DEEP LEARNING-BASED VEHICLE CLASSIFICATION UNDER LOW-QUALITY IMAGINING CONDITIONS

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

ÖZGEN SARI

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2022

DEEP LEARNING-BASED VEHICLE CLASSIFICATION UNDER LOW-QUALITY IMAGINING CONDITIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

ÖZGEN SARI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2022

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Electrical and Electronics Engineering Department, Atılım University.**

_____

Prof. Dr. Reşat Özgür DORUK
Head of Department

This is to certify that we have read the thesis DEEP LEARNING BASED VEHICLE CLASSIFICATION UNDER LOW-QUALITY IMAGING CONDITIONS submitted by ÖZGEN SARI and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Prof. Dr. Ali KARA Co-Supervisor	Asst. Prof.Dr. Yaser DALVEREN Supervisor

Examining Committee Members:

Prof.Dr. Reşat Özgür DORUK Department of Electrical and Electronics Engineering, Atılım University	_____
Asst.Prof.Dr.Yaser DALVEREN Department of Electrical and Electronics Engineering, Atılım University	_____
Ass.Prof.Dr. Şenol PAZAR Department of Computer Programming Biruni University	_____

Date: 12/09/2022

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.

Name, Last Name : Özgen SARI

Signature :

ABSTRACT

DEEP LEARNING-BASED VEHICLE CLASSIFICATION UNDER LOW-QUALITY IMAGINING CONDITIONS

Sarı, Özgen

M.S., Department of Electrical and Electronics Engineering

Supervisor: Asst. Prof. Dr. Yaser DALVEREN

Co-Supervisor: Prof.Dr.Ali KARA

September 2022, 45 pages

Today, the use of intelligent transportation systems (ITS) and traffic surveillance systems for the prevention of traffic jams, accidents, and security problems has a very important role. In such systems, the classification of road vehicles is one of the important key challenges. For this reason, in the literature, a plenty of methods have been proposed so far to ease vehicle classification. Mostly, these methods use high resolution images collected from high quality cameras. However, vehicle classification is not an easy task when low resolution images are used. In addition to this, several environmental factors affect vehicle classification performance, such as different weather conditions (rainy, snowy, and hazy). This thesis proposes a simple convolutional neural network(CNN)-based method for vehicle classification in low resolution surveillance images collected by a standard security camera installed distant from a traffic scene under different weather conditions. In order to assess its performance, the proposed model is tested on a dataset containing tiny and low resolution vehicle images (100 × 100 pixels and 96 dpi) collected in various weather conditions. The model proposed in this thesis is also compared with well-known VGG19-based CNN models in terms of accuracy and loss. Although, the accuracy and loss values of the proposed model are similar to well-known models, it performs better considering the complexity and energy loss.

Keywords: Vehicle Classification, Convolutional Neural Network, Deep Learning, Low Resolution, Low Quality, Intelligent Transportation System.

ÖZ

DÜŞÜK KALİTELİ GÖRÜNTÜLEME KOŞULLARINDA DERİN ÖĞRENMEYE DAYALI ARAÇ SINIFLANDIRMASI

Sarı, Özgen
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Dr.Öğr.Üyesi Yaser DALVEREN
Ortak Tez Yôneticisi : Prof. Dr. Ali KARA

Eylül 2022, 45 sayfa

Günümüzde trafik sıkışıklığının, kazaların ve güvenlik sorunlarının önlenmesi için akıllı ulaşım sistemlerinin (ITS) ve trafik gözetleme sistemlerinin kullanılması çok önemli bir yere sahiptir. Bu tür sistemlerde karayolu taşıtlarının sınıflandırılması önemli kilit zorluklardan biridir. Bu nedenle literatürde araç sınıflandırmasını kolaylaştırmak için şimdiye kadar birçok yöntem önerilmiştir. Çoğunlukla bu yöntemler, yüksek kaliteli kameralardan toplanan yüksek çözünürlüklü görüntüleri kullanır. Ancak, düşük çözünürlüklü görüntüler kullanıldığında araç sınıflandırması kolay bir iş değildir. Buna ek olarak, farklı hava koşulları (yağmurlu, karlı ve puslu) gibi çeşitli çevresel faktörler araç sınıflandırma performansını etkiler. Bu tez, farklı hava koşullarında bir trafik sahnesinden uzağa yerleştirilmiş standart bir güvenlik kamerası tarafından toplanan düşük çözünürlüklü gözetim görüntülerinde araç sınıflandırması için basit bir evrişimsel sinir ağı (CNN) tabanlı bir yöntem önermektedir. Performansını değerlendirmek için önerilen model, çeşitli olup olmadığı koşullarında toplanan küçük ve düşük çözünürlüklü araç görüntülerini (100 × 100 piksel ve 96 dpi) içeren bir veri seti üzerinde test edilmiştir. Bu tezde önerilen model aynı zamanda iyi bilinen VGG19 tabanlı CNN modelleri ile doğruluk ve kayıp açısından karşılaştırılmıştır. Önerilen modelin doğruluk ve kayıp değerleri karşılaştırıldığında iyi bilinen modellerle benzerlik gösterse de karmaşıklık ve enerji kaybı göz önüne alındığında daha iyi performans göstermektedir.

Anahtar Kelimeler: Araç Sınıflandırma, Evrişimli Sinir Ağı, Derin Öğrenme, Düşük Çözünürlük, Düşük Kalite, Akıllı Ulaşım Sistemi.

Dedicated To My Dear Wife Ayşe SARI, My Children Adem and Defne

Their support and patience…

ACKNOWLEDGMENTS

I would like to thank to my thesis supervisor Asst. Prof. Dr. Yaser DALVEREN for his guidance, help, and patience. I would also express my thanks sincerely to my thesis co-supervisor, Prof. Dr. Ali KARA.

A special thank also to Bamoye MAIGA and Sümeyra TAŞ, for their help during my master thesis period.

Lastly, I would like to thank Prof. Dr. Reşat Özgür DORUK and Ass. Dr. Şenol PAZAR who is my examining committee member for their valuable time.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Traffic is an important task all over the world. With increasing technology and the human population, there are lots of vehicles in the traffic. Then, it becomes difficult to classify vehicles by using traditional techniques. The intelligent transportation system (ITS) helps us to overcome that problem. It provides not only existence of transportation facilities effective [1], but also keep traffic safety and vehicle type testing [2]. Moreover, vehicle classification is also important for traffic surveillance which consists searching of vehicles to identify crimes [3].A robust algorithm for vehicle detection and classification has been proposed to overcome the limitation of existing techniques [4]. Traffic monitoring systems using roadside cameras are becoming extensively deployed as described in [5]. Further, another study proposes several deep learning-based frameworks for fusing different modalities (image, radar, acoustic, seismic) through exploitation [6]. A Lidar (light detection and ranging) sensor is used also in vehicle classification with a machine learning algorithm, Random Forest (RF) in [7].The machine learning algorithm, SVM (support vector machine), is also used for vehicle classification [8]. Various deep learning techniques are applied to different problems from highway detection scenarios to computer vision problems [9].In [10], a framework to reduce the complexity of CNN-based AVS methods is also proposed. A Convolutional Neural Network (CNN), a sort of deep learning run by vision data, can be used in classification [11]. In [12], PIR sensors and ultrasonic range finders are used jointly, some measurements are taken and a Bayesian Network and a Neural Network are used to join extracted features that come from these measurements in classification. Moreover, the research describes a comprehensive approach to localizing target vehicles in the video under various environmental conditions [13].A method for detecting, counting, and tracking vehicles in a round-about video is proposed in [14]. A system is capable of real-time to solve the challenge of detecting vehicles is also proposed in [15]. In [16],

a robust vehicle detection and tracking approach using a multi-scale deep convolution neural network is introduced [16].

In recent years, various vehicle classification methods have been proposed [17]. With the growth of processor capacities, machine learning methods have been appeared to be used in vehicle classification. A convolutional neural network (CNN) is also one of these methods that can be effectively used. In fact, its history starts with LeNet [18] in 2012, new methods appeared, AlexNet [19], it continued in 2013 The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [20] along with the Visual Geometry Group VGG methods in 2014 [21], and also Szegedy recently introduced Inception architecture [22].In addition to these, examples of vehicle detection and classification approaches provided in [23] and [24] are based on an improved Faster Region-based Convolutional Neural Network (R-CNN) and Single Shot Detector (SSD), respectively.

1.1. Related Work

The vehicle classification can be divided into two groups sensor-based methods [25] and vision-based methods [26]. The vision-based methods use cameras which are implemented on nearly all roads due to the observation of traffic etc. [27]. In addition to this, nearly all security cameras are generally low resolution to collect data. Some applications deal with images' quality with CNN in the studies of [28]. Besides, a CNN algorithm has a sufficient result on low-quality traffic cameras [29]. In [30], a study is proposed to use vehicles that are blurry, and another study presented in [31] is keen on weather conditions.

In the last decades, many vehicle types of classification have been applied [32], such as pose estimation, color recognition, and type classification are usually treated separately [33]. In addition to this, it extracts the 3D space curve or vehicle and classify them by appearance [34]. The other purposed method which is the deep neural network or deep learning approach is usedfor vehicle detection and vehicle classification [35].

In recent years, frontal images of cars have been reported for fine-grained classification [36]. Yu et al. used Faster R-CNN [37] for fine-grained vehicle classification.

In [38], a deep learning approach that directly performs target tracking and classification in the compressive measurement domain without any frame reconstruction is presented. They aim to analyze traffic flow patterns through fine-tuning pre-trained CNN models on domain-specific low-quality imagery, as captured in various weather conditions and seasons of the year 2018 [39].

The challenge is that the low quality of the traffic camera feeds and to the lack of standardization of cameras and camera positions [40]. Traffic surveillance has always been a challenging task to automate [41]. For an effective visual traffic surveillance system, it is essential to detect vehicles from the images and classify the vehicles into different types [42]. The presence of natural and artificial noise and different light and weather conditions make the detection and recognition process of these systems challenging [43].

1.2. Contributions

This thesis aims to build a simple but efficient CNN method that can classify vehicles under low-quality imagining conditions such as low resolution, highly blurred images, and different weather conditions. The main idea is to extract meaningful insights from the recordings of a particular location. Moreover, the region of interest is different from the other studies as it does not observe the vehicle directly, but takes a side view of the traffic road. This means that, it can be implemented any place to collect data. The dataset is limited number that is 4800 and they are small resolution which is 100 × 100 pixels as used in [44]. Additionally, all images have mixed view, hence this work is related to vehicle classification which considers different environmental conditions.

A CNN-based model was created from scratch with the aim of classifying the vehicles. The performance of this model, which was then created from scratch, is compared to well-known and efficient models such as the VGG19 pre-trained model

and the VGG19 pre-trained fine-tuning model in terms of accuracy and complexity. From the comparison results, it is reported that the VGG19 fine-tuning model provides higher performance. However, the proposed model has a simple and lightweight architecture, providing acceptable accuracy (93.4%) and loss (23%), fewer layers (thirteen layers) and parameters used (approximately 2.2 m). In addition, the proposed model provides a faster training time (51 minutes). These advantages make the proposed model as energy efficient over other well-known VGG19 models. Therefore, the proposed model shows that vehicle classification is possible even with a small dataset containing under low-quality imagining conditions at low resolution recorded by a standard security camera.

The rest of the thesis's structure is organized as follows; Chapter 2 a brief information about deep learning. The proposed method, pre-trained VGG19 method, and fine-tuning VGG19 methods are explained in Chapter 3. All methods are evaluated and analyzed in Chapter 4. Experimental results are given in Chapter 5. In Chapter 6, the conclusion and future works of this thesis study are presented.

CHAPTER 2

A BRIEF INTRODUCTION TO DEEP LEARNING

Artificial intelligence (AI) as the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans because they require human intelligence and discernment. In the 1940s, with the development of digital computers, many difficult mathematical problems can be solved or proven with their help of them[45].

Alan Turing was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist[4]. Turing was highly influential in the development of theoretical computer science, providing a formalization of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer[47][48][49]. He is widely considered to be the father of theoretical computer science and artificial intelligence[50]. In addition to this, in 1959, Ord. Prof. Dr. Cahit Arf said that "Can a machine think and how can it think? [51].

Today, machine learning is widely used in many fields that you will encounter in big data analytics, data mining, classification, or detection. It is divided into three parts supervised learning, unsupervised learning, and reinforcement learning. This thesis related to supervised learning that has both input and output which is learning the map between the input and output.

Unsupervised learning has only input without any output; clustering algorithms, such as K-means and reinforcement learning have input and output but it learns with grades that can be gained by the model.

Researchers have noted in their research that machine learning algorithms can be used in a number of technologies and languages, including Python, R, Java, and others. The "deep" in deep learning comes from the many layers that are built into

the deep learning models, which are typically neural networks. The AI's neural network layer system is inspired by the human nervous system [52]. All actions are transmitted by the neuron which is seen in Figure 2.1 and how they flow the information through from neurons is seen in Figure 2.2.

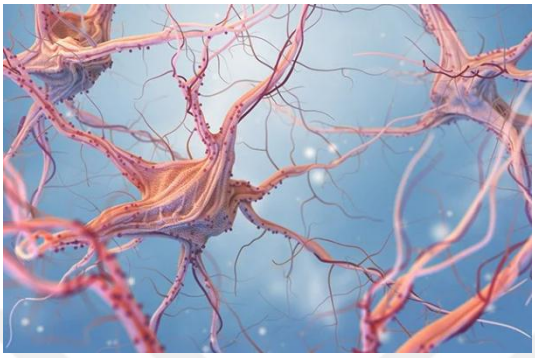


Figure 2.1. Neuron System [53]

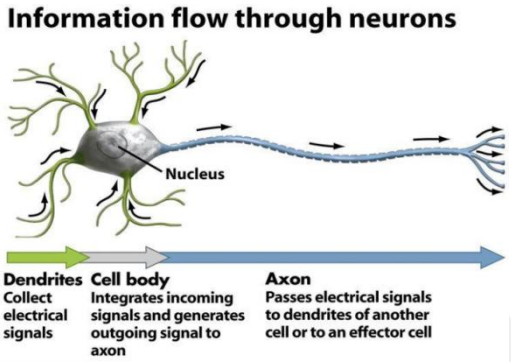


Figure 2.2 Info. Flow Through Neurons [54]

This system which is called deep learning (DL) because it resembles the working principle of neurons, is classified as a a sub-field of machine learning [55] and it deals with a deep neural network. There are lots of neural networks algorithms such as Perceptron, Feed Forward Neural Network, Multilayer Perceptron,Convolutional Neural Network, Radial Basis Functional Neural Network, Recurrent Neural Network, LSTM – Long Short-Term Memory, Sequence to Sequence Models, Modular Neural Network [56].

In recent decades, deep neural network systems have increasingly preferred due to their solving performance of massive problems. With the technological development of chips and growing computer calculation capacity, this ability gives results with very high accuracy in a very fast time. Nowadays, mostly convolutional neural networks (CNN) uses for the capability of vision networks which is the sub-field of DL. The thesis is related to the convolutional neural network (CNN) which is used for vision problems analyzes in deep learning.

CNN is a type of deep learning which is used for evaluating image processing systems in deep neural network operation. When we mention CNN, there is two main part, namely feature extraction and classification as shown in Figure 2.3.

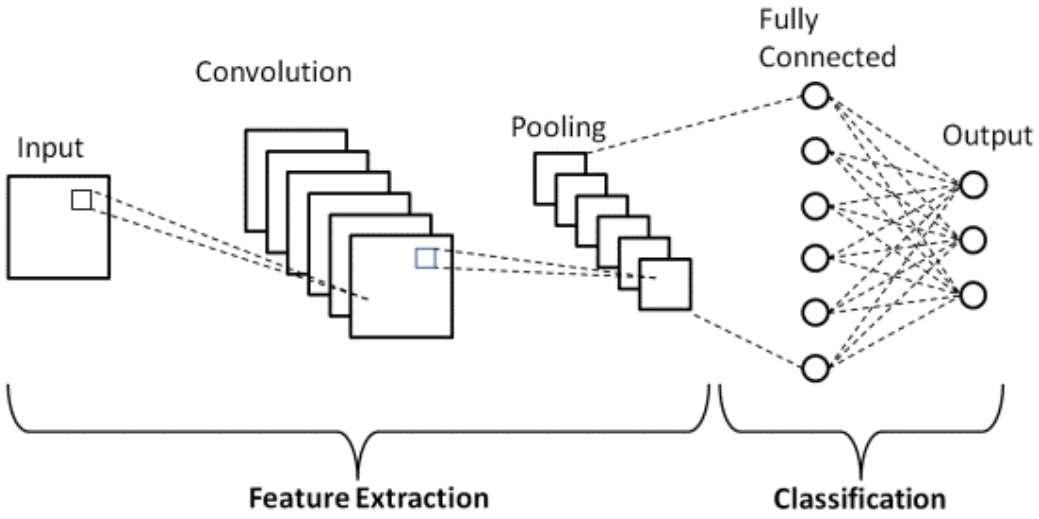


Figure 2.3 Basic CNN Architecture [57]

In the Feature extraction part, convolution layer divides and analyzes the different range of the image by using the convolution operation as shown in Figure 2.4. In this part, there are also other operations which are pooling and activation function sections.

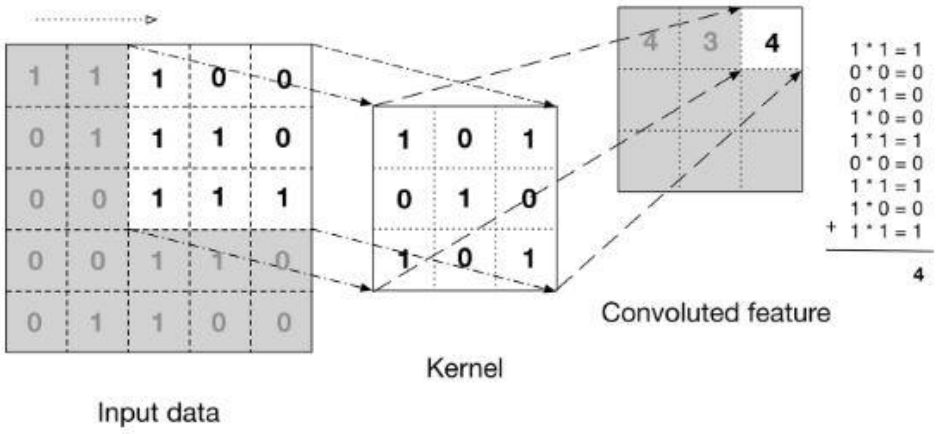


Figure 2.4 Convolution operation in 1D [58]

The pooling layer is a way of saving time due to a decrease in the image size. There are three kinds of pooling which are minimum pooling, average pooling, and maximum pooling. In CNN, max pooling operation is mostly preferred that helps us to get sharpness or edge of the image as shown in Figure 2.5. It has a two-parameter:

a) kernel size which determines the n × n pixel blocks, and b) stride that is an operation how many steps move by k time.

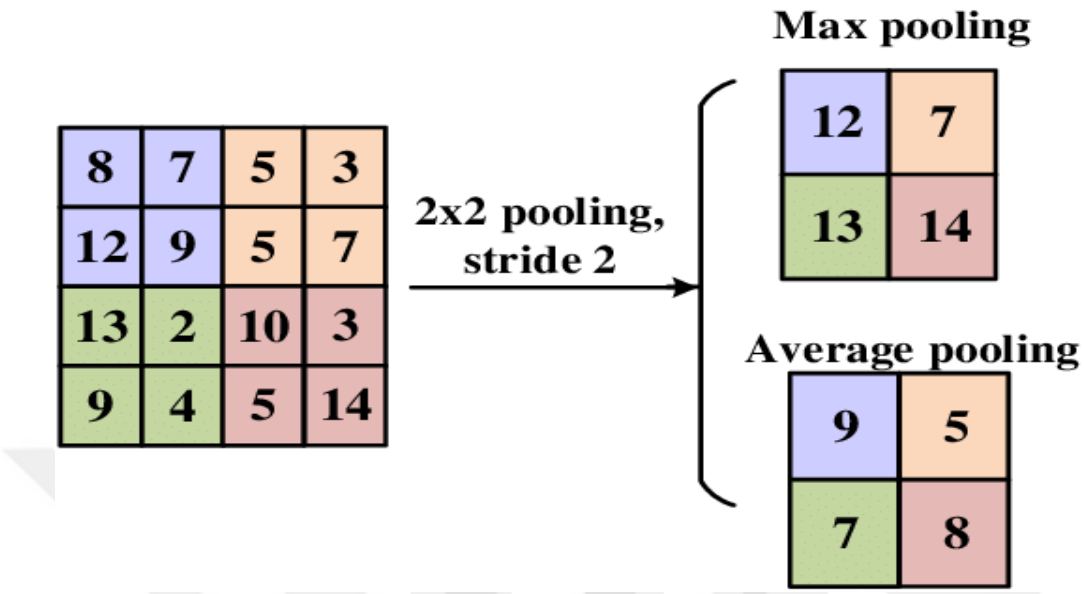


Figure 2.5 Pooling operations [59]

After pooling step, image lost its linearity which means all the images are nearly the same. Various activation functions such as Relu, Tanh, Sigmoid, Softmax, etc. are used to prevent this problem. Relu can be considered as an useful function among the CNN algorithms. For this reason, it was decided to use in this thesis study. In its implementation, it is a function and it takes the value itself, if the value is bigger than zero or it takes '0' instead of the value itself if the value is less than '0' as shown in Figure 2.6

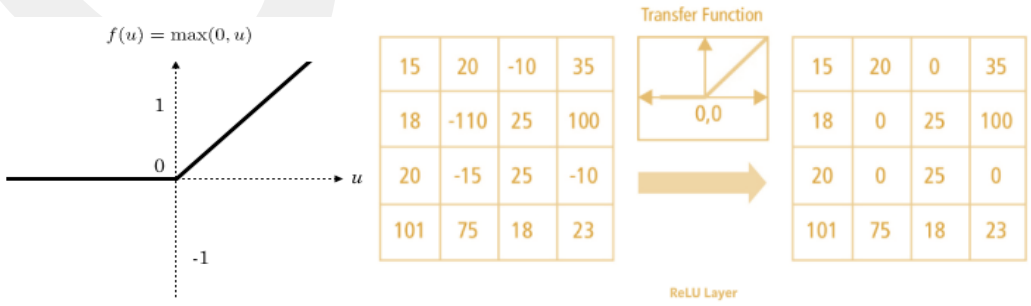


Figure 2.6 Relu Activation Function and operation [60]

Another part of CNN is the classification networks. There is a fully connected (FCN) or dense layer before that layered image operating many functions, so we have to transform the matrix into flattened as shown in Figure 2.7.

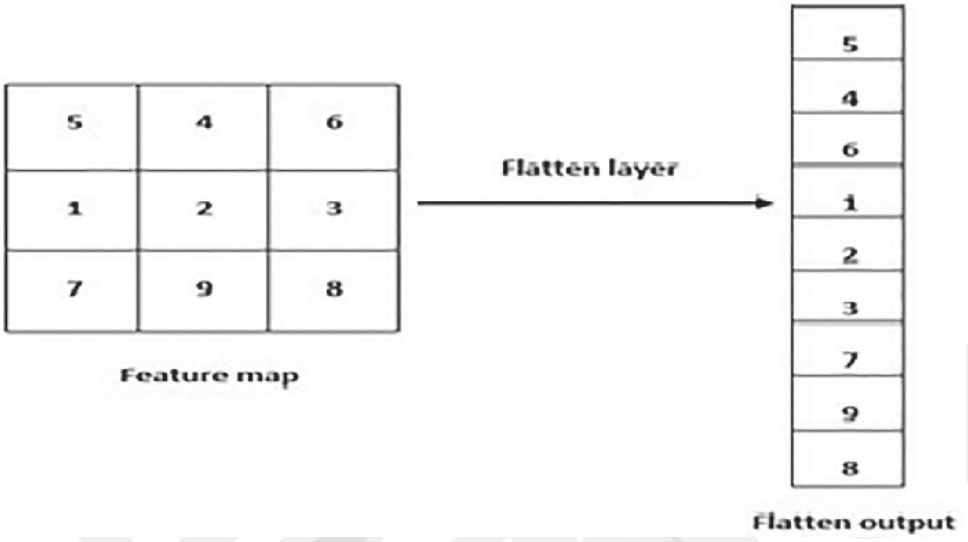


Figure 2.7 Flatten Operation [61]

When the n-dimensional matrix turns into a 1-d vector, then it follows that vector connects fully connected layer as shown in Figure 2.8.

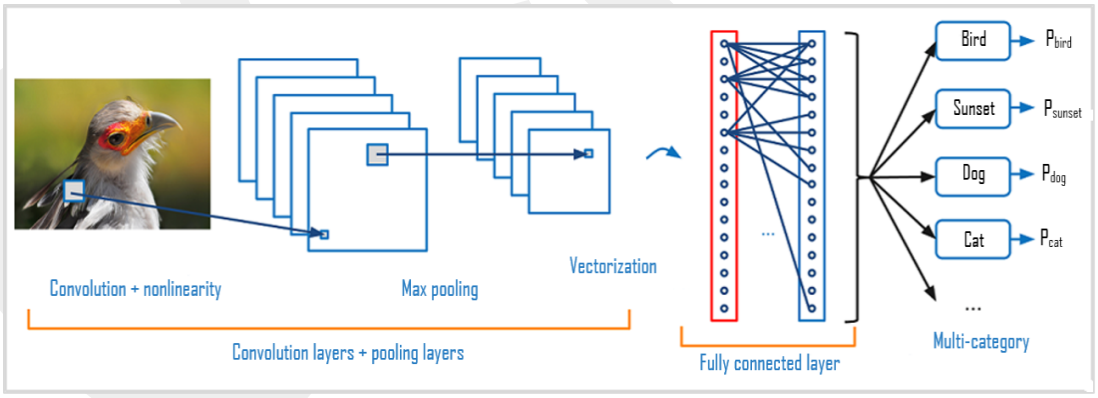


Figure 2.8 Sample of CNN with Fully Connected Layer [62]

Finally, CNN architecture ends with back-propagation algorithms. The weights (w) are updated orderly for decreasing the loss function, between the initial prediction (estimated class) and the label (real class) [63] generally using Stochastic Gradient

Descent (SGD) [64], Root Mean Squared Propagation (RMSProp) [65], and Adaptive Moment estimation (Adam) [67] optimizers which are used to solve optimization problems by minimizing the function.

There are only limitation methods that exist which are related to low-resolution images highly blurred images and different weather conditions. Moreover, the region of interest is(ROI) different from other works which are observed as opposed to the road as shown in Figure 1.1. In addition to this, the thesis network works its limited datasets in low-quality images with different weather conditions and blurred images which is shown in Figure 1.2.

Moreover, this thesis related to traffic surveillance cameras in ITS due to the use of traffic flow, avoiding congestion traffic density and observing traffic vehicles. It is considered that the system's cameras have low-resolution quality. Our focus is then steered to low-quality imagining conditions and also the region of interest is different from other works.

CHAPTER 3

MODELS FOR CLASSIFYING VEHICLES UNDER LOW-QUALITY IMAGINING CONDITIONS

3.1. The Proposed Method

The model is designed with the new structure, there were five convolutional layers and four max pooling layers in the feature extraction network. Then, a flatten operation was applied to transform the feature map that comes from the last pooling operation into a vector column. A fully connected layer (hidden layer) follows this with a dropout layer, which drops some nodes out randomly to make the model smaller to prevent overfitting. Finally, the last class fully connected layer is added on the top for determining six classes which is shown in Figure 3.1.

Figure 3.1Architecture of the proposed model.

3.2. Pre-Trained VGG19 Model

The VGG19 is a well-known CNN architecture and is widely used in many deep learning image classification techniques [66]. The network is trained on a dataset called as ImageNet which contains more than 14 million images. All of the image sizes are 224 × 224 × 3. It is then obvious that the use of this pre-trained network could be an efficient means to improve the accuracy of the proposed model.

The model architecture, a convolutional base of the VGG19 network which in short consists of 19 layers (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers, and 1 SoftMax layer) is used as shown in Figure 3.2. Similar to the proposed model, the remaining stage of the architecture consists of the flatten layer, a fully connected layer, a dropout layer, and a final fully connected layer.

Figure 3.2. Architecture of the Pre-Trained VGG19 model

3.3. Fine-TuningVGG19 Model

Fine-tune is used for the smaller dataset by freezing or tuning some parameters with pre-trained networks. In this method, all layers are the same as the pre-trained VGG19 models, with a fully connected layer added at the end, which not only improves accuracy but also reduces loss.

Figure 3.3 Architecture of the Fine Tuning VGG19 model

CHAPTER 4

EXPERIMENTS

4.1. Data Collection and Pre-processing

As a follow-up study of [44], we created a new dataset containing low-quality vehicle images under different weather conditions. Firstly, we gathered a set of video recordings captured by a standard surveillance camera monitoring a particular square in Konya city, Turkey, for security purposes. Figure 4.1 shows the position of the camera placed on one of the minarets of a mosque located in Konya.

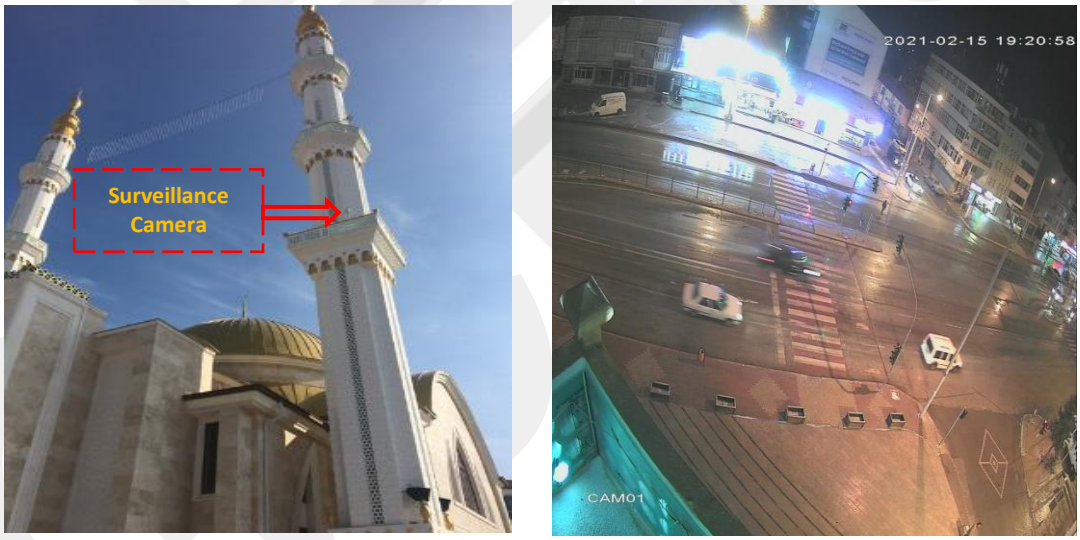


Figure 4.1 Position of the camera placed on the minaret and a view from the camera

Due to the CCTV camera specification, all of the images with 96 dpi resolution which was cropped from the zoomed video frames with the help of a video/media player and also transferred into 100 × 100 × 3 (height × wind × RGB) pixel size image. All cropped images were then distributed under the name of the following folder titles as shown in Figure 4.2.

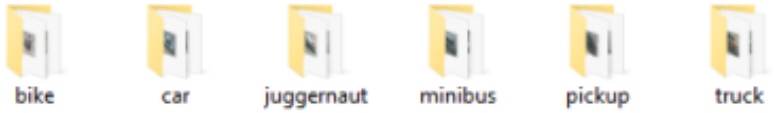


Figure 4.2. Six Vehicle Classes

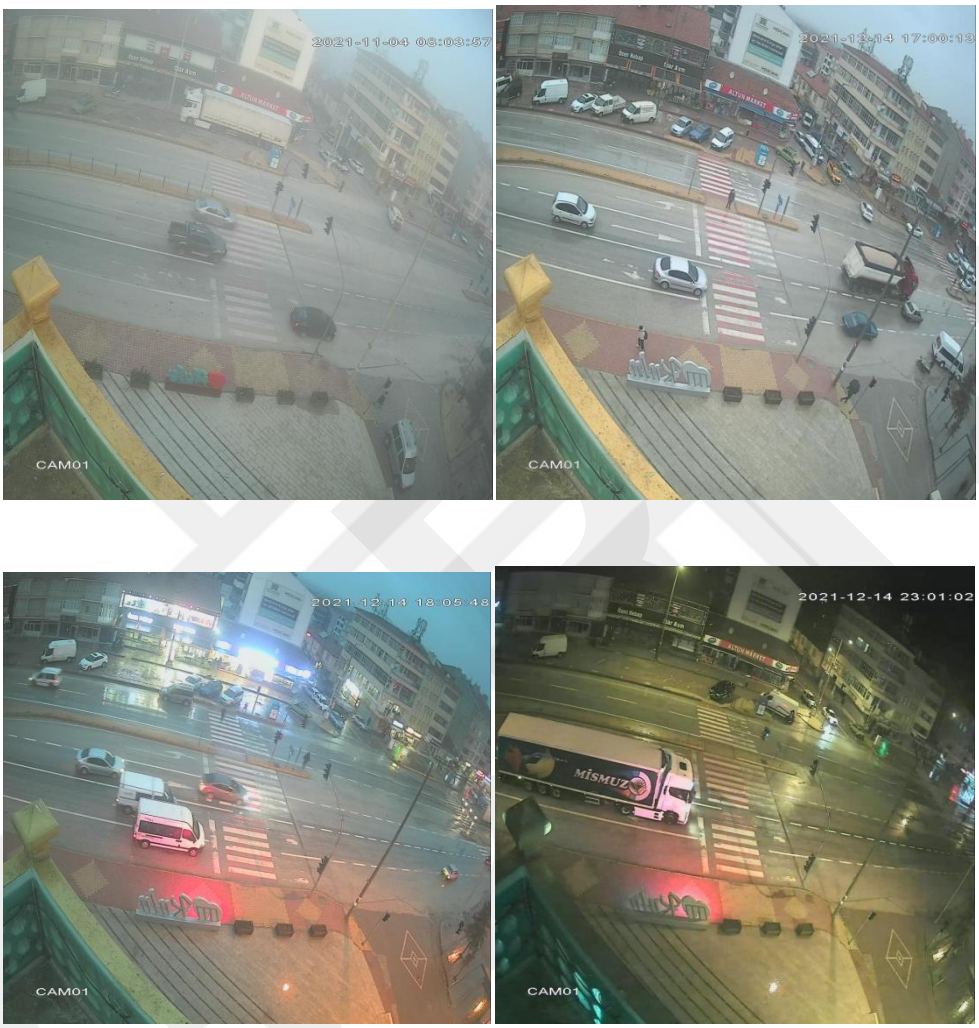


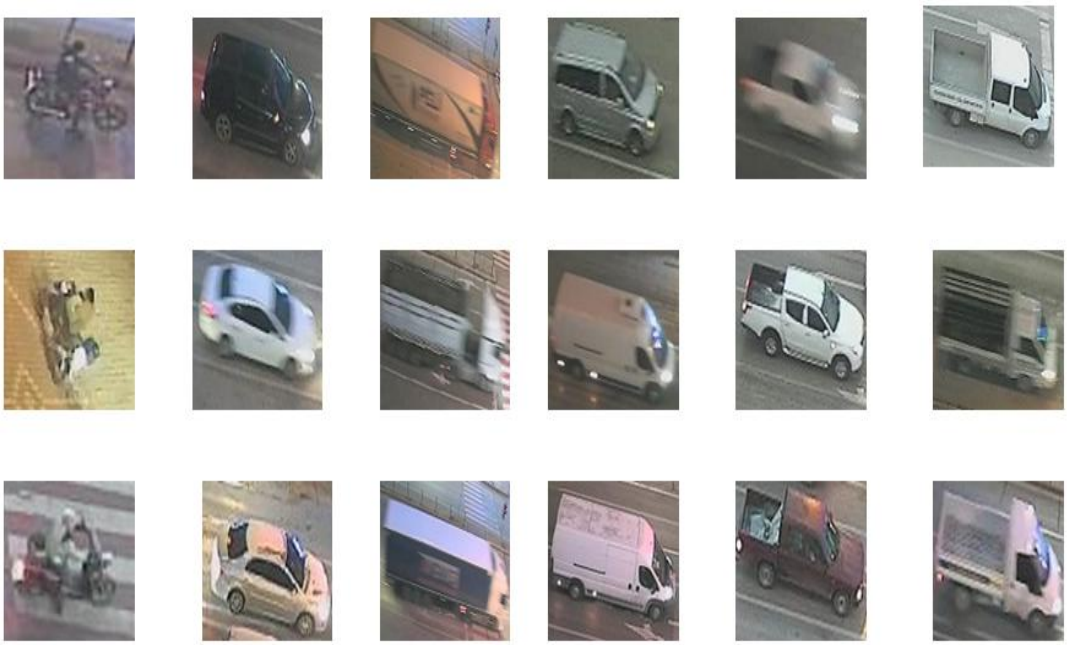
Figure 4.3 Samples of ROI under different weather conditions and blurred images

Figure 4.3 shows samples of ROI under low-quality imagining conditions. Since there might be various situations such as blurred images, different weather conditions, and distorted images, all the vehicles in the images were grouped into six classes.

Table 4.1. Number of images with 6 classes

Class Type	Number of Data
Bike	800
Car	800
Minibus	800
Juggernaut	800
Pick up	800
Truck	800
Total	**4800**

For each class(Bike, Car, Minibus, Juggernaut, Pickup, and Truck), 800 vehicle images were collected as listed in Table 4.1.In this way, the dataset consisted of 4800 vehicle images. Figure 4.4 shows samples of vehicles under low conditions.



a) Bike **b**) Car **c**) Juggernaut **d**) Minibus **e**) Pickup **f**) Truck

Figure 4.4 Samples of vehicles under low quality such as different weather conditions, blurred images

The flowchart representing the stages involved in the data preprocessing is shown in Figure 4.5.First of all, the data was encoded by indexing each class. All data were then resized to 100 × 100 pixels. Next, the features and labels were separated from each other that is followed by the feature normalization.

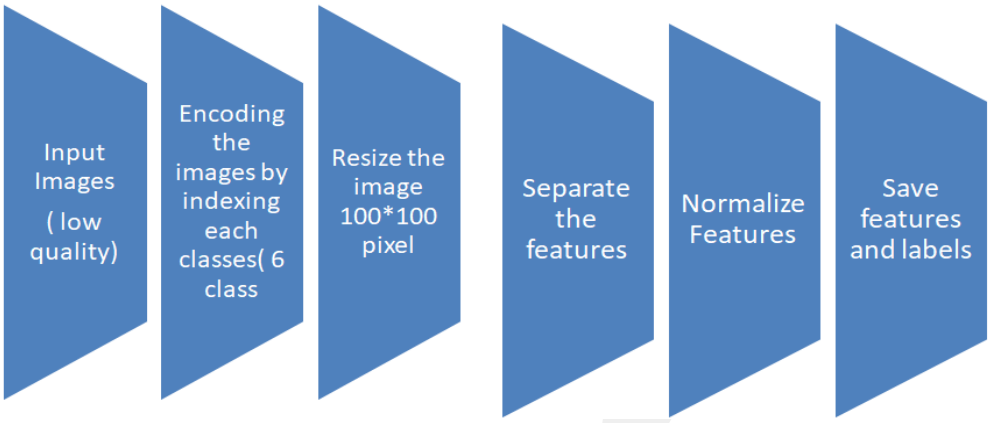


Figure 4.5 The flowchart of data preprocessing.

4.2. Parameters and Training Details

Before the experiments, firstly,we created different type of CNN model for investigating the effect of number of layer and parameters size result of the loss and accuracy.

Data that were prepared for the network for divided into train, validation, and test before training the model. Due to this, train data is used to train the model whereas validation and test data are used to see the model's performance on never-seen data. However, by using two unseen data is that one of them (validation data) is used to arrange the network's hyperparameters (except parameters, learnable values weights, and biases) according to the result of the model on this data and the test data is used to see the trained model's results on another new data. To sum up, the validation set is not used to train but to see the performance of the model to be able to arrange the hyperparameters while training and the test set is used after training to see the performance of the model and whether it is learned (trained) or not.

As the number of layers increases, the number of parameters also increases. In order to prevent the increase in the number of layers, we need to simultaneously reduce the number of CNN filters, kernel sizes or the number of hidden layers, so that the number of parameters decreases or vice versa.

We analyzed in detail the effect of relevant parameters on loss/ accuracy and training time. In this analysis, we observed what kind of results the training time and loss/acuuracy change under which conditions.

Table 4.1.2. Comparison table

# Layers	Accuracy (%)	Loss(%)	# parameters(m)	Training Time (s)
9	84	43	1,19	350
	86	40	2,38	620
	83	46	4,75	1200
10	88	68	0,16	920
	83	59	0,46	920
	84	44	1,2	500
	85	41	1,23	700
11	89	29	2	1220
	89	30	2	1111
	89	30	6,2	2950
12	85	40	0,97	3360
	85	37	1,64	1113
	87	33	1,79	1140
	89	30	3,11	770
	87	37	3,33	1220
	90	28	7,38	1390
	90	26	8,85	2240
13	90	29	2,04	1720
	88	33	2,09	2210
	93	22	2,2	3300
	90	30	2,85	1150
	89	31	11,5	650
	90	32	11,53	3400
	90	31	11,95	1330
	89	32	13,82	950
	88	33	14,7	990
	89	32	17,71	2600
14	90	37	0,1	2200
	91	32	0,14	1650
	93	29	2,7	1800
	94	20	4,2	3600
	92	23	4,8	3505
	93	24	7,9	3700

# Layers	Accuracy (%)	Loss(%)	# parameters (m)	Training Time (s)
15	86	38	0,82	1670
	86	35	0,92	1630
	89	32	1,09	1540
	90	25	1,096	1720
	86	35	1,1	1690
	90	27	1,3	1570
	92	26	2,8	2200
	90	30	3,47	2100
	92	24	4,8	3450
18	90	28	1,62	2530
	91	30	5,83	2630
20	90	28	2,6	1600
	90	30	2,9	3360
	90	30	3,9	1880
21	90	32	2,2	2190
	91	27	4,03	2250
	92	23	4,3	2310
	90	28	5,3	2540
	91	30	5,5	1720
	90	29	5,6	1780
	90	34	6,1	1930
	92	30	6,2	1680
	86	41	11,5	1650
22	85	45	1,7	1600
	91	24	2,8	2390
	90	31	4,1	2290
23	86	39	4,7	2150
	89	32	5,5	950
	89	35	6,1	4880
	91	24	3,01	2320
25	81	50	1,68	1800
	84	44	1,79	1250
	87	35	5,7	1560

As can be seen from Table 4.1.2, whenever we increase the number of layers, the number of parameters increases proportionally or decreases in the opposite case. In order to prevent this increase, we can adjust kernel size, droupout or hidden layer numbers. Considering the obtained results, the following table gives the results

where the accuracy and loss values are ideal, taking into account the parameters and layer concepts.

Table 4.1.3. Ideal results with accuracy and loss

# Layers	Accuracy (%)	Loss(%)	# parameters	Training Time (s)
12	90	26	8,85	2240
13	93	22	2,2	3300
14	94	20	4,2	3600
14	92	23	4,8	3505
14	93	24	7,9	3700
15	90	25	1,09	1720
21	92	23	4,3	2310
23	91	24	3,01	2320

When the Table 4.1.3 is examined, it is seen that the method with the most ideal accuracy (%93) and loss (%24) percentage is the 14-layer CNN method. On the other hand, the 13-layer CNN method, which has similar percentages in terms of loss(%22) and accuracy (%93), is lower in number of parameters(2.2m) and training time (3300s). However, the 13-layer CNN method was preferred as the proposedmethod in this thesis, since the number of parameters and the processing time are directly proportional to the energy consumption.

4.1.4 The proposed model parameters

Data/Layer/ Operation Types	**Information**
Train, Validation, Test Set:	Test separation: 30% from whole data firstly, Validation separation: 30% after test separation, Train separation: All remained data
Convolutional layer:	128 filters, filters' size = 5x5, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	128 filters, filters' size = 5x5, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2 , no padding (downsampling is done)
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	256 filters, filters' size = 5x5, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	128 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	64 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Flatten:	--------
Fully connected/Dense layer:	256 hidden units, (to prevent overfitting), ReLU
Dropout layer:	As 0.4 (40%)(to prevent overfitting)
Last classes' fully connected layer:	6 units (as the classes' numbers), Softmax
Compile:	Rmsporp optimizer with learning rate 0.0001, accuracy (as the measure of the success), categorical cross entropy (cross entropy loss with label encoding)
Training/ Fit:	10 epochs (shuffle training data on each epoch), batch size = 32 (as default)

Secondly, the Pre-trained VGG19 model is used with the same parameters as seen in Table4.1.5

Table 4.1.5. Pre-Trained VGG19 model parameter

Data/Layer/ Operation Types	**Information**
Train, Validation, Test Set:	Test separation: 10% from whole data firstly, Validation separation: 10% after test separation, Train separation: All remained data
Convolutional layer:	64 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	64 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	128 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	128 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Flatten:	--------
Fully connected/Dense layer:	4096 hidden units,
Fully connected/Dense layer:	4096 hidden units,
Dropout layer:	As 0.5 (50%)(to prevent overfitting)
Last classes' fully connected layer:	6 units (as the classes' numbers), Softmax
Compile:	Rmsporp optimizer with learning rate 0.0001, accuracy (as the measure of the success), sparse categorical cross entropy (cross entropy loss with label encoding)
Training/ Fit:	10 epochs (shuffle training data on each epoch), batch size = 32 (as default)

Finally, fine-tuning VGG19method is used with different dense layer sizes on its dataset as shown in Table 4.1.6

Table 4.1.6.Fine-Tuning VGG19 model

Data/Layer/ Operation Types	**Information**
Train, Validation, Test Set:	Test separation: 10% from whole data firstly,Validation separation: 10% after test separation,Train separation: All remained data
Convolutional layer:	64 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	64 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	128 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	128 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	256 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Convolutional layer:	512 filters, filters' size = 3x3, padding (to make the output size the same), stride =1, ReLU
Max-pooling layer:	Filter's size = 2x2, stride = 2, no padding (downsampling is done)
Flatten:	--------
Fully connected layer:	4096 hidden units,
Fully connected layer:	4096 hidden units,
Fully connected layer:	128 hidden units,
Dropout layer:	As 0.5 (50%)(to prevent overfitting)
Fully connected layer:	6 units (as the classes' numbers), Softmax
Compile:	Rmsporp optimizer with learning rate 0.0001, accuracy (as the measure of the success), sparse categorical cross entropy (cross entropy loss with label encoding)
Training/ Fit:	10 epochs (shuffle training data on each epoch), batch size = 32 (as default)

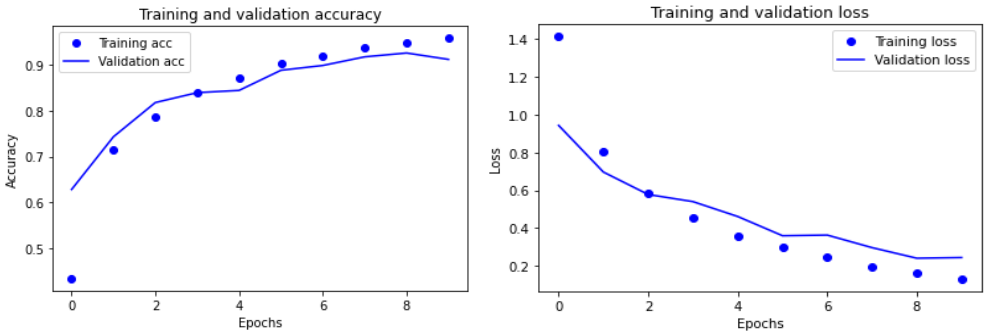
All works were computed in Python 3.8 programming language in the Spyder platform and which was used with Tensorflow and Keras libraries. All networks were trained and tested on a PC server, the specifications were Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz, NVIDIA GeForce 9GTX 1050 Ti,24GB RAM, and Windows 10 (64 bits) operating system.

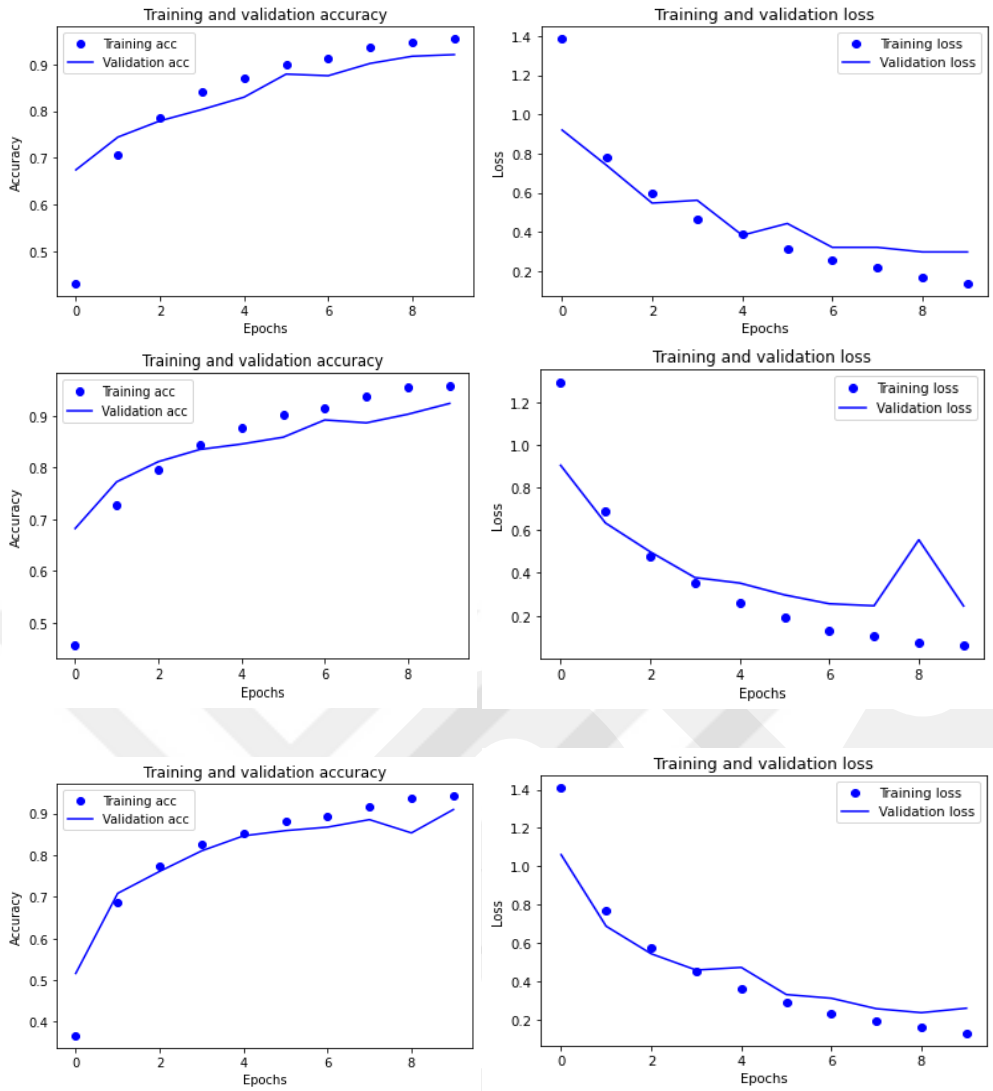
CHAPTER 5

RESULTS

Firstly, the dataset prepared for the network was divided into the train, validation, and test set data. Here, the train set was used to train the model whereas the validation and test sets were used to evaluate the model evaluate performance on never seen data. The validation set was used to tune the network hyperparameters such as except parameters while the test set was used to see how the trained model can generalize its results to other new data. Mostly, when working with CNN models under a limited number of datasets, the test set that should be preferred in order to obtain successful results has been expressed as 25-30% test/validation and 70-75% train. Therefore, in this thesis, both the test set and validation set consisted of 1440 vehicle images (30% for both), while the train set consisted of the remaining 3360 vehicle images (70%).

The training of the proposed model was completed in 10 epochs where the batch size was 32 (default value), the learning rate of the RmsProp optimizer was set at 0.0001, and the dropout layer was 0.4, due to the lower number of image number as seen in Figure 5.1.In order for the machine not to learn by itself, we deleted the previous learning algorithm from the program and ran it again and repeated this process 4 times. The results are shown in Figure 5.1.





(a) (b)

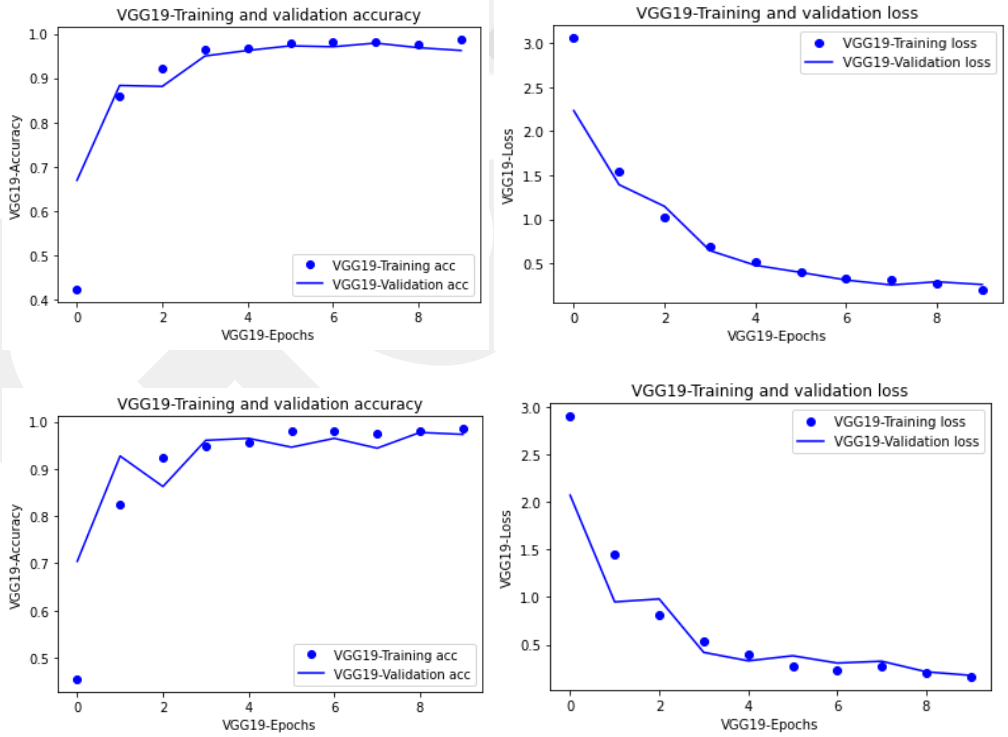
Figure 5.1 The proposed model (a)Training/validation accuracy and(b)Loss with Rmsprop optimizer with 256 dense(hidden) layer after four runs

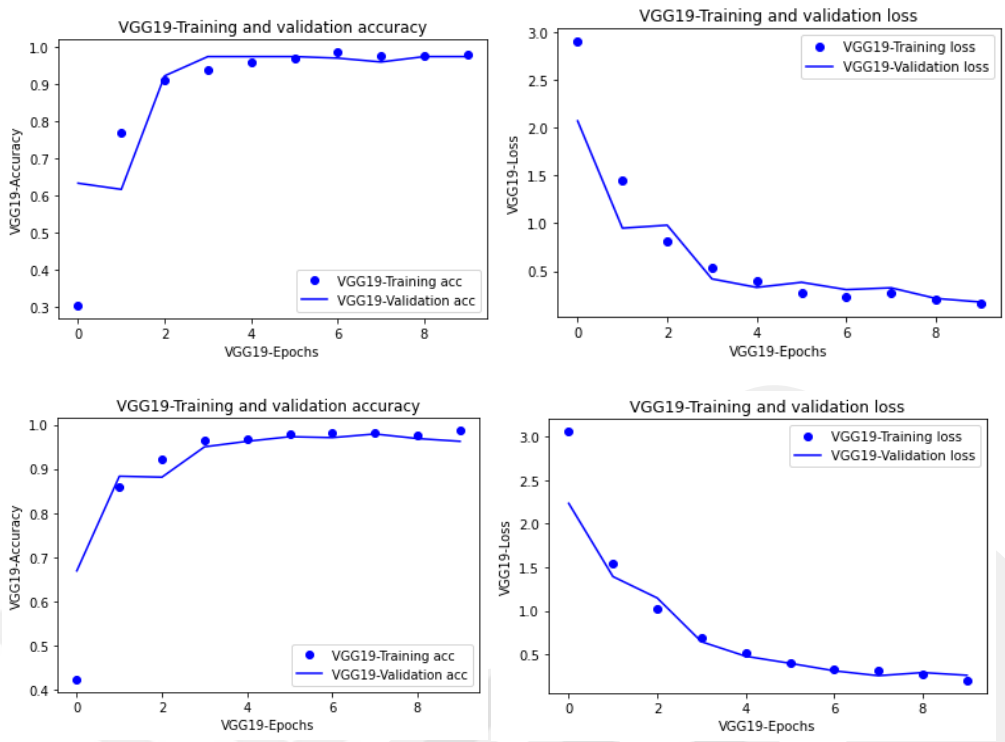
The final results achieved after four runsfor the proposed method are listed in Table 5.1.

Table 5.1 Test accuracy and loss results on proposed method

Order of Run Time with Overall Average	**Time (Sec)**	**Accuracy (%)**	**Loss (%)**
1st	3338	92,1	23
2nd	3300	93,2	21,3
3rd	3340	91,2	20,8
4th	3400	92,7	21,5
Average	**3344,5**	**92,3**	**21,65**

Secondly, the thesis dataset is used in pre-trained VGG19 model. The parameters are; RMSProp optimizer, learning rate 0.0001, dropout is 0.5, the batch size is 32 (default) layer. 6 dense (six vehicle classes) layer and10 epochs in four times that are shown inFigure 5.2 and their results with average in Table 5.2.





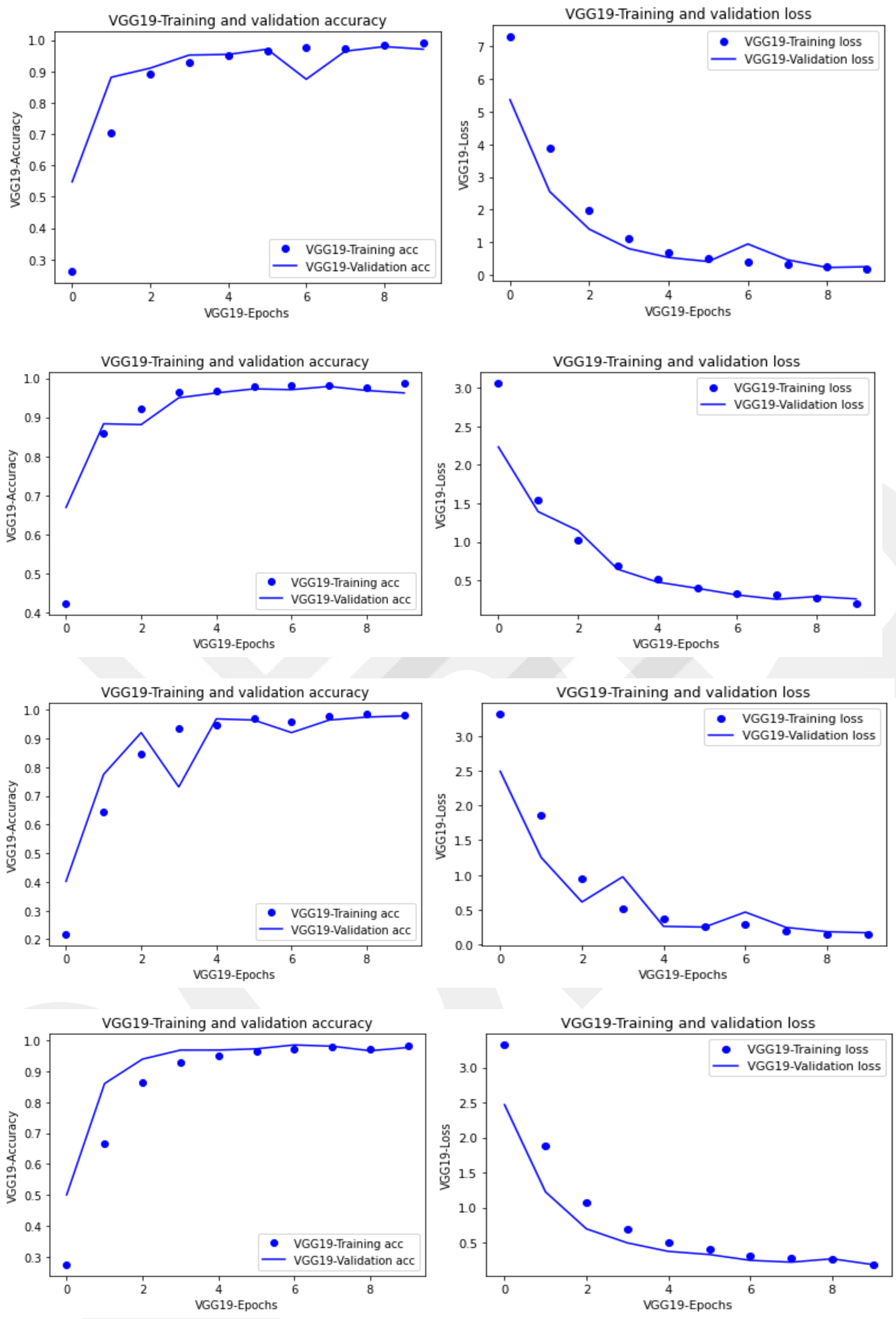
(a) (b)

Figure 5.2 (a) Pre-Trained VGG19 model training/validation accuracy and (b) Loss with Rmsprop optimizer

Table 5.2 Test accuracy and loss results of Pre-Trained VGG19

Order of Running Time with Overall Average	**Accuracy (%)**	**Loss (%)**
1st	95,1	21
2nd	94,3	23,1
3rd	94,7	22,11
4th	95,6	19,90
Average	**94,93**	21,58

Finally, Fine-tuning VGG19 method was implemented independently four times. All results are shown in Figure 5.3, and the averaged results are listed in Table 5.3.



(a) (b)

Figure 5.3 (a) Fine-Tuning VGG19 model training/validation accuracy and (b) Loss with Rmsprop optimizer

Table 5.3 Test Accuracy and loss results on Fine Tuning VGG19

Order of Running Time with Overall Average	**Accuracy (%)**	**Loss (%)**
1st	97,92	16,94
2nd	97,71	18,51
3rd	98,12	16,89
4th	96,51	18,13
Average	**97,57**	17,62

CHAPTER6

CONCLUSION

6.1. Conclusion and Discussion

This thesis is related to vehicle classification under low-quality imagining conditions. All images were cropped from video records in simple security cameras used for observing traffic flow. The records were in low quality because of various lighting conditions, different weather conditions, and blurred images.

Although a small size of dataset was used, the purposed model provides acceptable results in comparison with other well-known CNN models such as pre-trained VGG19 and fine-tuning VGG19. As listed in Table 6.1, Pre-Trained VGG19 and Fine-Tuning VGG19 methods achieved good results to classify the vehicles under low-quality imagining conditions such as different lighting views, various weather conditions, and blurring images. However, when the energy consumption and hardware limitation are concerned, the proposed method can be preferred. Additionally, all methods showed that vehicle classification can be accomplished with the limited dataset consisting low-quality images under different imaging conditions.

Table 6.1. Comparison of the test accuracy and loss for the CNN-based models

Methods with 10 epochs (Average of 4 runs)	**Time (sec)**	**Layer Size**	**# Parameter (m)**	**Accuracy (%)**	**Loss (%)**
Proposed Method	3344	13	2,2	92,3	21,65
VGG19 Pre-Trained Model	3850	24	20,5	94,93	21,58
VGG19 Fine Tuning Model	3600	25	20,9	97,57	17,62

According to the results, it is observed that the layer distribution was directly proportional to the number of parameters. It is found that this can be changed by varying the number of filters, stride or hidden layers in the proposed CNN-based model. With this variation, the accuracy and loss values could be able to compete with other well-known CNN models. Still, the experimental results show that the proposed model established with a limited number of datasets under poor imaging conditions, could be used to classify vehicles under these conditions.

The region of interest (ROI) is different from other researches, so we are deal with not only limited view but also distorted images. This difference caused us to work in a limited area. Inevitably, both the limited field of view and a limited number of distorted pictures make our work unique.

As a final note, there are several open source datasets that have been presented to assist many researchers working on the development of vision-based classification methods [68][69].As an alternative to current datasets, the dataset created in this study will be shared with public for the sake of research community.

6.2. Future Work

The proposed model will improve with some conditions. For example, as a result of training the model by increasing the limited dataset, we want to obtain a new model that targets higher accuracy and lower loss rate, as well as lower energy consumption. We aim to do this with some parameter changes on the proposed method. The thesis system does not perform instantaneous analysis because it is of-line. In the next study, we want to design it as an online system and provide the ability to make instant classification.

REFERENCES

[1] X. Wang, W. Zhang, X. Wu, L. Xiao, Y. Qian, Z. Fang, "Real-time Vehicle Type Classification with Deep Convolutional Neural Networks," *J Real Time Image Proc* vol. 16, pp 5-14, Feb.2019.

[2] L.Suhao ,L. Jinzhao ,L, Guoquan, B. Tong, W. Huiqian, P. Yu, "Vehicle Type Detection Based on Deep Learning in Traffic Scene," *Procedia Computer Science* vol.131, pp.564-572, 2018.

[3] B. Hicham, A. Ahmed and M. Mohammed, "Vehicle Type Classification Using Convolutional Neural Network," *IEEE 5th International Congress on Information Science and Technology (CiSt)*, 2018, pp. 313-316.

[4] S. Kumar, A. Jain, S. Rani, H. Alshazly, S. Ahmed Idris, "Deep neural network based vehicle detection and classification of aerial images," *Intelligent Automation & Soft Computing*, vol. 34, no.1, pp. 119–131, 2022.

[5] A. Chetouane, S. Mabrouk, I. Jemili, M. Mosbah, "Vision-based vehicle detection for road traffic congestion classification." *Concurrency Pract Exper*. vol.34,2022

[6] D. Roy, Y. Li, T. Jian, P. Tian, K. R. Chowdhury, & S. Ioannidis," Multi-modality Sensing and Data Fusion for Multi-vehicle Detection," *IEEE Transactions on Multimedia.vol.1,pp.1,2022.*

[7] J. Wu, H. Xu, Y. Zheng, Y. Zhang, B. Lv, and Z. Tian, "Automatic Vehicle Classification using Roadside LiDAR Data," *Transportation Research Record*, vol. 2673(6), pp. 153-164, 2019.

[8] W. Maungmai and C. Nuthong, "Vehicle Classification with Deep Learning," *IEEE 4th International Conference on Computer and Communication Systems* (ICCCS), 2019, pp. 294-298

[9] A. Çetin, A. Uçar, M.A. Arserim,"Vehicle Detection By Using Deep Learning: A Comparative Study," N, *International Engineering and Natural Sciences Conference* (IENSC 2018),Diyarbakır,2018,

[10] Z. Charouh, A. Ezzouhri, M. Ghogho, Z. Guennoun,"A Resource-Efficient CNN-Based Method for Moving Vehicle Detection," *Sensors* ,vol. 22,pp. 1193. 2022.

[11] W.Mayngmai and C. Nuthong,"Vehicle Classification with Deep Learning, " *IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, 2019, pp. 294-298.

[12] E. Odat, J. S. Shamma and C. Claudel, "Vehicle Classification and Speed Estimation Using Combined Passive Infrared/Ultrasonic Sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1593-1606, May 2018.

[13] A. Jazayeri, H. Cai, J. Y. Zheng and M. Tuceryan, "Vehicle Detection and Tracking in Car Video Based on Motion Model," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 583-595, June 2011.

[14] E. Avşar, Y.Ö. Avşar, " Moving vehicle detection and tracking at roundabouts using deep learning with trajectory union," *Multimed Tools Appl* **81**,pp. 6653–6680 2022.

[15] G. V. Konoplich, E. O. Putin, & A. A. Filchenkov, " Application of deep learning to the problem of vehicle detection in UAV images," *XIX IEEE International Conference on Soft Computing and Measurements (SCM,)2016*,pp. 4-6.

[16] M. Hassaballah, M. A. Kenk, K. Muhammad, & S. Minaee, " Vehicle detection and tracking in adverse weather using a deep learning framework," *IEEE transactions on intelligent transportation systems*, 2020,pp. 4230-4242.

[17] M. Watcharin," Vehicle Classification with Deep Learning," *IEEE Gren Energy and Systems Conference(IGESC)*,2015, pp.17-76.

[18] F. Chollet, " Xception: Deep learning with depthwise separable convolutions," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017 ,pp. 1251-1258.

[19] F. Chollet, " Xception: Deep learning with depthwise separable convolutions," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017 ,pp. 1097-1105.

[20] C. François, "Deep Learning with Depthwise Separable Convolutions,″ *Visualizing and understanding convolutional networks. In Computer Vision – ECCV 2014*, pp. 818-833.

[21] C. François, "Deep Learning with Depthwise Separable Convolutions,″ *Visualizing and understanding convolutional networks. In Computer Vision – ECCV 2014*, pp. 1409-1556.

[22] C. François, "Deep Learning with Depthwise Separable Convolutions," *Going Deeper with convolutions*. In Pro,2015 pp. 1-9.

[23] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[24] D. He, C. Lang, S. Feng, X. Du, &, C. Zhang ,"Vehicle detection and classification based on convolutional neural network," In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service,2015*, pp. 1-5.

[25] B. Yang, Y. Lei,"Vehicle Detection and Classification for Low-Speed Congested Traffic With Anisotropic Magnetoresistive Sensor," *IEEE Sensors Journal* ,2015, pp.1132–1138.

[26] S. Gupte, O. Masoud, R. F. K. Martin and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2002,vol. 3, no. 1, pp. 37-47.

[27] Y. Chen, G. Qin,″ Video-Based Vehicle Detection and Classification in Challenging Scenarios,″ *International Journal on Smart Sensing and Intelligent Systems*, Springer, 2014,pp. 221–227.

[28] U. Mittal, R. Potnuru and P. Chawla, "Vehicle Detection and Classification using Improved Faster Region-Based Convolution Neural Network,″ *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2020, pp. 511-514.

[29] C. M. Bautista, C. A. Dy, M. I. Mañalac, R. A. Orbe and M. Cordel, "Convolutional neural network for vehicle detection in low-resolution traffic videos," *2016 IEEE Region 10 Symposium (TENSYMP)*, 2016, pp. 277-281.

[30] X. Wang, X. Chen and Y. Wang, "Small vehicle classification in the wild using a generative adversarial network," *Neural Comput & Applic*, vol. 33, pp. 5369–5379, 2021.

[31] C. Tsai, K. Tseng, C. Tang and I. Guo, "Vehicle Detection and Classification based on Deep Neural Network for Intelligent Transportation Applications,"*Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018, pp. 1605-1608.

[32] B. Hicham, A. Ahmed, M Mohammed,,"Vehicle Type Classification Using Convolutional Neural Network," *IEEE 5th International Congress on Information Science and Technology (CiSt)*; October 2018, pp. 313–316.

[33] L. Yang, J. Liu, X. Tang,"Object Detection and viewpoint estimation with an auto-masking neural network," *Proc. ECCV*, 2024 ,pp. 441-455.

[34] Z. Yi, L. Li, S. Ling, M. Matt, "Vehicle type Classification using unsupervised convolutional neural Networks," *NIPS* ,2012,pp.1907-1105.

[35] Y. Zhou, H. Nejati, T.T.Do, N.M. Cheung, l. Cheah,"Image-based vehicle analysis using deep neural network:A systematic study," *IEEE International Conference on Digital Signal Processing*, Beijing,2016, pp.276-280.

[36] A. N. Syeda, H. R. Rana, Y. Adeel, "Automatic make and model recognition from frontal images of cars," *Department of Electronics and Power Engineering Pakistan Navy Engineering College National University of Sciences and Technology Karachi*, 2011, pp. 373-378.

[37] S. Yu, Y. Wu, W. Li, Z. Song, and W. Zeng," A model for fine-grained vehicle classification based on deep learning", *Neurocomputing*, vol.257,pp.97-103,2021.

[38] C. Kwan, B. Chou, J. Yang, A. Rangamani, T. Tran, J. Zhang, & R. Etienne-Cummings, " Deep learning-based target tracking and classification for low quality videos using coded aperture cameras," *Sensors*, vol.*19*(17),pp. 3702,2019.

[39] M. V. Peppa, D. Bell, T. Komar, & W. Xiao, "Urban traffic flow analysis based on deep learning car detection from CCTV image series," In *SPRS TC IV Mid-term Symposium "3D Spatial Information Science–The Engine of Change"*. Newcastle University,2018.

[40] D. Mittal, A. Reddy, G. Ramadurai, K. Mitra, & B. Ravindran," Training a deep learning architecture for vehicle detection using limited heterogeneous traffic data," *10th International Conference on Communication Systems & Networks (COMSNETS)* , 2018, pp. 589-294.

[41] R. Theagarajan, F. Pala, & B. Bhanu," EDeN: Ensemble of deep networks for vehicle classification," In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2017,pp. 33-40.

[42] P. Jagannathan, S. Rajkumar, J. Frnda, P. B. Divakarachari, & P. Subramani," Moving vehicle detection and classification using gaussian mixture model and ensemble deep learning technique," *Wireless Communications and Mobile Computing*, *2021*.pp.1

[43] J. Pirgazi, M.M.P. Kallehbasti, A.G. Sorkhi,"An End-to-End Deep Learning Approach for Plate Recognition in Intelligent Transportation Systems," *Hindawi Wireless Communications and Mobile Computing Volume 2022*, pp 13

[44] S. Tas, O. Sari,, Y. Dalveren, S. Pazar, A. Kara, & M. Derawi," Deep learning-based vehicle classification for low quality images" *Sensors*, vol.*22*(13), pp.4740,2022.

[45] B. J. Copeland, Artificial Intelligence, Internet: https://www.britannica.com/technology/artificial-intelligence/Reasoning, Aug 24, 2022, [Sep.13.2022]

[46]Wikipedia, Who was Alan Turing?, Internet : https://en.wikipedia.org/wiki/Alan_Turing, Sep 8,2022 [Sep 13, 2022]

[47] H. M. Alexander, " Alan Mathison Turing," vol.1912-1954 *Biogr. Mems Fell. R. Soc.,pp.*253–263,1955

[48] G. Paul, Alan Turing – Time 100 People of the Century. https://content.time.com/time/subscriber/article/0,33009,991227,00.html, Jun 14, 1999 [Sep 13 2022].

[49] M. Sipser, *Introduction to the Theory of Computation*, Boston, USA: Thomson Learning, 2012.

[50] A. Beavers, "Alan Turing: Mathematical Mechanist," *Elsevier*. pp. 481–487,2013

[51] C. Arf ,"Machine Learning, Deep Learning, and AI: What's the Difference*?, Data Scientist Innovation Day,*Ataturk University, July 2017.

[52] F. Chollet, "Deep Learning with Python,"*Shelter Island*, NY: Manning Publications Co., 2018, pp. 8.

[53] R. Radhakrishnan, " What Are the 4 Main Functions of the Nervous System? ", Internet: www.medicinenet.com/4_main_functions_of_the_nervous_system/article.htm , Sep 10, 2021 [Sep 13,2022]

[54] S.Freeman, *Biological Science*, USA, Pearson Prentice Hall, 2005,pp.100.

[55] F. Chollet, "Deep Learning with Python."*Shelter Island*, NY: Manning Publications Co., 2018, pp. 4.

[56] Great Learning Team, "Types of Neural Networks and Definition of Neural Network," Internet: https://www.mygreatlearning.com/blog/types-of-neural-networks/ , Aug 4,2022 [Sep 13, 2022]

[57] M.K. Gurucharan, " Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network", Internet: https://www.upgrad.com/blog/basic-cnn-architecture/ , Jul 28,2022 [Sep 13 ,2022].

[58] Wikipedia, " Convolution Operation", Internet: http://www.gabormelli.com/RKB/Convolution_Operator, Aug 1,2022 [Sep 12,2022].

[59] H. Yingge, I. Ali and K.Y. Lee, "Deep Neural Networks on Chip - A Survey," *IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020, pp. 589-592.

[60] L. Pauly, D. Hogg, , R. Fuentes, & H. Peel," Deeper networks for pavement crack detection," In *Proceedings of the 34th ISARC* ,2017,pp. 479-485.

[61] M. Suriya, V. Chandran, & M. G. Sumithra," Enhanced deep convolutional neural network for malarial parasite classification," *International Journal of Computers and Applications*,2019,pp. 1-10.

[62] Huawei, " What Is Convolutional Neural Network?", Internet : https://forum.huawei.com/enterprise/en/what-is-convolutional-neural-network/thread/738709-895?page=1&authorid=2976461, May 12,2021 [Sep 13, 2022]

[63] W. Njima, I. Ahriz, R. Zayani, M. Terre,, & R. Bouallegue, " Deep CNN for indoor localization in IoT-sensor systems". *Sensors*, 2019,vol.14,pp. 3127.

[64] S. Ruder," An overview of gradient descent optimization algorithms," *arXiv,2017,pp.1*.

[65] G. Hinton, N. Srivastava, K. Swersky, ″Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.″Available online: https://www.cs.toronto.edu/~hinton/ coursera/lecture6/ (accessed on 20 June 2018).

[66] D. P. Kingma,, & J. Ba,"Adam: A method for stochastic optimization", *arXiv, 2018*,pp.1

[67] K. Simonyan,, & A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv,2016,pp.1409.1556*.

[68] J. Krause, M. Stark, J. Deng, L. Fei-Fei,,″ 3D Object Representations for Fine-Grained Categorization.″ In Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops; December 2013; pp. 554–561.

[69] L. Yang, P. Luo, C.C Loy, X. Tang, ″ A Large-Scale Car Dataset for Fine-Grained Categorization and Verification,″In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 3973–3981.

APPENDIX A

Data Preparing and Proposed Methods Codes

```
import pickle

import numpy as np

import os import cv2 from tqdm

import tqdm import random

DATADIR = " C:/_____our images directory_____/" #defining the path directory

CATEGORIES = ["bike", "car", "juggernaut", "minibus", "pickup", "truck" ] #categories

IMG_SIZE = 100 #image's size

training_data = []

def create_training_data():

    for category in CATEGORIES: # do for each category of vehicle

     path = os.path.join(DATADIR,category) # create path to each category of vehicle

    class_num = CATEGORIES.index(category) #classification (0 to 5).

    for img in tqdm(os.listdir(path)): # iterate over each image per category of vehicle

    img_array= cv2.imread(os.path.join(path,img) ,cv2.IMREAD_COLOR) #convert to array

     new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE)) #resize to normalize data size

    training_data.append([new_array, class_num]) # add this to our training_data

create_training_data()

random.shuffle(training_data)

for sample in training_data[:20]:
```

```
        print(sample[1])
x_feature = []
y_label = []
for features,label in training_data:
        x_feature.append(features)
        y_label.append(label)
x_feature = np.array(x_feature).reshape(-1, IMG_SIZE, IMG_SIZE, 3) # For CNN, 3 dimentional data
#x_train = np.array(x_train).reshape(-1, IMG_SIZE, IMG_SIZE) # For RNN, 2 dimentional data
x_feature=x_feature / 255.0
y_label=np.uint8(y_label)
# Saving data sets
pickle_out=open('C:/_________our features' file's directory_____/x_feature.pkl','wb')
pickle.dump(x_feature,pickle_out), pickle_out.close()
pickle_out=open('C:/_________our labels' file's directory_____/y_label.pkl','wb')
pickle.dump(y_label,pickle_out), pickle_out.close()
```

The Proposed Method Codes

```
## basic libraries → data exploration
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import pylab as pl
from keras import backend as K
import matplotlib.pyplot as plt
from keras.utils import np_utils
from keras.models import Sequential
```

```
from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.layers.core import Dense, Dropout, Activation, Flatten

from tensorflow.keras import layers, activations

from keras import optimizers

import pickle

from keras import regularizers

from sklearn.model_selection import train_test_split

from keras import models

from keras import layers

from PIL import Image

from glob import glob

from tensorflow.keras.models import Model

from tensorflow.keras import regularizers

from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
BatchNormalization, Flatten, Input, Conv1D, Conv2D, MaxPooling2D

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.applications.vgg16 import VGG16

from tensorflow.keras.applications.vgg16 import preprocess_input

from tensorflow.keras.preprocessing.image import img_to_array, load_img

from tensorflow.keras.applications import efficientnet

from keras.applications import VGG16

from keras.preprocessing.image import ImageDataGenerator

base_dir=r"C:/…./…../…../….."

train_datagen=ImageDataGenerator(rescale=1./255,validation_split=0.30)

test_datagen=ImageDataGenerator(rescale=1./255,validation_split=0.30)

train_datagen=train_datagen.flow_from_directory(base_dir,target_size=(100,100),su
bset="training",batch_size=32,shuffle=True)

test_datagen=test_datagen.flow_from_directory(base_dir,target_size=(100,100),subs
et="validation",batch_size=32,shuffle=True)
```

```
"""
print(train_datagen.class_indices)
for _ in range(5):
    img,label=test_datagen.next()
    print(img.shape)
    plt.imshow(img[0])
    print(label[0])
    plt.show()

"""
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=5,padding="same",
activation="relu",input_shape=[100, 100, 3]))
model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=5,padding="same",
activation="relu"))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
model.add(tf.keras.layers.Conv2D(filters=256,kernel_size=5,padding="same",
activation="relu"))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding="same",
activation="relu"))
model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same",
activation="relu"))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(units=256))
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Dense(units=6, activation='softmax'))
```

```
model.summary()


optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001)
loss=tf.keras.losses.CategoricalCrossentropy()
model.compile(optimizer=optimizer,loss=loss,metrics=["accuracy"])
result=model.fit(train_datagen,epochs=10,verbose=1,validation_data=test_datagen)
model.evaluate(test_datagen)
result.history
acc = result.history['accuracy']
loss= result.history['loss']
val_loss = result.history['val_loss']
val_acc= result.history['val_accuracy']
epochs_range = range(10)
plt.plot(epochs_range, loss, 'bo', label='Training loss')
plt.plot(epochs_range, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.figure()
plt.plot(epochs_range, acc, 'bo', label='Training acc')
plt.plot(epochs_range, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

VGG19 Model

```
#loading data from the path
pickle_in=open('C:/……./……./……./……../x_feature.pkl','rb')
x_feature=pickle.load(pickle_in)
pickle_in.close()
pickle_in=open('C:/……./……./……../……/y_label.pkl','rb')
y_label=pickle.load(pickle_in)
pickle_in.close()
#splitting data as train,validation,test
x_train, x_test, y_train, y_test = train_test_split(x_feature,
 y_label,
 test_size = 0.1,
 shuffle = True,
 random_state=None, stratify=y_label)
x_train_, x_val, y_train_, y_val = train_test_split(x_train, y_train,
 test_size = 0.1,
 shuffle = True,random_state=None,stratify=y_train)
#setting VGG19 as convolutional base
from keras.applications import VGG19
base_model=VGG19 (weights= 'imagenet', include_top=False, input_shape =
(100,100,3))
#building the model
for layer in base_model.layers:
    layer.trainable = True
# Flatten the output layer to 1 dimension
x = layers.Flatten()(base_model.output)
# Add a dropout rate of 0.5
x = layers.Dropout(0.5)(x)
```

```
# Add a final softmax  layer with 1 node for classification output
x = layers.Dense(6, activation='softmax')(x)
model = tf.keras.models.Model(base_model.input, x)
model.summary()
model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
#training the model
history=model.fit(x_train_, y_train_, validation_data=(x_val, y_val), epochs=10,
shuffle=True)
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test accuracy: {}".format(test_accuracy))
acc3 = history.history['accuracy']
loss3= history.history['loss']
val_loss3 = history.history['val_loss']
val_accuracy3= history.history['val_accuracy']
epochs = range(1, len(acc3) + 1)
plt.plot(epochs, loss1, 'bo', label='Training loss')
plt.plot(epochs, val_loss3, 'b', label='Validation loss')
plt.title('VGG19 Training and validation loss with Rmsprop')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.figure()
plt.plot(epochs, acc3, 'bo', label='Training acc')
plt.plot(epochs, val_accuracy3, 'b', label='Validation acc')
plt.title('VGG19 Training and validation accuracy with Rmsprop')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```