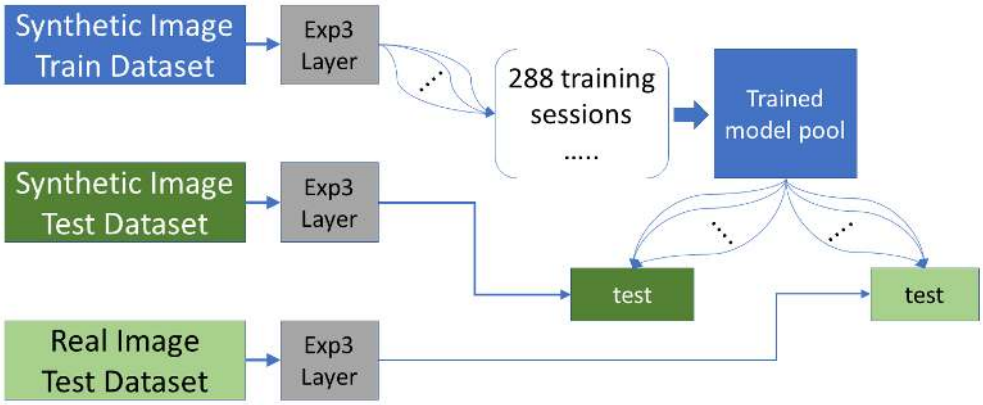

Figure 4.6 Experiment 3 train and test scenario schematic representation.

The experimental results trained and tested in accordance with the schematic representation are listed in Table 4.5. Since testing is performed with each combination of multiple Deep learning-based AI models, optimizers, dropout values, batch size values , and learning rate values, the table is filled for all combination values.

Table 4.5 Exp 3: F-score, accuracy (Acc), and AUC values of the test results.

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
1	AlexNet, Adam, 32, 0.001	73	50	65	64.6	**60**	**62**	**61**	**61.6**
2	AlexNet, SGD, 32, 0.1	**75**	**54**	**68**	**67.5**	54	63	59	58.6
3	LeNet, Adam, 128, 0.01	<55				61	54	58	59.1
4	SquuezeNet, SGD, 32, 0.001					61	56	59	59.9

The results of Experiment 3 are listed in Table 4.5. As a result of the experiment, the models subjected to the classification test process with synthetic and real images gave successful results. Combination number 2 provided a 68% success rate in the classification test process of synthetic images. SGD optimizer, 32 batch size values ,

and 0.001 learning rate values have been used in the AlexNet network, which was 68% successful on synthetic images. In the real image test process, the 1st test was the most successful combination with a rate of 61%. Combining the number 1, Adam optimizer, with 32 batch size values, the AlexNet network achieved 65% success in classifying both synthetic and real images. The proposed model was partially successful in only 4 combinations. This achievement is not remarkable achievement.

The threshold layer is explained in detail in the method section. The purpose of using the layer is to minimize the difference between real and synthetic images. In this way, it is aimed that the AI model can be trained with synthetic images to recognize real images in the real world. The test results of Experiment 3 showed that the added layer did not achieve remarkable success. Adding the layer to the input of the network has the disadvantage of prolonging the training and testing time. The layer's training, validation, synthetic test, and real test are applied to every image in the image dataset. The time taken to transform an average image is 9.81 milliseconds.

Table 4.6 Experiment 3, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	8.19	10.5	23.5
LeNet	8.06	15	30.5
SquuezeNet	9.14	10	22.5
VGG16	12.8	15.5	31.5

Table 4.6 lists the average time cost for an image of each network. The fastest learning belongs to LeNet with 8.06 milliseconds. The slowest learning time belongs to the VGG16 network with 12.8 milliseconds. The same time difference ratio is seen in the test process, which is the classification process of the trained network. Classification times of synthetic images vary between 10 and 15.5 milliseconds according to networks, while classification times of real images vary between 22.5 and 31.5 milliseconds. Classification of synthetic images is faster than the classification of real images. The reason for this speed difference is because real images have more color gamut.

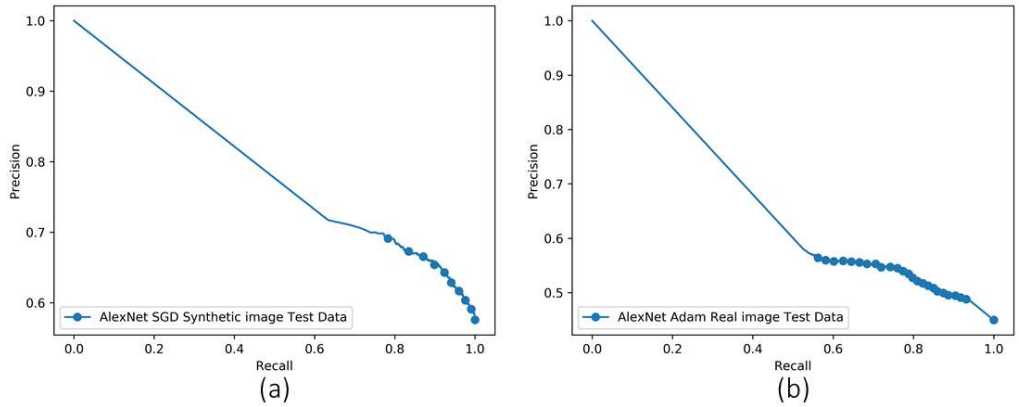

Figure 4.7 Precision and recall graph of experiment 3 synthetic and real image test results.

The results of the tests performed with the synthetic and real image test data of Experiment3 are given in the table. The precision and recall graphs of the most successful combinations of the nets shown in Table 4.5 is shown in Figure 4.7. The graph labeled as (a) in the figure is the precision and recall graph of the test data set consisting of synthetic images. The graph labeled as (b) in the figure is the precision and recall graph of the test data set consisting of real images. Experiment 3 test results are unsuccessful considering the ideal precision and recall graph.

4.2.4. Experiment 4 Results

In the experiment, the effect of synthetic data training of a layer applying the Canny edge detection filter on the image on the real data has been measured. Experiment 4 consists of the training and test results of the network model created by adding an edge detection algorithm layer that reveals the edges of the image to the entry point of known deep learning-based AI models. A feature extraction layer, which uses the canny edge detection method, is used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.8, the methodology of experiment 4 is shown schematically. Synthetic images have been transformed by layering before entering the AI training model. In addition, the Trained AI model is directly tested with synthetic and real images.

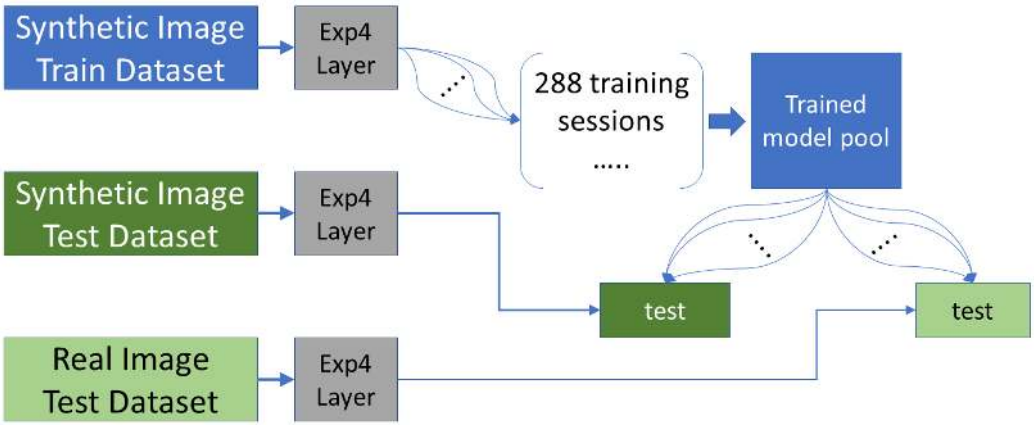

Figure 4.8 Experiment 4 train and test scenario schematic representation.

The experimental results trained and tested in accordance with the schematic representation are listed in Table 4.7. Since testing is performed with each combination of multiple Deep learning-based AI models, optimizers, dropout values, batch size values , and learning rate values, the table is filled for all combination values.

Table 4.7 Exp 4: F-score, accuracy (Acc), and AUC values of the test results.

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
1	AlexNet, AdaDelta, 128, 0.001	<55				75	71	73	74.8
2	AlexNet, AdaDelta, 128, 0.01	71	34	60	59.7	72	75	73	73.5
3	AlexNet, AdaDelta, 128, 0.1	70	30	58	57.7	68	72	70	70.3
4	AlexNet, AdaDelta, 32, 0.001	<55				69	65	67	68.7
5	AlexNet, AdaDelta, 32, 0.01					72	71	71	72.3
6	AlexNet, AdaDelta, 32, 0.1					67	69	68	68.2

Table 4.7(cont.) Exp 4:F-score, accuracy (Acc), and AUC values of the test results

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
7	AlexNet, Adam, 128, 0.001	<55				74	63	70	72
8	AlexNet, SGD, 128, 0.001					76	74	75	76
9	AlexNet, SGD, 128, 0.01					**75**	**77**	**76**	**76.3**
10	AlexNet, SGD, 128, 0.1	64	59	62	61.9	56	73	67	65.2
11	AlexNet, SGD, 32, 0.001	<55				74	76	75	74.9
12	AlexNet, SGD, 32, 0.01					74	76	75	75.2
13	AlexNet, SGD, 32, 0.1	**72**	**42**	**62**	**62.4**	67	74	71	70
14	LeNet, AdaDelta, 128, 0.001	<55				**79**	**80**	**79**	**79.6**
15	LeNet, AdaDelta, 128, 0.01	<55				78	77	78	78.5
16	LeNet, AdaDelta, 128, 0.1					77	77	77	78
17	LeNet, AdaDelta, 32, 0.001					77	76	77	77
18	LeNet, AdaDelta, 32, 0.01					78	78	78	78
19	LeNet, AdaDelta, 32, 0.1					77	77	77	77

Table 4.7(cont.) Exp 4:F-score, accuracy (Acc), and AUC values of the test results

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
20	LeNet, Adam, 128, 0.001	<55				77	77	77	78
21	LeNet, SGD, 128, 0.001					74	80	78	77.2
22	LeNet, SGD, 128, 0.01					72	79	76	75.1
23	LeNet, SGD, 32, 0.001					73	80	77	76.3
24	SqueezeNet, AdaDelta, 32, 0.1					**60**	**76**	**70**	**68.5**
25	SqueezeNet, Adam, 32, 0.1	68	46	60	59.6	70	62	66	68.3
26	SqueezeNet, RMSprop, 128, 0.01	77	64	72	72	60	74	69	67.1
27	SqueezeNet, RMSprop, 32, 0.01	**69**	**75**	**72**	**72**	<55			
28	SqueezeNet, SGD, 128, 0.01	69	65	67	67.1				
29	VGG16, AdaDelta, 32, 0.1	**72**	**41**	**62**	**62**	**58**	**60**	**59**	**59.4**

The results of Experiment 4 are listed in Table 4.7. As a result of the experiment, the models subjected to the classification test process with synthetic and real images gave successful results. 29 successful results are listed in Table 4.7. In the classification test process of synthetic images, the combination number 27 provided a 72% success rate. RMSprop optimizer, 32 batch size values , and 0.01 learning rate values have been used in the network SqueezeNet network, which was 72%

successful on synthetic images. In the real image test process, test number 14 was the most successful combination with a rate of 79%. The AdaDelta optimizer, which makes up the number 14 combination, has achieved success with the 128-batch size value of the LeNet network. The proposed model was only partially successful in 14 combinations. In the proposed model, the most successful combination success rate is 79%.

The Canny Edge detection layer is explained in detail in the method section. The purpose of using the layer is to minimize the difference between real and synthetic images. In this way, it is aimed that the AI model can be trained with synthetic images to recognize real images in the real world. Adding the layer to the input of the network has the disadvantage of prolonging the training and testing time. The layer's training, validation, synthetic test, and real test are applied to every image in the image dataset. The time taken to transform an average image is 7.85 milliseconds.

Table 4.8 Experiment 4, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each	Average Real Test time for each image (ms)
AlexNet	7.89	10	23
LeNet	8.50	14.5	30.5
SquuezeNet	8.10	10	22
VGG16	12.86	15.5	31.5

Table 4.8 lists the average time cost for an image of each network. The fastest learning belongs to AlexNet with 7.89 milliseconds. The slowest learning time belongs to the VGG16 network with 12.86 milliseconds. The same time difference ratio is seen in the test process, which is the classification process of the trained network. Classification times of synthetic images vary between 10 and 15.5 milliseconds according to networks, while classification times of real images vary between 22.5 and 31.5 milliseconds. Classification of synthetic images is faster than the classification of real images. The reason for this speed difference is because real images have more color gamut.

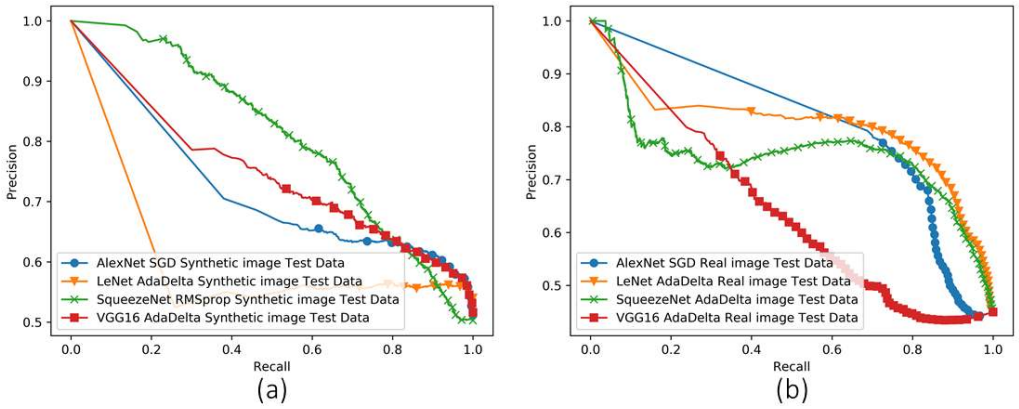

Figure 4.9 Precision and recall graph of experiment 4 synthetic and real image test results.

The results of the tests performed with the synthetic and real image test data of Experiment 4 are given in the table. The precision and recall graphs of the most successful combinations of the nets shown in Table 4.8 is shown in Figure 4.9. The graph labeled as (a) in the figure is the precision and recall graph of the test data set consisting of synthetic images. The graph labeled as (b) in the figure is the precision and recall graph of the test data set consisting of real images. Experiment results are not close to the ideal precision-recall graph.

4.2.5. Experiment 5 Results

In the experiment, the effect of synthetic data training of a layer that applies median blur and Bilateral Filter on the image, respectively, on the real data was measured. Experiment 5 consists of the training and test results of the network model created by adding a layer that transforms the style of the image into cartoon style to the entry point of known deep learning-based AI models.

Feature extraction layer using cartoon transformation method has been used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.10, the methodology of experiment 5 is shown schematically. Synthetic images were transformed by layering before entering the AI training model. In addition, the Trained AI model is directly tested with synthetic and real images.

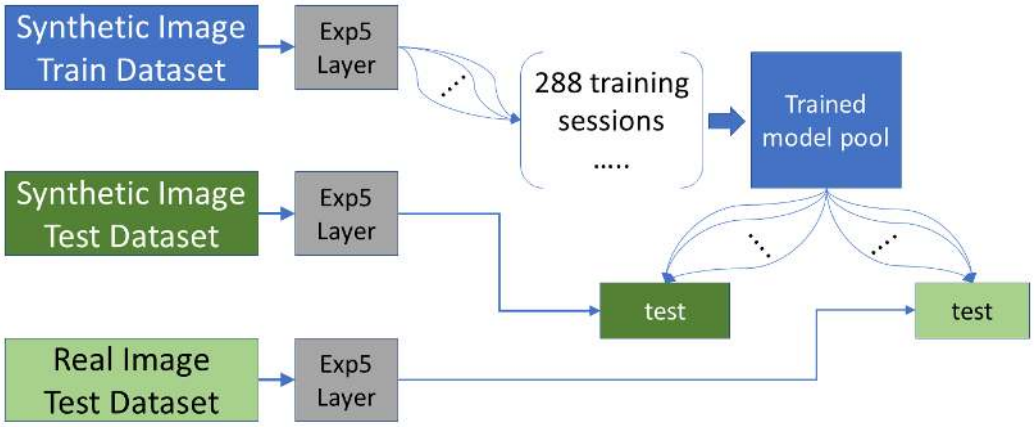

Figure 4.10 Experiment 5 train and test scenario schematic representation.

The test results table, which was trained and tested in accordance with the schematic representation, was not made. All the networks trained in Experiment 5 failed the image classification process. Accuracy values of all networks are below 55%. In addition, the average time cost for each image of the layer used in the experiment is 47.12 milliseconds.

Table 4.9 Experiment 5, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	7.83	13	32.5
LeNet	8.50	17	40
SquuezeNet	8.10	12.5	32.5
VGG16	13.58	17.5	41

The time spent for each image in Experiment 5 training and testing is listed in table 4.9. During the training phase, time was spent between 7.83ms and 13.58ms for each image. Between 12.5ms and 17.5ms was spent in testing the synthetic images. The time between 32.5ms and 41ms was spent testing the real images. Depending on the complexity of the network used, the time spent for training and testing varies.

4.2.6. Experiment 6 Results

In the experiment, the effect of synthetic data training of a layer that applies a Bilateral Filter on the image on the real data was measured. Experiment 6 consists of

the training and test results of the network model created by adding a layer that blurs the image of the image to the entry point of known deep learning-based AI models.

The feature extraction layer, which uses the image blurring method, is used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.11, the methodology of experiment 6 is shown schematically. Synthetic images have been transformed by layering before entering the AI training model. In addition, the Trained AI model is directly tested with synthetic and real images.

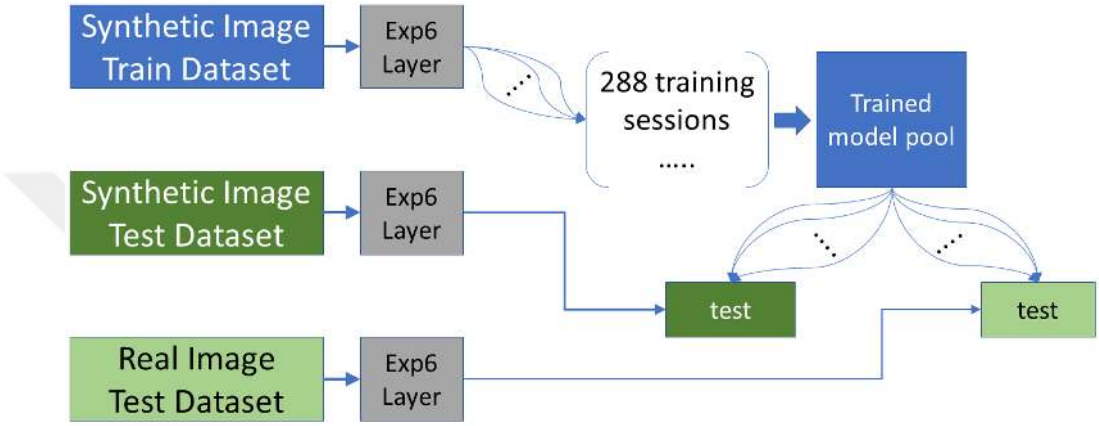

Figure 4.11 Experiment 6 train and test scenario schematic representation.

The test results table, which was trained and tested in accordance with the schematic representation, was not made. All the networks trained in Experiment 5 failed the image classification process. Accuracy values of all networks are below 55%. In addition, the average time cost for each image of the layer used in the experiment is 21.59 milliseconds.

Table 4.10 Experiment 6, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	7.87	10	23.5
LeNet	7.86	14	30.5
SquuezeNet	8.12	9.5	22.5
VGG16	14.36	15	32

The time spent for each image in Experiment 6 training and testing is listed in table 4.10. During the training phase, time was spent between 7.86ms and 14.36ms for each image. Between 10ms and 15ms was spent in testing the synthetic images. The time between 22.5ms and 32ms was spent testing the real images. Depending on the complexity of the network used, the time spent for training and testing varies.

4.2.7. Experiment 7 Results

In the experiment, the effect of synthetic data training of layer segmenting on the image on real data has been measured. Experiment 7 consists of the training and test results of the network model created by adding a layer that segments the parts of the objects in the image to the entry point of known deep learning-based AI models.

The feature extraction layer, which uses the image segmentation method, is used to reduce the difference between the synthetic image and the real image. In the model shown in Figure 4.12, the methodology of experiment 7 is shown schematically. Synthetic images have been transformed by layering before entering the AI training model. In addition, the Trained AI model is directly tested with synthetic and real images.

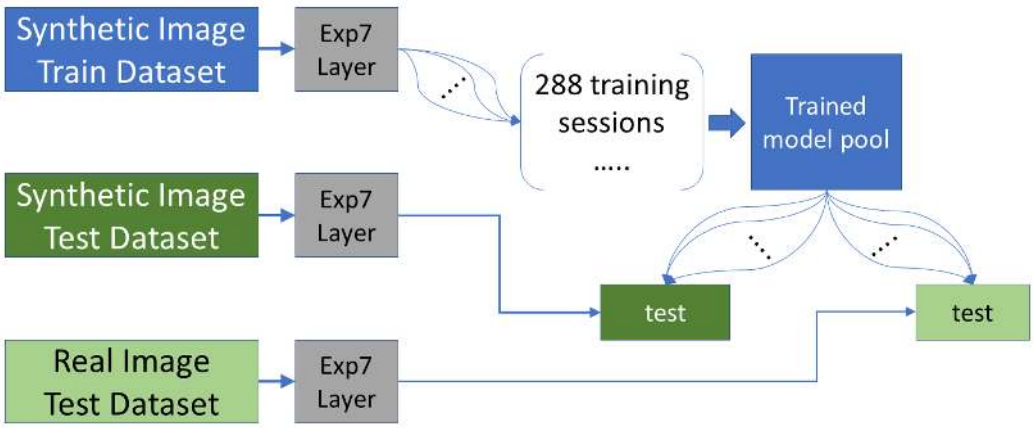

Figure 4.12 Experiment 7 train and test scenario schematic representation.

The experimental results trained and tested in accordance with the schematic representation are listed in Table 4.11. Since testing is performed with each combination of multiple Deep learning-based AI models, optimizers, dropout values, batch size values , and learning rate values, the table is filled for all combination values.

Table 4.11 Exp 7: F-score, accuracy (Acc), and AUC values of the test results.

Id	Network, Optimizer, Batch Size, Learning Rate	Synthetic Image Test				Real Image Test			
		F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)	F-Score Bird (%)	F-Score RW-UAV (%)	Acc (%)	AUC (%)
1	AlexNet, AdaDelta, 128, 0.1	<55				52	76	67	54.8
2	AlexNet, AdaDelta, 32, 0.1					**60**	**75**	**69**	**67.4**
3	AlexNet, SGD, 128, 0.1					49	78	69	65.8
4	AlexNet, SGD, 32, 0.1					43	77	67	63.2
5	LeNet, RMSprop, 32, 0.01					**59**	**65**	**62**	**61.8**
6	SqueezeNet, AdaDelta, 32, 0.1					**43**	**75**	**66**	**62.2**
7	SqueezeNet, RMSprop,128, 0.001					62	56	59	60.5

In Experiment 7, successful test results of the trained networks are listed. At the end of the experiment, all results in the synthetic image test are below the 55% success value. Therefore, the synthetic image test is unsuccessful. As a result of the real image test, 7 combinations showed a success of over 55%. Combination number 2 is the most successful with 69%. Successful Combination AlexNet network is achieved by using AdaDelta optimizer, 128 batch size value, and 0.1 learning value. In general, the experiment that succeeded in very few combinations was considered unsuccessful.

The image segmentation layer is explained in detail in the method section. The purpose of using the layer is to minimize the difference between real and synthetic images. In this way, it is aimed that the AI model can be trained with synthetic

images to recognize real images in the real world. Adding the layer to the input of the network has the disadvantage of prolonging the training and testing time. The layer's training, validation, synthetic test, and real test are applied to every image in the image dataset. The time taken to transform an average image is 3993 milliseconds. The time spent in the use of the layer is almost 1000 times slower compared to other experiments. This speed difference is a significant disadvantage.

Table 4.12 Experiment 7, time consumption table.

Network	Average Training time/for each image (ms)	Average Synthetic Test Time for each image (ms)	Average Real Test time for each image (ms)
AlexNet	7.82	9.5	22
LeNet	7.77	13.5	29
SquuezeNet	7.80	9	20.5
VGG16	11.08	14.5	30

The time spent for each image in Experiment 7 training and testing is listed in table 4.12. During the training phase, time was spent between 7.77ms and 11.08ms for each image. Between 9ms and 14.5ms was spent in testing the synthetic images. The time between 20.5ms and 30ms was spent testing the real images. Depending on the complexity of the network used, the time spent for training and testing varies.

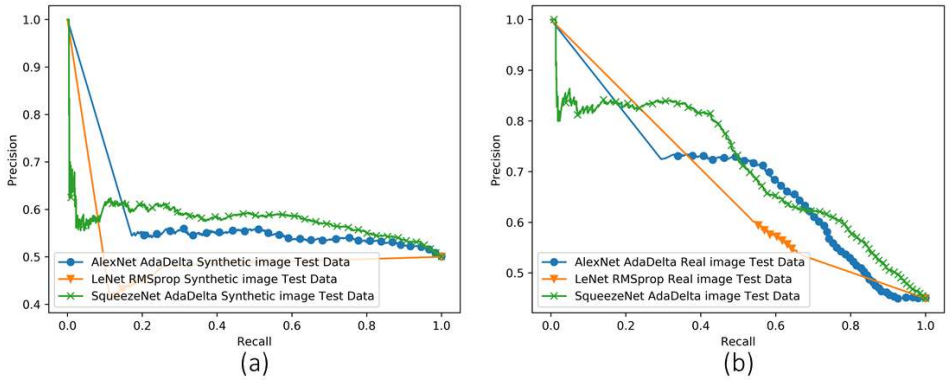

Figure 4.13 Precision and recall graph of experiment 7 synthetic and real image test results.

The results of the tests performed with the synthetic and real image test data of Experiment4 are given in the table. The precision and recall graphs of the most successful combinations of the nets shown in Table 4.11 is shown in Figure 4.13.

The graph labeled as (a) in the figure is the precision and recall graph of the test data set consisting of synthetic images. The graph labeled as (b) in the figure is the precision and recall graph of the test data set consisting of real images. Experiment results are unsuccessful considering the ideal precision recall graph.

4.3. Comparation of Experimental Results

The 7 experiments carried out were named EXP1 – EXP7, respectively. The experiments have been compared in terms of training times of deep learning-based networks, image transformation times of the feature extraction layer of the experiments, time to classify the image in the test process of the trained networks, time cost, and measuring the accuracy of the network. Measured training, test times have been calculated based on the average time spent on an image. The trained network in each experiment was tested with synthetic and real images. The accuracy value measured because of the test process has been compared between the experiments and 4 different networks used in each experiment. All-time measurements were made on a computer system using a GPU computing system (GeForce GTX 1080ti, Nvidia Corp., Santa Clara, CA, USA), 128 GB RAM, I7 processors, and the TensorFlow-based and Keras library.

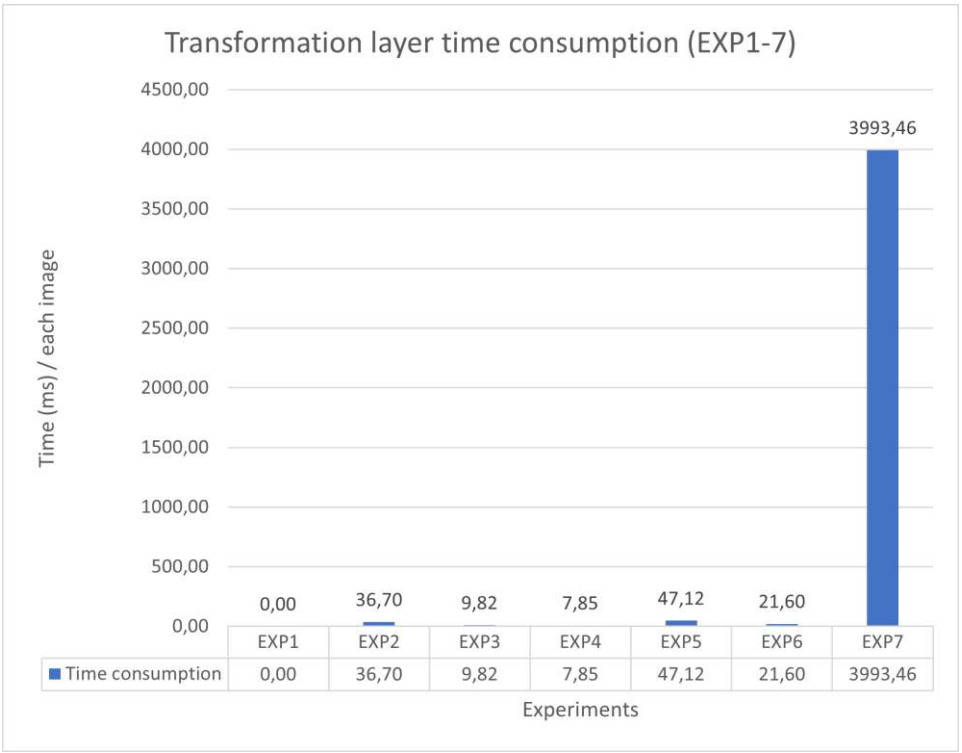

Figure 4.14 Transformation layer time consumption (EXP1-EXP7)

In Figure 4.14, the average time spent by the feature extraction layers for an image, which determines the characteristics of the experiments, is shown in milliseconds. Since no feature extraction layer is used in Experiment 1, the feature extraction layer usage time is 0 ms. The time spent for the feature extraction layer used in the studies from Experiment 2 to Experiment 7 is 36.7 ms, 9.82 ms, 7.85 ms, 47.12 ms, 21.60 ms, 3993.46 ms, respectively. The time cost of the Experiment 7 feature extraction layer is significantly larger than the layer of other experiments. Experiments 1 – 6, which could not be examined due to the result of experiment 7 in the graph, are shown separately in the next graph.

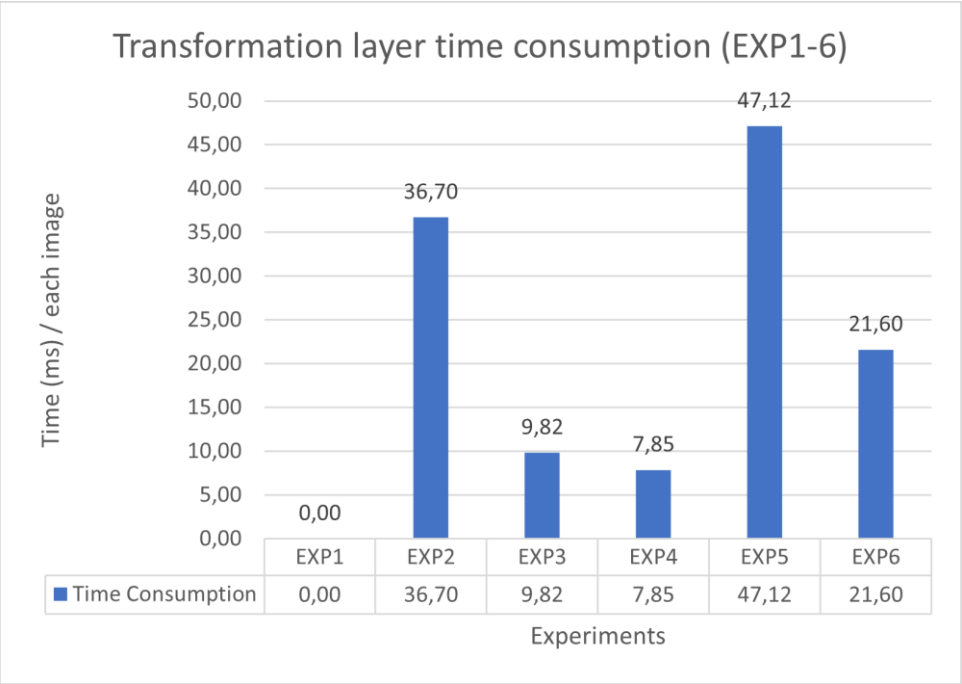

Figure 4.15 Transformation layer time consumption (EXP1-EXP6)

Figure 4.15 shows the image conversion times of the feature extraction layer used in the experiments. When the previous graph and the graph in Figure 4.16 are examined, the longest and time-consuming feature extraction layer is the image segmentation layer used in exp 7. The second most time-consuming feature extraction layer is the median blur and Bilateral Filter layer used in exp. The third most time-consuming feature extraction layer is the CDNTS layer we pioneered. The time cost of the proposed CDNTS layer seems high, but it is reasonable compared to the time required to train a deep learning-based network.

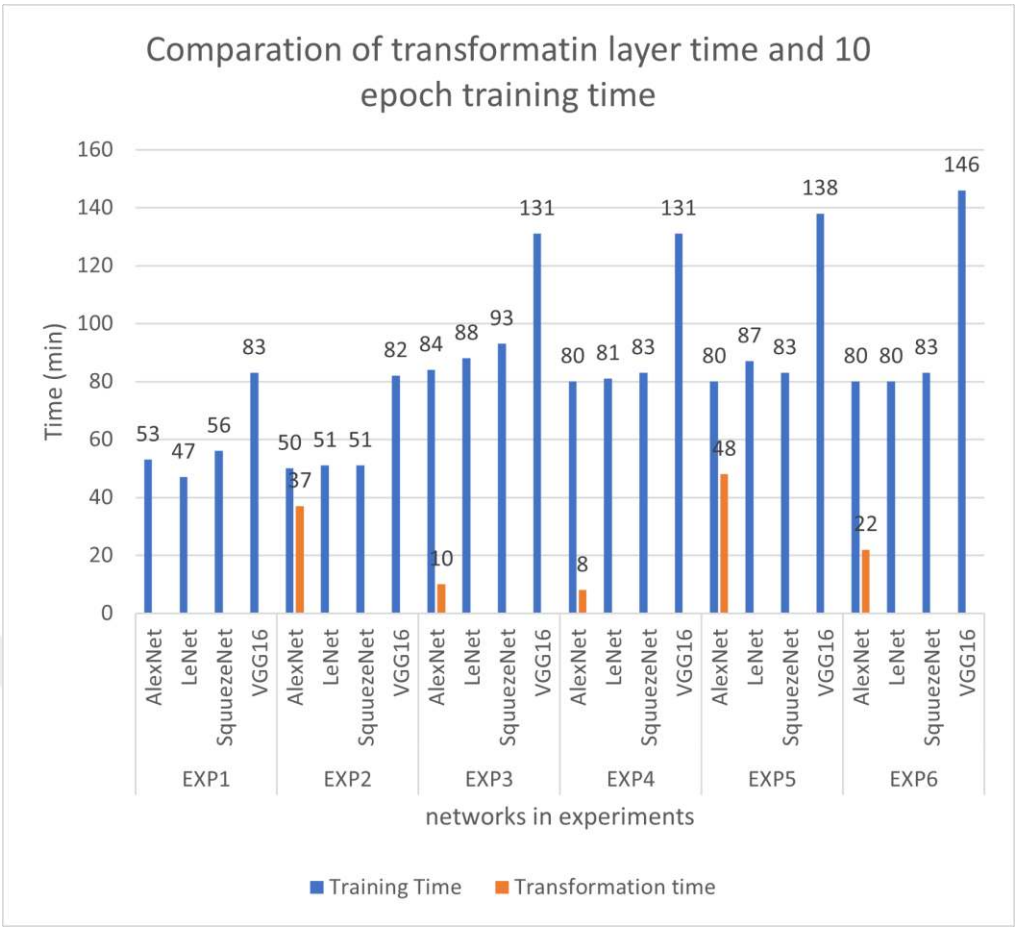

Figure 4.16 Comparation graph of transformation layer time of experiments (Exp1 Exp6)

The graph shown in Figure 4.16 compares the time required to train the networks with the time required for the feature extraction layer to transform images. In Experiment 1, the training time for a deep learning-based network consisting of 10 epochs, for 61120 pieces of 224x224 px RGB image dataset, varies between 46 minutes and 82 minutes. In Experiment 2, the training period changes in the same way. The time cost of the CDNTS layer in Experiment 2 is 36 minutes. The time cost of the CDNTS network in Experiment 2 was spent only once and was used in 4 networks. When we calculate the combinations of 4 networks and the elapsed training time, the CDNTS layer covers 36 minutes in training that takes an average of 8 days for each network.

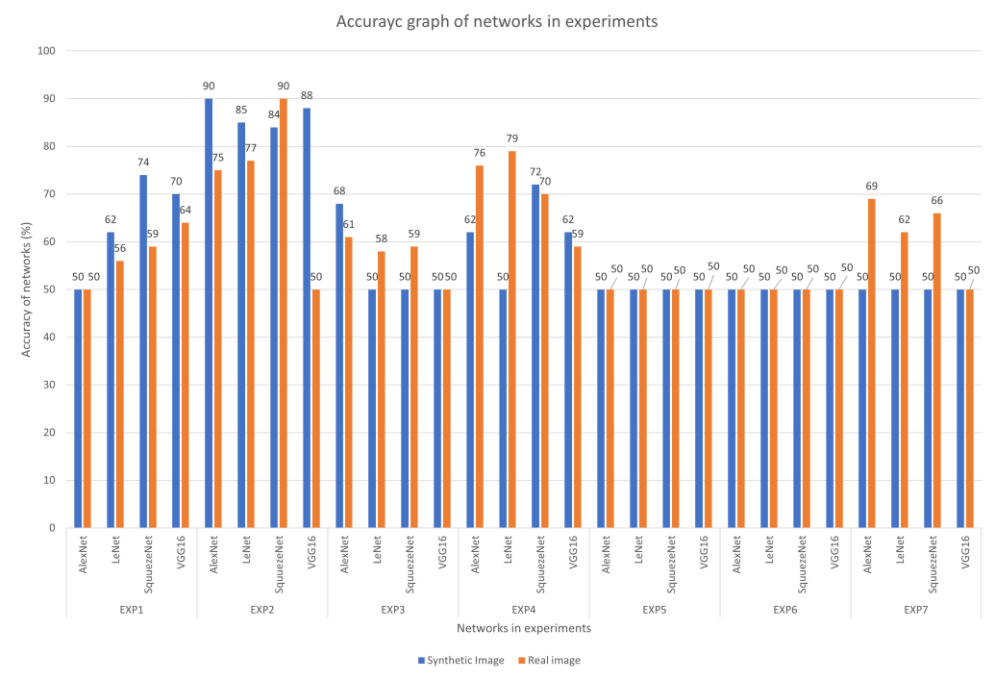

Figure 4.17 Comparation of accuracy value of experiments (Exp1-Exp7)

For each experiment in Figure 4.17, the accuracy values of test results of AlexNet, LeNet, SqueezeNet, and VGG16 networks have been examined. Considering the successes above 50%, it is seen that 4 different networks show success differences according to the experiments. The AlexNet network fails in experiment 2 and experiments 4 external tests. AlexNet's most successful experiment is experiment number 2. It was successful with 90% accuracy in the classification of synthetic images and 75% accuracy in the classification of real images. The LeNet network gave successful results in experiment 1, experiment 2, experiment 3, experiment 4, and experiment 7. However, experiment 2 showed the most successful result. The LeNet network was successful with 85% accuracy in the classification of synthetic images and 77% accuracy in the classification of real images. The SqueezeNet network was successful in experiments 1, 2, 3, 4, and 7. The most successful experiment of the network is experiment number 2. The network showed success with 90% accuracy in the classification of real images and 84% accuracy in the classification of synthetic images. The VGG16 network was successful in experiments 1,2 and 3. The VGG16 network has been able to classify synthetic images with an accuracy of 88% in experiment 2.

The common experiment in which all networks succeed is experiment number 2. In Experiment 2, the CDNTS layer, which is the feature extraction layer, has a positive

effect on the classification process. The main purpose of the study; The aim is to enable the AI model, which is trained using synthetically produced images, to classify real images. The graph shown in figure 4.18 provides the purpose of the study with the CDNTS layer used in experiment 2 and the squeezeNet network used. The CDNTS layer and squeezeNet network can be trained using synthetic images to classify real images with 90% accuracy. The proposed CDNTS layer shows that the AI model trained with synthetic images can be used in the classification of real images. In this respect, it is an advantageous method to use the CDNTS layer in AI applications. Besides the advantage of the CDNTS layer, it has a disadvantage such as time cost.

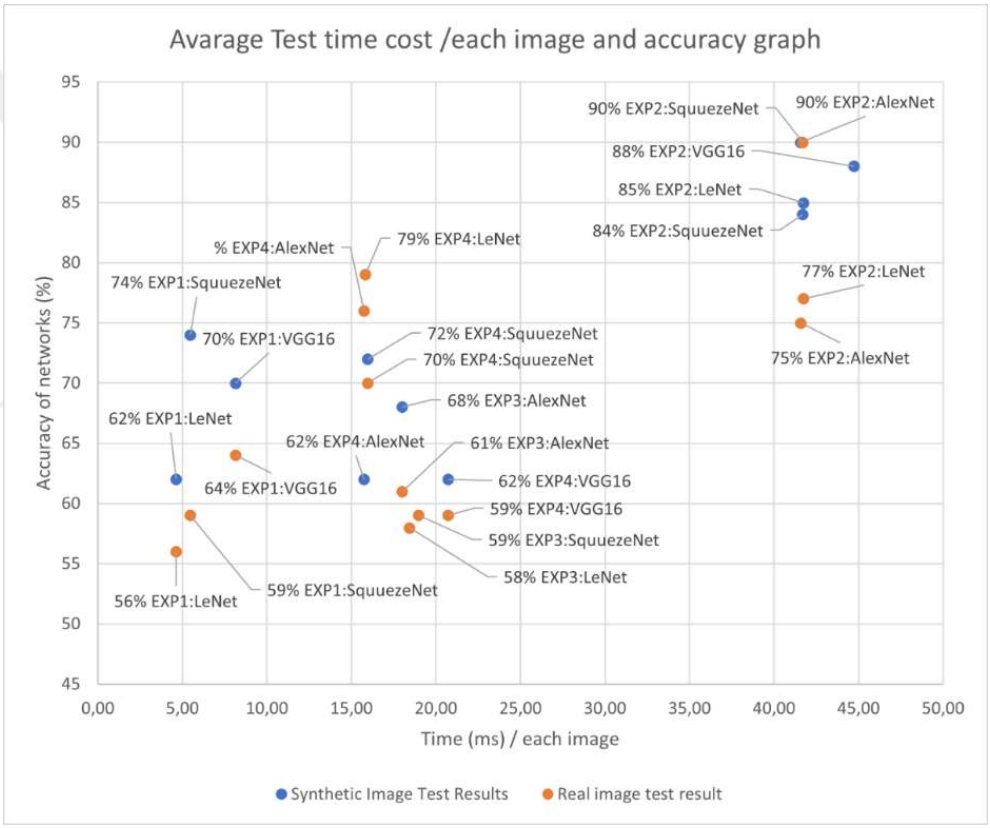

Figure 4.18 Comparation of accuracy and speed of experiments test results (Exp1 exp6)

The graph of time spent to classify an image and classification accuracy rate is shown in Figure 4.18. Experiment 7 results are not shown in Figure 4.18 because the time cost of experiment 7 is 130 times higher than the other experiments and this reduces the readability of the graph. Results with Experiment 7 are reported in Figure 4.14 and Figure 4.17. In the graph, it is seen that the results of experiment 2, the

CDNTS layer, are in the upper right corner. Experiment 2 was generally successful for all networks. On the other hand, the fact that it is located on the right side of the graphic area means that the time cost is high. The CDNTS layer used in Experiment 2 enabled the AI model trained with synthetic images to be used in classifying real images with 90% accuracy. CDNTS layer has increased the performance of networks. The CDNTS layer, which increased the performance, slowed down the classification process. The average time cost of the CDNTS layer and the classification process of the networks is 41.68ms. In a 24 frame per second (FPS) camera image transmission, there is a 41.6ms time difference between each frame. The CDNTS layer and the time cost of networks 41.68ms is enough to process the image streaming from a camera momentarily without delay. The time cost of the CDNTS layer can be reduced by increasing the hardware capabilities of the AI computer.

CHAPTER V

CONCLUSION AND FUTURE WORK

The success factor of deep learning-based AI classification networks depends on the size and diversity of the dataset required for the training validation and testing processes. Gathering the image dataset needed to solve problems such as RW-UAV and bird classification is a labor-intensive task. Although there are ready-made image datasets to create a dataset, they may not be enough in terms of diversity and number. In this case, it may be necessary to collect the data set from the internet and photograph the objects manually. This study focused on the ability to train AI-based image classification problems with limited datasets using synthetic datasets and the trained AI model's ability to classify real images in real life. Synthetic image production is the reproduction of real-world models created using game engines in a virtual environment by photographing. The research question that forms the basis of the study; Can an AI model trained using synthetic data generated in a game engine environment be used to classify real images in real-life conditions?

In this study, two different cities have been designed in the old-style city design and the modern style city design created in the game engine environment. In addition to this, a green box studio has been designed where synthetic data set duplication is made by photographing 3D RW-UAV and bird models from different angles. In the designed studio, 61120 RW-UAV and bird images with a green background have been created. For the testing of the trained AI model, 2500 RW-UAV and bird images have been created using city designs. 21129 real RW-UAV and bird photos have been downloaded from the internet and the photos showing 80% similarity have been deleted. As a result of this process, the real image test process resulted in 3385 real RW-UAV and bird photos.

The image dataset prepared for training, validation, synthetic image testing, and real image testing has been trained in seven different experiments with 288 different combinations of 4 different deep learning-based networks, each with different training parameters. When the number of experiments, the 4 different networks used in the experiment, and the combinations obtained by changing the parameters of these networks are calculated, the total number of trainings is 2016. The time spent on this training is 1848 hours. This period corresponds to 77 days in days. Experiment 1 consists of training and testing with standard deep learning-based networks. In experiment 2, the CDNTS feature extraction layer recommended by us was added for the introduction of deep learning-based networks. In the other 5 experiments, the known feature extraction layers have been added, and the results were reported. The feature extraction layers used in these 5 experiments are, respectively, thresholding, canny edge detection, median blur, bilateral filter, and image segmentation layers. They transform the image structure of these layers by changing them. This method used was applied to eliminate the difference between real images and synthetic images. The purpose of the variety of experiments is to find an answer to the question of how advantageous are feature extraction layers for AI models trained with synthetic data.

When the results of the experiments are examined, the most successful result on synthetic test data and real test data belongs to experiment number 2. The success rate in the classification of synthetic data in experiment number 2 using the CDNTS layer, suggested by us, is between 84% and 90%. The highest classification success rate of synthetic test data is the AlexNet network with 88%. The success rate in classifying real test data in the same experiment is between 75% and 90%. The network that most successfully classifies real data is the SqueezeNet network with 90%. Experiment 2 test results are generally 75% to 90% successful for all networks. In the 1st experiment where no feature extraction layer was used, the success rates remained between 56% and 74%. In general, the tests of the networks were unsuccessful. This showed that the CDNTS layer is a layer that directly increases the success of the network. Thanks to the CDNTS layer, an AI model trained with synthetic images was 90% successful in classifying real images. When the other test results were examined, the 3rd experiment was successful at a rate of 58% to 68%. Experiment number 4 was successful at a rate of 59% to 79%. Experiment 5 and

experiment 6 were completely unsuccessful. Experiment 7 was successful 62% to 69%.

Experiment 1 model, which does not contain any feature extraction layer, is faster than other experimental models. The order of experiments from fastest to slowest is as follows. Experiment 1 is fastest followed by experiment 4, experiment 3, experiment 2, and experiment 7, respectively. Experiments 5 and 6 are not ranked as they were completely unsuccessful. CDNTS layer enabled to increase the performance in 4 different deep learning-based networks. In addition to this advantage, there is a time cost disadvantage. Experiment 2 with the CDNTS layer increased the classification process of each image to 41.68 ms. In a 24 frame per second (FPS) camera image transmission, there is a time difference of 41.6 ms between each frame. The CDNTS layer and the time cost of networks 41.68 ms are enough to process the image streaming from a camera momentarily without delay. The time cost of the CDNTS layer can be reduced by increasing the hardware capabilities of the AI computer.

Table 5.1 Accuracy results comparison from literature.

Id	**Name of study**	**Training data -> testing data type**	**Accuracy of model**
1	Semantic Segmentation for Autonomous Car [54]	Sythetic and Real data mixed -> Real data	94%
2	Smoke Detection [55]	Synthetic and Real data mixed -> real data	99.71%
3	Joint Detection [56]	Synthetic and real data mixed -> Real data	80%
4	Dental Occlusion Classification [57]	Synthetic data -> Real data	78.95%
5	Anomaly Detection in Crowd Scenes [59]	Synthetic data -> Real data	92%
6	**Real UAV-Bird Image Classification [83]**	Synthetic data -> Real data	90%

The accuracy comparison table of studies similar to this study is listed in Table 5.1. In the studies carried out, the training dataset is completely synthetic or consists of a mixture of synthetic and real data. The most successful study listed in Table 5.1 was study number 2, which achieved 99.71% success. Study number 2 training dataset consists of a mixture of synthetic and real data. Models trained with fully synthetic

data are studies 4, 5 and 6. The accuracy of work number 4 is 78.95%. The accuracy of work number 5 is 92%. The accuracy of work number 6, which is the subject of the thesis, is 90%. Work number 6 has a better score than work number 4. Study number 5 has a higher score than study number 6. But work number 5 is sequential image detection, not image classification. Studies are applications with different experimental setups. This comparison table has been made to benefit the positioning of our study with the literature.

Despite the limitations of our application mentioned above, the current study has demonstrated that networks can be trained with synthetic images generated by game engines using 3D models to solve real-image classification problems. The success of classification networks trained with synthetic data in real-image classification has been improved thanks to the CDNTS layer. This research also found that using synthetic images is effective for speeding up the solution of image classification problems with insufficient or no image data for training and improving testing performance. It is planned to develop a more robust classification model for night vision and thermal vision cameras in future research. It also aims to train artificial intelligence using synthetic data generated from only a few 2D images.

REFERENCES

[1] Marti, E., de Miguel, M. A., Garcia F., Perez, J. (2019). A Review of Sensor Technologies for Perception in Automated Driving. *IEEE Intelligent Transportation Systems Magazine*, **11(4)**, 94-108.

[2] Zhu, H., Wei, H., Li, B., Yuan, X., Kehtarnavaz, N. (2020) A Review of Video Object Detection: Datasets, Metrics and Methods. *Applied Sciences,* **10**, 7834.

[3] Yang, T., Li, Z., Zhang, F., Xie, B., Li J., Liu, L. (2019). Panoramic UAV Surveillance and Recycling System Based on Structure-Free Camera Array. *IEEE Access*,**7**,25763-25778.

[4] Prates, P.A., Mendonça, R., Lourenço, A., Marques, F., Matos-Carvalho, J. P., Barata, J., (2018). Vision-based UAV detection and tracking using motion signatures. *IEEE Industrial Cyber-Physical Systems (ICPS)*, 482-487

[5] Wang, A., Zhang, W., Wei, X. (2019). A review on weed detection using ground-based machine vision and image processing techniques. *Computers and Electronics in Agriculture,* **158,** 226-240.

[6] Liakos, K.G., Busato, P., Moshou, D., Pearson, S., Bochtis, D. (2018). Machine Learning in Agriculture: A Review. *Sensors*, **18**, 2674.

[7] Dogan, A.; Birant, D. (2021) Machine learning and data mining in manufacturing. *Expert System with Applications*, **166**, 114060.

[8] Kim, J., Gadsden, S.A., Wilkerson, S.A. (2020) A Comprehensive Survey of Control Strategies for Autonomous Quadrotors. *Canadian Journal of Electrical and Computer Engineering*, **43**, 3–16

[9] Tsouros, D.C., Bibi, S., Sarigiannidis, P.G. (2019) A Review on UAV-Based Applications for Precision Agriculture. *Information*, **10**, 349.

[10] Nguyen H., Nguyen D. (2021) Drone Application in Smart Cities: The General Overview of Security Vulnerabilities and Countermeasures for Data Communication. In: Krishnamurthi R., Nayyar A., Hassanien A.E. (eds)

[11] Development and Future of Internet of Drones (IoD): Insights, Trends and Road Ahead. Studies in Systems, Decision and Control, *Springer*, Cham 332.

[12] Chen, S., Yin, Y., Wang, Z., Gui, F. (2020) Low-altitude protection technology of anti-UAVs based on multisource detection information fusion. *International Journal of Advantage Robotic Syst*ems, **17**.

[13] Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L., Matson, E.T. (2020) Real-Time and Accurate Drone Detection in a Video with a Static Background. *Sensors*, **20**, 3856.

[14] Santos, I., Castro, L., Rodriguez-Fernandez, N., (2021) Torrente-Patino, A.; Carballal, A. Artificial Neural Networks and Deep Learning in the Visual Arts: A review. *Neural Computing Applications*, **33**, 121–157.

[15] Lei Y. Yang B., Jiang X., Jia F., Li N., Nandi A.K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap, *Mechanical Systems and Signal Processing.* **138,** 106587.

[16] Rahman A., Muniyandi R.C. (2018). Review of GPU implementation to process of RNA sequence on cancer. *Informatics in Medicine Unlocked.* **10,** 17-26.

[17] Mittal S., (2019). A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. *Journal of Systems Architecture,* **97,** 1383-7621.

[18] Nalepa J., Marcinkiewicz M., Kawulok M., (2019). Data Augmentation for Brain-Tumor Segmentation: A Review. Front. *Computational Neuroscience* 13-83.

[19] Tieleman, T., Hinton, G., (2012) Lecture 6.5-RMSProp. *COURSERA Neural Network Machine Learning*, **4**, 26–30.

[20] Chollet, F., (2018). *Deep learning with Python*, Manning.

[21] Yao X., Liu Y., (2014) Machine Learning. In: Burke E., Kendall G. (eds) Search Methodologies. *Springer*.

[22] D. Graupe, (2007) Principle of Artificial Neural Networks, 2nd Edition, USAK, TURKEY.

[23] Girshick, R., Donahue, J., Darrell, T., Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, **38(1)**, 142-158.

[24] Kingma, D., Ba, J. (2015) Adam: A method for stochastic optimization. *In Proceedings of the ICLR 2015, San Diego, CA, USA*, 7–9.

[25] Zeiler, M.D. (2012) ADADELTA: An Adaptive Learning Rate Method. *arXiv*.

[26] Sutskever, I., Martens, J., Dahl, G., Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA,* **28**, 1139–1147.

[27] Zurada, J. M. (1992). Introduction to Artificial Neural Systems **8**. St. Paul: West.

[28] Hirasawa, K., Ohbayashi, M., Koga M., Harada, M. (1996). Forward propagation universal learning network. *Proceedings of International Conference on Neural Networks (ICNN'96)*, **1,** 353-358.

[29] Wechsler H. (1992). Neural Networks for Perception Theory of the Backpropagation Neural Network Based on "nonindent" by Robert Hecht-Nielsen., *Academic Press.* 65-93.

[30] Shoham, Y., Perrault, R., Brynjolfsson, E., Clark, J., LeGassick, C. (2017) Artificial Intelligence Index, *aiindex.org,* 1-101.

[31] Buduma, N., Locascio, N., (2017) "Fundamentals of Deep Learning". *OREILL.*

[32] Goertzel, B. (2015). Are there Deep Reasons Underlying the Pathologies of Today's Deep Learning Algorithms. *In International Conference on Artificial General Intelligence*, 70-79.

[33] Craik A., Yongtian He Y., Contreras-Vidal J. (2019). Deep learning for electroencephalogram ({EEG}) classification tasks: a review. *Journal of Neural Engineering.* **16,** 031001.

[34] Ling Z. et al., (2015). Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends. in *IEEE Signal Processing Magazine*, **32**, 35-52.

[35] Popel, M., Tomkova, M., Tomek, J. et al. (2020) Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications* **11,** 4381.

[36] Wang, Y., Deng, W., Liu, Z., Wang, J. (2019) Deep learning-based vehicle detection with synthetic image data. *IET Intelligent Transportation Syst*ems, **13**, 1097–1105

[37] Krizhevsky, A., Sutskever, I., Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *In Proceedings of the NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems, Stateline, NV, USA*.

[38] Simonyan K., Zisserman A., (2015) Very Deep Convolutional Networks For Large-Scale Image Recognition, *ICLR*.

[39] Lecun, Y. Bottou, L., Bengio, Y., Haffner, P., (1998). Gradient-based learning applied to document recognition., *Proceedings of the IEEE*, **86(11)**, 2278 – 2324.

[40] He, K., Zhang, X., Ren, S., Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE transactions on pattern analysis and machine intelligence*, **37(9)**, 1904-1916.

[41] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015) ImageNet Large

Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, **115**, 211–252.

[42] Iandola, F., Han, S., Moskewicz, M., Ashraf, K., Dally, W., Keutzer, K. (2016) SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size. *arXiv*, arXiv:1602.07360v4.

[43] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available online: *tensorflow.org,* 7.08.2021.

[44] Chollet, F. Keras. 2015. Available online: *https://keras.io*, 5.08.2021.

[45] Du, F., Liu, P., Zhao, W., Tang, X. (2020) Correlation-Guided Attention for Corner Detection Based Visual Tracking. *In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 6835–6844.

[46] Bai, Z., Li, Y., Chen, X., Yi, T., Wei, W., Wozniak, M., Damasevicius, R., (2020) Real-Time Video Stitching for Mine Surveillance Using a Hybrid Image Registration Method. *Electronics*, **9**, 1336.

[47] Li, J., Yan, D., Luan, K., Li, Z., Liang, H. (2020) Deep Learning-Based Bird's Nest Detection on Transmission Lines Using UAV Imagery. *Applied Science*, **10**, 6147.

[48] Kilimci Z.H., Akyokus S., (2018) Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification. *Complexity* 1-20.

[49] Aydoğan M., Karci A., (2020). Improving the accuracy using pre-trained word embeddings on deep neural networks for Turkish text classification. *Physica A: Statistical Mechanics and Its Applications,* **1**, 541.

[50] Duo, J., Zhao, L., (2021). An Asynchronous Real-Time Corner Extraction and Tracking Algorithm for Event Camera. *Sensors*, **21**, 1475.

[51] Krizhevsky, A. (2009) Learning Multiple Layers of Features from Tiny Images CIFAR-10 Dataset. Available online: *https://www.cs.toronto.edu/~kriz/cifar.html,* 5.08.2021.

[52] Gambella, C., Ghaddar, B., Naoum-Sawaya, J., (2021) Optimization problems for machine learning: A survey. *European Journal of Operational Res*earch, **290**, 807–828.

[53] Jaderberg M., Simonyan K., Vedaldi A., Zisserman A. (2014) Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *arXiv.*

[54] Li, Y., Su, H., Qi, C.R., Fish, N., Cohen-Or, D., Guibas, L.J. (2015). Joint Embeddings of Shapes and Images via CNN Image Purification. *ACM Trans. Graph*. **34**.

[55] Ros, G., Sellart, L., Materzynska, J., Vazquez D., Lopez, A.M., (2016). The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3234-3243.

[56] Zhang Q., Lin G., Zhang Y., Xu G., Wang J., (2018). Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images. *8th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE).***211**, 441-446.

[57] Yangyan L. Hao S., Charles Q., Noa F. and Cohen D., Leonidas J. (2015). Joint Embeddings of Shapes and Images via CNN Image Purification. *Association for Computing Machinery.* **34**.

[58] Juneja, M., Singla, R., Saini, S.K. et al. (2020). OCLU-NET for occlusal classification of 3D dental models. *Machine Vision and Applications* 31, 52.

[59] Wang Q., Gao J., Lin W., Yuan Y., (2019). Learning from Synthetic Data for Crowd Counting in the Wild. *arXiv.*

[60] Lin, W., Gao, J., Wang, Q., Li, X., (2021). Learning to detect anomaly events in crowd scenes from synthetic data. *Neurocomputing.* **436,** 248-259.

[61] Hou, L., Chen, H., Zhang, G.K., Wang, X., (2021) Deep Learning-Based Applications for Safety Management in the AEC Industry: *A Review. Applied Science*, **11**, 821.

[62] Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D., (1997) Face recognition: A convolutional neural-network approach. *IEEE Transactions Neural Network and Learning Systems*, **8**, 98–113.

[63] Prijono B., (2018) Convolutional Neural Networks (CNN) Introduction, *indoml.*

[64] LeCun, Y., Bottou, L., Bengio, Y., Haffner P., (1998) Gradient- Based Learning Appled to document Recognation, *Proceedings Of The IEEE.*

[65] Li F., Karpathy A., Johnson J. (2016) "CS231n: Convolutional Neural Networks for Visual Recognition *http://cs231n.stanford.edu,* 08.08.2021.

[66] Baldi P., Sadowski P., Understanding Dropout courses, *Acedemia.*

[67] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research*, 1929-1958.

[68] Sobolev, V.V., Guilemany, J.M. (1999) Flattening of droplets and formation of splats in thermal spraying: A review of recent work—Part 1. J *Thermal Spraying Technology,* **8**, 87–101.

[69] Standford university. (2018). Deep Learning software. *https://rpblic.github.io,* 5.08.2021.

[70] Arenas M.G., Romero G., Mora A.M., Castillo P.A., Merelo J.J. (2012) GPU Parallel Computation in Bioinspired Algorithms: A Review. In: Kołodziej J., Khan S., Burczy´nski T. (eds) Advances in Intelligent Modelling and Simulation. Studies in Computational Intelligence, 422. *Springer*, Berlin, Heidelberg.

[71] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T. (2014, January). Decaf: A Deep Convolutional Activation Feature for

GenericVisual Recognition. *In International conference on machine learning* 647-655.

[72] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O. (2017). Understanding Deep Learning Requires Rethinking Generalization. *arXiv*.

[73] Koirala, A., Walsh, K.B., Wang, Z. (2021). Attempting to Estimate the Unseen—Correction for Occluded Fruit in Tree Fruit Load Estimation by Machine Vision with Deep Learning. *Agronomy* **11**, 347.

[74] Case, E. E., Zelnio A.M., Rigling, B. D., (2008). Low-Cost Acoustic Array for Small UAV Detection and Tracking. *IEEE National Aerospace and Electronics Conference*, 110-113.

[75] Jacek W., Marek, Z., Tadeusz D., Marcin J., Marek Z., Michał, M. (2021). Distinguishing Drones from Birds in a UAV Searching Laser Scanner Based on Echo Depolarization Measurement. *Sensors*. **21**, 16.

[76] Hong, S.J., Han, Y., Kim, S.Y., Lee, A.Y., Kim, G. (2019) Application of Deep-Learning Methods to Bird Detection Using Unmanned Aerial Vehicle Imagery. *Sensors*, **19**, 1651.

[77] Zhao, W., Du, S., (2016). Spectral–Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach. *IEEE Transactions Geoscience Remote Sensing*, **54**, 4544–4554.

[78] Morra, L., Lamberti, F., (2019). Benchmarking unsupervised near-duplicate image detection. *Expert Systems Applications,* **135**, 313–326.

[79] Unity. Unity Asset Store. Available online: *https://assetstore.unity.com***,** 10.08.2021.

[80] Prasad, P.J.R., Survarachakan, S., Khan, Z.A., Lindseth, F., Elle, O.J., Albregtsen, F., Kumar, R.P. (2021) Numerical Evaluation on Parametric Choices Influencing Segmentation Results in Radiology Images—A Multi-Dataset Study. *Electronics*, **10**, 431.

[81] Perez, S.H., Marinho, M.M., Harada, K. (2020). The effects of different levels of realism on the training of CNNs with only synthetic images for the semantic segmentation of robotic instruments in a head phantom. *International Journal Computer Assisted Radiology Surg*ery (IJCARC), **15**, 1257–1265.

[82] Hinton, G.E., Salakhutdinov, R.R., (2016) Reducing the Dimensionality of Data with Neural Networks. *Science*, **313**, 504–507.

[83] Sasaki, Y. The Truth of the f-Measure. 2007. Available online: *https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf,* 5.08.2021.

[84] Öztürk, A.E., Erçelebi, E., (2021) Real UAV-Bird Image Classification Using CNN with a Synthetic Dataset. *Applied Sciences.* **11**, 9.

[85] Qiao, W.B., Créput, J.C., (2021) Component-based 2-/3-dimensional nearest neighbor search based on Elias method to GPU parallel 2D/3D Euclidean Minimum Spanning Tree Problem. *Applied Soft Computing*, **100**, 106928.

[86] Keivani, O., Sinha, K., (2021) Random projection-based auxiliary information can improve tree-based nearest neighbor search. *Information Science*, **546**, 526–542.

[87] Bradski, G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. (2000). Available online: *https://opencv.org***,** 5.08.2021.

[88] Tomasi, S.J. (1994) Good features to track. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June* 593–600.

[89] Tao, Y., Papadias, D., Shen, Q., (2002) Chapter 26—Continuous Nearest Neighbor Search. In *Proceedings of the VLDB '02: Proceedings of the 28th International Conference on Very Large Databases, Kong, China,*; Bernstein, P.A., Ioannidis, Y.E., Ramakrishnan, R., Papadias, D., Eds.; Morgan Kaufmann.

[90] Canny, J., (1986). A Computational Approach to Edge Detection. in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**, 679-698.

[91] Shapiro, L. G., Stockman, G. C, (2001) *Computer Vision*, page 137, 150. Prentice Hall.

[92] Ono, S., Miyata T., Yamada, I., (2014). Cartoon-Texture Image Decomposition Using Blockwise Low-Rank Texture Characterization," in *IEEE Transactions on Image Processing*, **23**.

[93] Rudin, L.I., Osher, S., Fatemi E., (1992) Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena.* **60**, 259-268.

[94] Gonzales R. Woods R., (1992). Digital Image Processing, *Addison-Wesley Publishing Company*, 443 – 452.

[95] Banterle, F., Corsini, M., Cignoni, P., Scopigno, R., (2011). A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain. *Computer Graphics Forum.* **31**,1.

[96] Felzenszwalb, P.F., Huttenlocher, D.P. (2004) Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision,* **59**, 167–181.

[97] Van de Sande, K. E., Uijlings, J. R., Gevers, T., Smeulders, A. W. (2011). Segmentation as Selective Search for Object Recognition. In 2011 *International Conference on Computer Vision* 1879-1886.

[98] Gulamhussene, G., Joeres, F., Rak, M., Pech, M., Hansen, C. (2020) 4D MRI: Robust sorting of free breathing MRI slices for use in interventional settings. *PLoS ONE* , **15**.

CIRRICULUM VITAE

PERSONAL INFORMATION

Name Surname: Ali Emre ÖZTÜRK

EDUCATION

Degree	University	Department	Year
Bachelor	Gaziantep University	Electrical-Electronics Engineering	2003-2009
Master	Gaziantep University	Electrical-Electronics Engineering	2011-2014
Doctorate	Gaziantep University	Electrical-Electronics Engineering	2014-2021

PUBLICATIONS

1. Öztürk, A.E., Erçelebi, E., (2021) Real UAV-Bird Image Classification Using CNN with a Synthetic Dataset. Applied Sciences. 11, 9.

PATENTS

1. Öztürk, A.E. Taşınabilir bilgisayarlar için portatif joystick, 2015, 2015 / 08663.

2. Acıoğlu A., Öztürk A.E. Sürücü uyku uyarı sistemi, 2015, 2015/00911