DEEP LEARNING BASED VEHICLE MAKE AND MODEL CLASSIFICATION

Burak SATAR

T.C.
ULUDAĞ UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEEP LEARNING BASED VEHICLE MAKE AND MODEL CLASSIFICATION

Burak SATAR

Assoc. Prof. Dr. Ahmet Emir DİRİK

(Supervisor)

MASTER OF SCIENCE THESIS
DEPARTMENT OF ELECTRONIC ENGINEERING

BURSA - 2018

TEZ ONAYI

Burak Satar tarafından hazırlanan "Derin Öğrenme Tabanlı Araç Marka ve Model Sınıflandırma" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Uludağ Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman :	Doç. Dr. Ahmet Emir Dirik	İmza
Başkan :	Doç. Dr. Ahmet Emir Dirik Uludağ Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği	İmza
Üye:	Dr. Öğr. Üyesi Murat Türe Bursa Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Mekatronik Mühendisliği	İmza
Üye:	Doç. Dr. Fatih Çavdur Uludağ Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği	İmza

Yukarıdaki sonucu onaylarım.

Prof. Dr. Ali Bayram
Enstitü Müdürü
11.../09/2018 (Tarih)

U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,

- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,

- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,

- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,

- kullanılan verilerde herhangi bir tahrifat yapmadığımı,

- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

07/08/2018
İmza
Burak Satar

ABSTRACT

M.Sc. Thesis

DEEP LEARNING BASED VEHICLE MAKE AND MODEL CLASSIFICATION

Burak SATAR

Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronic Engineering

Supervisor: Assoc. Prof. Dr. Ahmet Emir DİRİK

Many pieces of research have been performed on the vehicle make and model classification recently. This thesis studies the problems regarding this topic. Being able to reach high classification accuracy is one of the main challenges as well as to reduce the annotation time of the images. In this thesis, it is first created a fine-grained dataset by using online marketplaces of Turkey to address these challenges by implementing all experiments on it. Then, it is proposed a pipeline to combine an SSD (Single Shot Multibox Detector) model with a CNN (Convolutional Neural Network) model. In the pipeline, the vehicles are detected by following an algorithm to diminish the time of annotation. The detected vehicles are fed into the CNN model. The results show that the classification accuracy reaches roundly 4% better score when compared with a conventional CNN model. Later, the detected vehicles are picked as Ground Truth Bounding Boxes (GTBB) of the images. Thus, every single image in the dataset contains its GTBB. As a result, they are fed into an SSD model in a different pipeline. By that, it is reached acceptable classification & detection accuracy results even though it is not used perfectly shaped GTBB. Lastly, it is proposed an application which focuses on a use case by using our proposed pipelines. Assuming that license plates are readable, it detects the unlawful vehicles by comparing their license plate numbers and make & models.

Keywords: deep learning, vehicle, classification, CNN, ResNet, detection, SSD
2018, x + 67 pages

ÖZET

Yüksek Lisans Tezi

DERİN ÖĞRENME TABANLI
ARAÇ MARKA & MODEL SINIFLANDIRMA

Burak SATAR

Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ahmet Emir DİRİK

Son zamanlarda araç marka ve model sınıflandırma üzerine çok sayıda araştırma yapılmaktadır. Bu bağlamda karşılaşılan problemler tez kapsamında ele alınmaktadır. Yüksek başarı oranı ile sınıflandırma yapabilmek ve imgelerin etiketlenme süresini azaltabilmek karşılaşılan ana problemler arasındadır. Bu çalışmada, online araç satış sitelerinden veriler toplanarak bir veritabanı oluşturuldu. Tez boyunca yapılan deneylerde bu veritabanı kullanıldı. Sonrasında, SSD (Single Shot Multibox Detector) tabanlı bir model CNN (Convolutional Neural Network) tabanlı bir model ile birleştirildi ve yeni bir model akışı önerildi. Bu bağlamda araçlar bir algoritma aracılığı ile tespit edildi. Bu sayede, etiketleme süresinde önemli bir azalma sağlandı. Tespit edilen araçlar CNN modelinin eğitiminde kullanıldı. Klasik bir CNN modeli ile kıyaslandığında, sınıflandırma başarı oranında yaklaşık olarak %4'lük bir artış görüldü. Akabinde, tespit edilen araçların koordinatları ilgili imgelerin gerçek referans değerleri olarak alınmıştır. Başka bir model akışında, bu imgeler SSD modelinin eğitiminde kullanılmıştır. Sonuç olarak bu model akışında; oldukça iyi tanımlanmamış gerçek referans değerlerine rağmen, kabul edilebilir derecede sınıflandırma ve tespit etme başarı oranlarına ulaşılmıştır. Son olarak, bahsi geçen model akışlarını kullanarak gerçek bir senaryoya odaklanan bir uygulama önerilmiştir. Bu uygulamada, plaka numarası ile araç marka & model bilgisi eşleştirip veritabanı üzerinden kontrolü yapılmaktadır. Plakanın okunulabilir olduğu varsayılmıştır.

Anahtar Kelimeler: derin öğrenme, araç, sınıflandırma, evrişimsel ağ, ResNet, tespit etme, SSD
2018, x + 67 sayfa

ACKNOWLEDGEMENTS

I would like to present my unique thanks to Assoc. Prof. Dr. Ahmet Emir Dirik for his professional supervision and guidance throughout the thesis study at the Uludağ University.

A special thank to my father Müfit Satar, my mother Türkan Satar, and my sister Fulya Satar for their dedicated support and continuing encouragement throughout my study.

I would also like to thank Ayşe Mutlu, Ahmet Anıl Yaylalıoğlu, Bahadır Gölcük, Göktuğ Enes Göktürk and the rest members of Bursa I/O community. Following deep learning based online courses with them kept my progress on schedule. My other thanks to Eyüp Görkem Bayram for helping me to write a web crawler to collect the data through online sources.

Burak Satar
07/08/2018

TABLE OF CONTENTS

Page

ABSTRACT i

ÖZET ii

ACKNOWLEDGEMENTS iii

LIST OF SYMBOLS AND ABBREVIATIONS vi

LIST OF FIGURES viii

LIST OF TABLES x

1 INTRODUCTION 1

2 THEORETICAL FUNDAMENTALS 4

2.1 Machine Learning 4

2.2 Neural Networks 5

2.3 Convolutional Neural Networks 7

2.3.1 Convolution 8

2.3.2 Max pooling 9

2.3.3 Hyper-parameters 10

2.3.4 Activation functions 11

2.3.5 Zero padding 13

2.3.6 Dropout 13

2.3.7 Batch normalization 14

2.3.8 Adaptive moment estimation (Adam) optimizer 15

2.3.9 Categorical cross entropy loss 16

2.4 Residual Networks (ResNet) 16

2.5 Single Shot MultiBox Detection (SSD) 18

3 MATERIALS and METHODS 22

3.1 Dataset Generation 23

3.2 Pre-processing 26

3.3 Vehicle Classification and Detection 29

3.4 An application 33

4 RESULTS 35
4.1 Experiment I: Classification by a ResNet Model 37
4.2 Experiment II: Classification by a ResNet + an SSD Model 41
4.3 Experiment III: Detection by an SSD Model 45
5 DISCUSSION AND CONCLUSION 54
REFERENCES 56
APPENDICES 59
APPENDIX 1 — PRECISION-RECALL AND AVERAGE PRECISION 60
APPENDIX 2 — RESNET CODE BLOCK 63
CURRICULUM VITAE 66

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols	**Explanation**
α	Learning rate
b_x	Centre point of a bounding box over x axis
b_y	Centre point of a bounding box over y axis
b_h	Height of a bounding box
b_w	Width of a bounding box
β_1	The exponential decay rate for past gradients
β_2	The exponential decay rate for past squared gradients
ε	Trivial number to prevent a division by zero
μ	Mean
σ	Standard deviation
W_i	Weights of i_{th} block
W_s	Weights of shortcut block
x	Input to a perceptron

Abbreviations	**Explanation**
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DL	Deep Learning
FN	False Negative
FP	False Positive
FPS	Frame Per Second
GPU	Graphics Processing Unit
GTBB	Ground Truth Bounding Boxes
ILSVRC	ImageNet Large Scale Visual Recognition Competition
mAP	Mean Average Precision
ML	Machine Learning
MS COCO	Microsoft Common Objects in Context
NMS	Non-Maximum Suppression
OpenCV	Open Source Computer Vision Library
RAM	Random Access Memory
ReLU	Rectified Linear Unit

ResNet	Residual Network
RMSProp	Root Mean Square Propagation
SSD	Single Shot Multibox Detector
TN	True Negative
TP	True Positive
TurkStat	Turkish Statistical Institute
VGG	Visual Geometry Group
VOC	Visual Object Classes

LIST OF FIGURES

Page

Figure 2.1. The difference among AI, ML and DL (Goodfellow et al. 2016) 4

Figure 2.2. Performance comparison between deep learning and other neural networks (Ng 2017) 5

Figure 2.3. Biological neurons (Kriesel 2007) 5

Figure 2.4. A perceptron (Nielsen 2015) 6

Figure 2.5. The architecture of a basic neural network 6

Figure 2.6. A convolutional neural network, LeNet architecture (LeCun and Bengio 1998) 7

Figure 2.7. Convolution process 9

Figure 2.8. Max pooling process with the stride of 2 10

Figure 2.9. Choosing the learning rate 10

Figure 2.10. Activation functions 11

Figure 2.11. Softmax classifier 12

Figure 2.12. Zero padding 13

Figure 2.13. Using Dropout in a model 14

Figure 2.14. Training error performance between a plain network and a ResNet model (Ng 2017) 17

Figure 2.15. A ResNet block (He et al. 2015) 18

Figure 2.16. SSD framework [taken from (Liu et al. 2015)] 19

Figure 2.17. Intersection over union 20

Figure 3.1. Jupyter notebook architecture 22

Figure 3.2. Normalization of the data (Fei-Fei Li 2017) 26

Figure 3.3. Some data augmentation techniques 27

Figure 3.4. Adding zero-padding to an image 28

Figure 3.5. Linear and non-linear classification 29

Figure 3.6. Differences among classification, classification with localization and detection 30

Figure 3.7. Showing a bounding box 31

Figure 3.8. Custom ResNet architecture 32

Figure 3.9. The architecture of SSD Model for Detection 33
Figure 3.10. A use case diagram, Case I: license plates match however models don’t . 34
Figure 4.1. Overview of Experiments 35
Figure 4.2. Loss graph of Experiment I on (224,224,3) shaped test set 38
Figure 4.3. Accuracy graph of Experiment I on (224,224,3) shaped test set 38
Figure 4.4. Confusion matrix of Experiment I on (224,224,3) shaped test set 39
Figure 4.5. Loss graph of Experiment I on (300,300,3) shaped test set 40
Figure 4.6. Accuracy graph of Experiment I on (300,300,3) shaped test set 40
Figure 4.7. Confusion matrix of Experiment I on (300,300,3) shaped test set 41
Figure 4.8. Loss graph of Experiment II on (224,224,3) shaped test set 42
Figure 4.9. Accuracy graph of Experiment II on (224,224,3) shaped test set 42
Figure 4.10. Confusion matrix of Experiment II on (224,224,3) shaped test set 43
Figure 4.11. Loss graph of Experiment I on (300,300,3) shaped test set 44
Figure 4.12. Accuracy graph of Experiment I on (300,300,3) shaped test set 44
Figure 4.13. Confusion matrix of Experiment II on (300,300,3) shaped test set 45
Figure 4.14. Loss graph of Experiment III with VGG based weights on the test set . . . 46
Figure 4.15. Accuracy graph of Experiment III with VGG based weights on the test set 46
Figure 4.16. Confusion matrix of Experiment III with VGG based weights on the test set 47
Figure 4.17. Loss graph of Experiment III with fine-tuned weights on the test set 49
Figure 4.18. Accuracy graph of Experiment III with fine-tuned weights on the test set 49
Figure 4.19. Confusion matrix of Experiment III with fine-tuned weights on the test set 50

LIST OF TABLES

Page

Table 3.1. Class distribution of the dataset 24

Table 3.2. Inner distribution of the other class in the dataset 24

Table 4.1. Comparison of the experimental results 37

Table 4.2. True decisions on our fine-grained dataset 51

Table 4.3. False decisions on our fine-grained dataset 52

Table 4.4. True decisions on the test videos 53

Table 4.5. False decisions on the test videos 53

1 INTRODUCTION

Vehicle make-model classification and detection have become one of the most significant subjects in intelligent transportation systems. There are many studies on this topic with different approaches. The very first studies focus on understanding the existence of a vehicle (Gupte et al. 2002). Some studies only focus on classification of vehicle type (Buch 2010). Some other studies imply several feature extraction to recognize the logo of the vehicle for the classification of vehicle make (Singh 2012). The other studies focus on vehicle make-model classification by using a conventional method like Haar cascades (Baran et al. 2015). However, it is computationally costly and not suitable for real-time cases. There are also studies that use deep learning with various degree for vehicle make-model classification (Tafazzoli et al. 2017; Zhou and Cheung 2016; Liu 2016). This thesis proposes a deep learning approach to vehicles classification and detection by handling some principal lacks of previous studies. The first issue is related with the dataset which corresponding to the number of images per class, the types of the images, and annotation methods. The second issue is related to reach high accuracy results regarding classification and detection. The third issue is relevant to turn these studies into practice in such a way that can be implied on a use case.

Foremost, the number of available datasets which include various vehicles are quite limited. There are two common open-source datasets which are called the Stanford Cars dataset (Yang et al. 2015) and the Comprehensive Car (Krause et al. 2015) dataset. They mainly contain fewer images per class and don't include widely known vehicles especially for many countries. In this thesis, Turkey-specific dataset has been created to address this subject. The dataset is generated by gathering the images via online sources such as vehicle marketplace websites. It is developed a script to use it as a web crawler to pick the photos. Thus, it is formed a fine-grained dataset. Besides, the process of annotation of the dataset is also an important subject. Since it is generally worked on a vast dataset in deep learning, making all annotations in manual could take a substantial amount of time. For example, three million images should be annotated manually in some studies

(Dehghan et al. 2017). Instead of annotating manually, a method like Amazon Mechanical Turk could be used to save time. However, it could be a costly choice when the dataset includes a significant amount of data. In this thesis, Algorithm 3.1 is followed to make the annotation process in a semi-autonomous way. Thus, the time and the computation cost are reduced for annotation.

Moreover, plenty of similar studies implement only CNN based architectures (Simonyan and Zisserman 2014) for vehicle make-model classification (Dehghan et al. 2017). However, it is seen that reaching high classification accuracy results is still an issue. This thesis performs three different experiments to examine and solve the issue of reaching high accuracy using our dataset. In the first experiment, a Residual Network (ResNet) (He et al. 2015) based model is implemented to compare with the other two pipelines. This experiment refers to a classical way to approach the problems. Normalization and data-augmentation processes are applied to the dataset. Then, the images are fed into a ResNet model for classification. This pipeline is called Experiment I. In the second pipeline, the vehicles are detected by an SSD (Liu et al. 2015) model. This model is pre-trained on Microsoft Common Objects in Context (MS COCO) (Lin et al. 2014) and PASCAL Visual Object Classes (VOC) (Everingham et al. 2015) datasets. The detected vehicles are sent through the same pre-processing methods. They are fed into the same ResNet model for classification. We call it Experiment II. It is shown that Experiment II reaches a higher classification accuracy result than Experiment I. Next, the coordinates of detected vehicles are selected as GTBB of the images in Experiment III according to Algorithm 3.1. The weights of the SSD model are fine-tuned on our dataset. When it is compared with the Experiment I, it is obtained a reasonably close classification accuracy result in Experiment III. It should be noted that it is reached this score without possessing GTBB of the images which don't have a perfect shape.

Furthermore, it is proposed an application to use our make & model classification methods on a use case. The use case is related to the detection of an authorized vehicle. Plenty of researches address the topic of reading & recognizing the license plates (Du et al. 2013; Chang et al. 2004). However, there are some moments that detection of license plate number of a vehicle may not help to catch the illicit vehicle. For instance, if the detected license plate is recurrent, it is hard to define which car is unauthorized especially in urgent cases. With this motivation, the proposed make & model classification methods are suggested to combine with detecting the license plate to catch the unlicensed vehicle. In this use case, it is assumed that license plates are readable. Reading and recognizing the license plates are out of the scope of this thesis. Instead, an open source project is used for implementation of reading the license plates (Dahms 2016).

The rest of the thesis is organized as follows. Section 2 introduces the materials and methods that this research contains. It first points out what it is a neural network and how it is evolved to the convolutional networks. Then, it gives details corresponding to classification and detection which are used in the experiments. Section 3 provides all the details about the experimental setup to implement the experiments. It includes data generation, annotation, testing models, classification and detection architectures respectively. Section 4 discusses the results of the experiments. Section 5 presents the discussion and conclusion. It also refers to future studies.

2 THEORETICAL FUNDAMENTALS

2.1 Machine Learning

Artificial Intelligence (AI) is first introduced around 1950's by John McCarthy. It aims to have machines which can imitate what the human mind ordinarily conducts. There are mainly two different concepts about AI. First is General AI, also known as human-level AI. It refers to a machine that can perceive the world and reason its environment as a human does. Second is Narrow AI. This work only aims to focus on the second concept which refers that machines can imitate particular focused areas in which a human does. Machine Learning (ML) algorithms are introduced to make those concepts happen. Machine learning includes a vast amount of approaches like Bayesian networks, decision tree learning, clustering, deep learning, etc (Mitchell 1997). Figure 2.1 shows how Deep Learning (DL), ML and AI are correlated with each other. As a conclusion, it is stated that deep learning is a branch of machine learning to reach several artificial intelligence concepts. In this thesis, DL approach is chosen for various reasons.

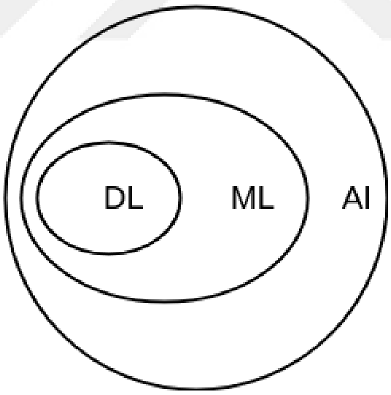


Figure 2.1. The difference among AI, ML, and DL (Goodfellow et al. 2016)

DL could help to boost the performance when the given data is vast. Since this thesis aims to continue to extend its dataset, increase to performance, and widen the applications on the use cases; deep learning is chosen for vehicle make and model classification tasks. Figure 2.2 shows the performance benchmark of various neural network models.

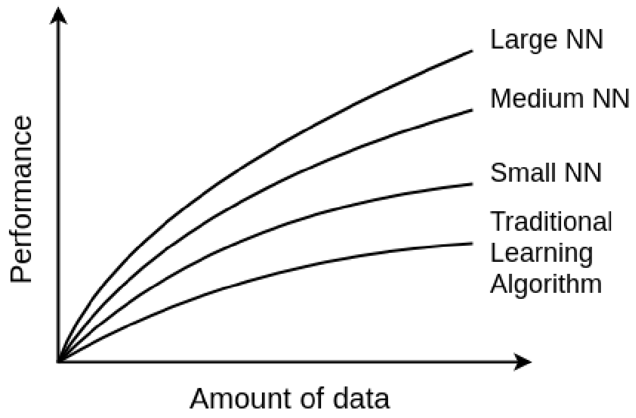
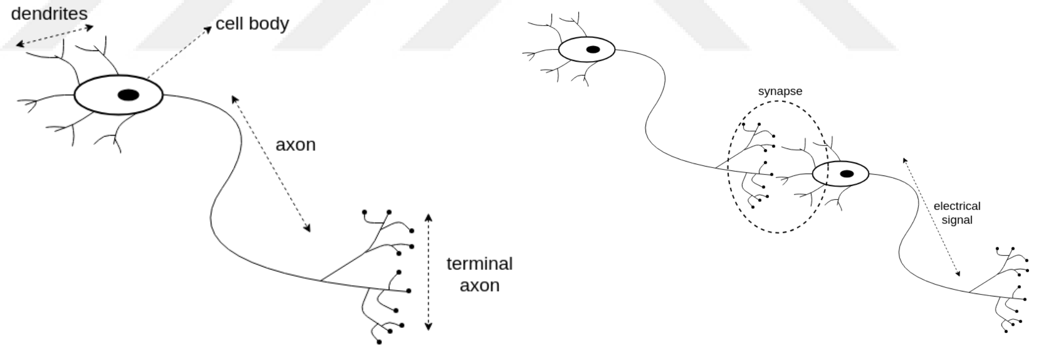


Figure 2.2. Performance comparison between deep learning and other neural networks (Ng 2017)

2.2 Neural Networks

DL uses Artificial Neural Networks (ANN) to simulate what a real neural network in a brain does. Figure 2.3 shows how a biological neural network works.



(a) A biological neuron

(b) The connection between the two neurons

Figure 2.3. Biological neurons (Kriesel 2007)

It starts with the smallest node, an artificial neuron which is also called perceptron as shown in Figure 2.4. A perceptron includes a sum function to sum over all inputs or outputs of other nodes. It acts as a transfer function. Then, an activation function is implemented to keep non-linearity for further layers.

According to the activation function, an output is sent to other nodes. The output of a perceptron can be shown in Equation 2.1.

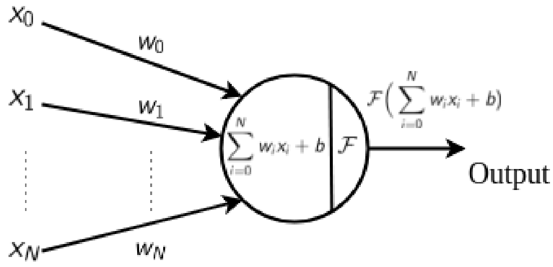

x_0
w_0
x_1
w_1
x_N
w_N
$\sum_{i=0}^{N} w_i x_i + b$
$\mathcal{F}$
$\mathcal{F}\left(\sum_{i=0}^{N} w_i x_i + b\right)$
Output

Figure 2.4. A perceptron (Nielsen 2015)

$$\text{output} = \begin{cases} 0 & \text{if } \mathscr{F}\left(\sum_{i=0}^{N} w_i x_i + b\right) \leq \text{threshold} \\ 1 & \text{if } \mathscr{F}\left(\sum_{i=0}^{N} w_i x_i + b\right) > \text{threshold} \end{cases} \tag{2.1}$$

Many neurons variously connect with each other and form a multilayer neural network which is shown as in Figure 2.5. In this example, it includes five perceptrons as an input. It has two hidden layers with three and four perceptrons respectively. It contains an output layer at the end. As a convention, input layer isn't added for counting the layers of the network. Thus, it can be said that this network is a three-layer neural network.

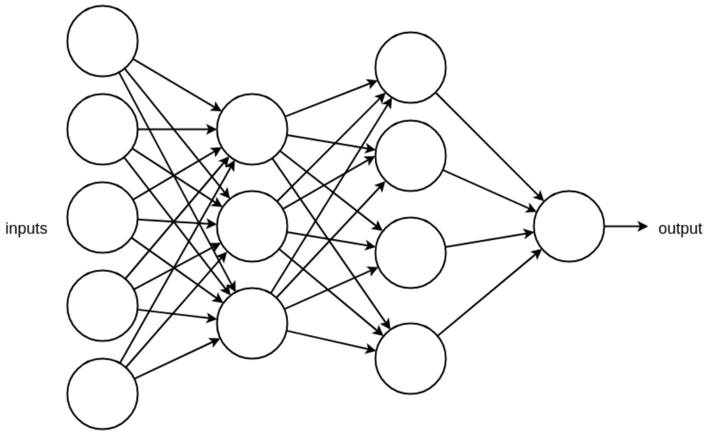


Figure 2.5. The architecture of a basic neural network

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are generally used in computer vision related tasks and consist of various processes. Firstly, the main advantages and the reasons to use are explained. Then, the processes of CNN are introduced such as convolution, max-pool, etc. Figure 2.6 demonstrates the architecture of LeNet (LeCun et al. 1998) which is known as the first CNN model. The input shape is shown as $28 \times 28 \times 6$. In this notation; first value refers to the width of a filter, second value refers to the height of the filter, and third value refers to the depth which means to the filter number.

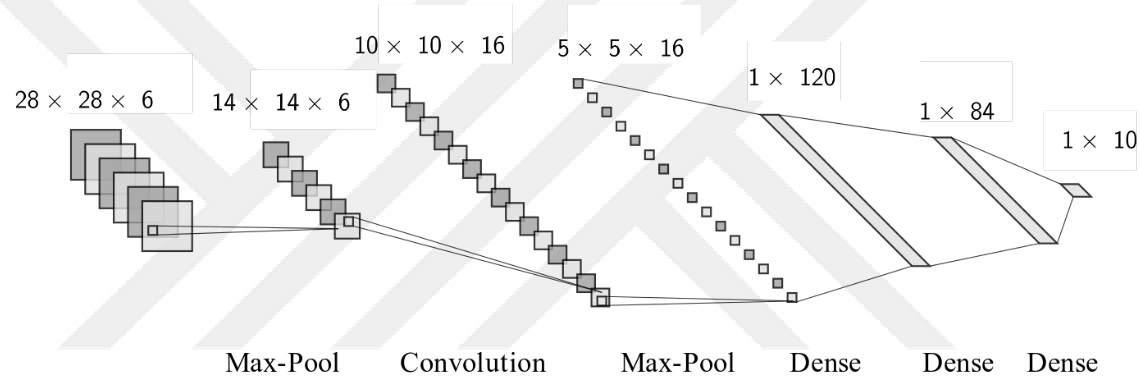


Figure 2.6. A convolutional neural network, LeNet architecture (LeCun and Bengio 1998)

There are two significant advantages of a CNN model. The first one is reducing to the parameters of the filters dramatically which are needed to be learned. When there are lots of parameters to learn and needing plenty of neurons, lots of problems could start to occur. If there is an image with high resolution as an input, it causes some million value referring to the pixels. It would need a very deep fully connected layers which may lead to over-fitting and requiring too much computational power. However, CNNs can reduce the probability of over-fitting and save computation power, especially in computer vision related tasks.

Total parameters are calculated by using standard neural networks and by using CNN based model respectively to observe the differences. They are both calculated only for one layer. Equation 2.2 shows the output when it is used a classical neural network instead of using CNN. It would become;

$$\begin{aligned}\text{\# of Total Parameters} = &(\text{filter\_height} \times \text{filter\_width} \times \text{filter\_depth} + 1) \times \\ &(\text{new\_height} \times \text{new\_width} \times \text{filter\_number})\end{aligned} \tag{2.2}$$

However; Equation 2.3 shows the output when it is used CNN. It dramatically reduces the number of parameters.

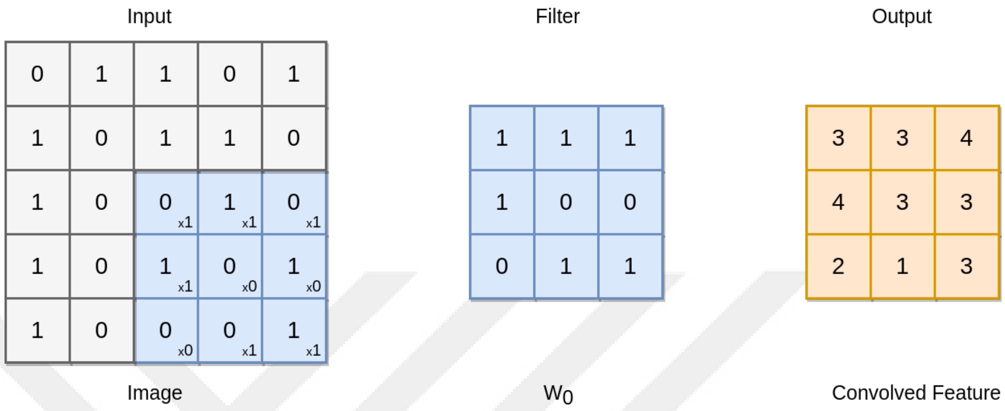
$$\begin{aligned}\text{\# of Total Parameters} = &(\text{filter\_height} \times \text{filter\_width} \times \text{filter\_depth} + 1) \times \\ &\text{filter\_number}\end{aligned} \tag{2.3}$$

The second main advantage is parameter sharing. With CNN, parameters that it is learned are shared through to next layer as input. Therefore the same weights are reused through the layer which means it is no need to learn it again in the layers. By that, further layers can learn more complex feature and patterns.

2.3.1 Convolution

In CNN, some filters act as learning parameters like *w* parameter in a multilayer neural network. They are mainly used in convolution processes. CNN is based on the processes of convolutions through the layers. Figure 2.7 demonstrates how convolution happens. In this example, the input is represented as a 5×5 matrix of values. There is a filter with the shape of 3×3. The filter is slid around the image in every position. Thus, the values of the filter are multiplied by the values of the image in that window size. It occurs a single number which represents all the values of the image in that window size.

Filters are also called as the kernel. They are generally square sized. In this example, the window slides by one pixel. It means the stride value equals 1. This value can be changed. Moreover, zero-padding is usually implemented around the image. It could keep the values of the corner and prevents the center-side values from becoming more dominant in the representation.



0	1	1	0	1
1	0	1	1	0
1	0	0 x1	1 x1	0 x1
1	0	1 x1	0 x0	1 x0
1	0	0 x0	0 x1	1 x1

1	1	1
1	0	0
0	1	1

3	3	4
4	3	3
2	1	3

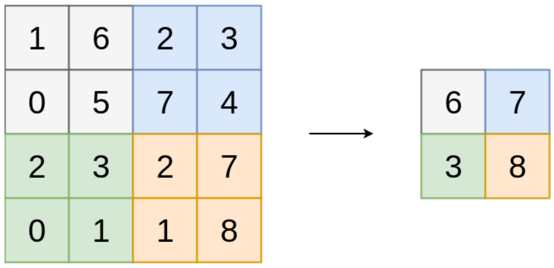
Figure 2.7. Convolution process

Convolutions and convolutional layers are used for filtering. Filters are updated continuously and learned by the model. They are the learned parameters like *w* in standard neural networks. They keep the patterns of an image. Various filters help the model to predict the content of an image. The output of the convolution generally has a smaller size than the original image. Equation 2.4 shows how to calculate the new size of a feature after the convolution process.

$$\begin{aligned} \mathrm{new\_height} &= (\mathrm{input\_height} - \mathrm{filter\_height} + 2 \times \mathrm{P})/\mathrm{S} + 1 \\ \mathrm{new\_width} &= (\mathrm{input\_width} - \mathrm{filter\_width} + 2 \times \mathrm{P})/\mathrm{S} + 1 \end{aligned} \tag{2.4}$$

2.3.2 Max pooling

Max pooling works like convolution. There is a square size window which can be called as a kernel. It slides over the image. However, the function isn't linear. It takes the highest value within the window. This value represents all values of the window. The main reason to use it to reduce parameters while trying to keep the most important values of an image. The process is shown in Figure 2.8 with the stride of 2.



1	6	2	3
0	5	7	4
2	3	2	7
0	1	1	8

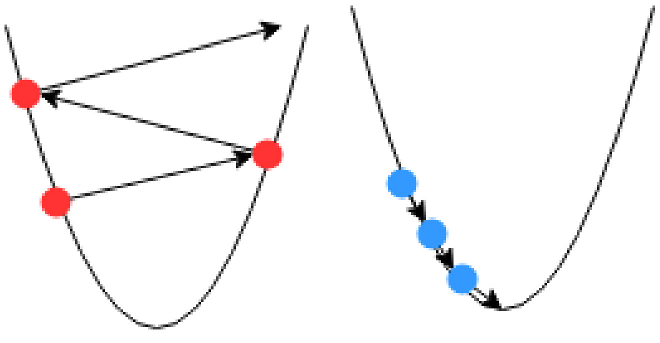
6	7
3	8

Figure 2.8. Max pooling process with the stride of 2

There is also another pooling method called Average Pooling. It takes the average of the window instead of taking the maximum value.

2.3.3 Hyper-parameters

A network model contains lots of hyper-parameters. These parameters can't be directly learned by training. Batch size, numbers of the epoch, number of hidden layers, size of filters, choosing activation functions, learning rates are some of the hyper-parameters which are needed be tuned throughout many iterations.



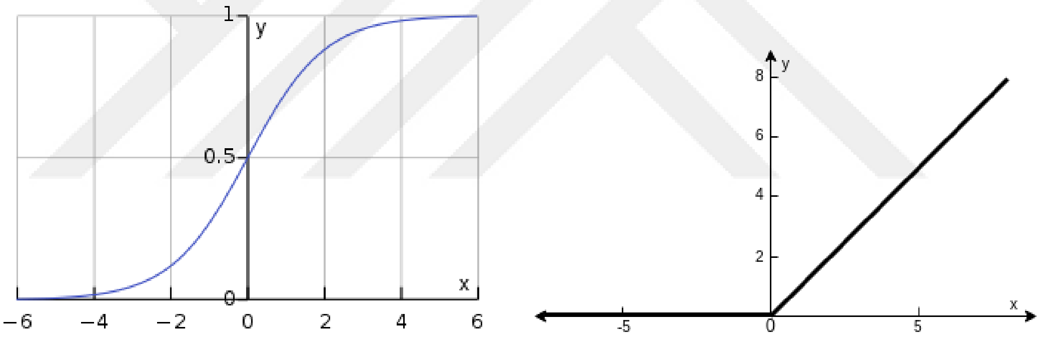
(a) Big learning rate **(b)** Small learning rate

Figure 2.9. Choosing the learning rate

For instance, choosing a learning rate is quite substantial. If it is picked a learning rate with a high number, it can overshoot the global minimum as shown in Figure 2.9. It may also take a long time to reach to the global minimum if it is selected a learning rate with a quite low number. For those reasons, various methods have been developed. Adaptive Moment Estimation (Adam) optimizer can be used as one of the best alternatives to change the learning rate adaptively. It is introduced in the following sections.

2.3.4 Activation functions

Another hyper-parameter is choosing the activation functions. It uses a function to pass through a value which maps the value into a specific range. There are some alternatives; however, **Rectified Linear Unit ReLU** is mainly preferred as a default activation function on many pieces of research.



(a) Sigmoid function [taken from (Wikipedia 2018)]

(b) ReLU function

Figure 2.10. Activation functions

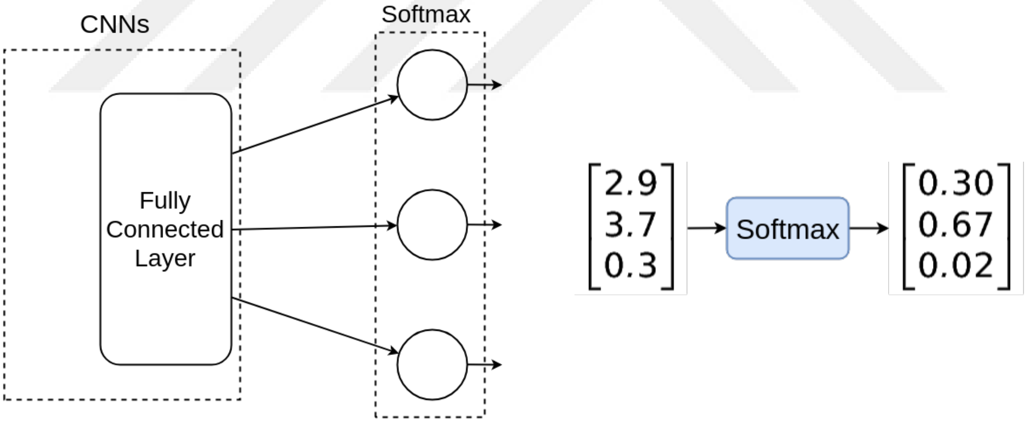
Figure 2.10 and Equation 2.5 shows the function of ReLU. If the number is negative, the output becomes a zero. Otherwise, the number keeps its value in the output.

$$A(x) = \max(0, x) \tag{2.5}$$

One of the main advantages is to prevent gradient vanishing. The other one is that it provides sparsity when the value lower than zero. It is computationally cheap to use the ReLU. It is one of the commonly used activation functions in deep learning. Its value doesn't entirely increase or decrease because values of itself and its derivative is monotonic. Sigmoid function can be shown in Equation 2.6. It isn't preferred since it tends to vanish the gradients when the input gets higher.

$$A(x) = \frac{1}{1+e^{-x}} \tag{2.6}$$

Softmax is a generalization of the Sigmoid activation function for multicategories. It takes the output of the last fully connected layer as input and maps them into probabilities in such a way that its sum equals to one. Thus, Softmax normalizes the outputs and make their sum equals 1. Especially for image classification; Softmax classifier can be used with the help of CNN. Figure 2.11 displays the expression.



(a) CNN model to Softmax classifier **(b)** Softmax probability mapping

Figure 2.11. Softmax classifier

Equation 2.7 gives the formula of Softmax function. While *K* refers to the total number of categories, *j* refers to only a single number among the categories.

$$A(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.7}$$

2.3.5 Zero padding

Figure 2.12 shows the implementation of zero-padding with size of two. Zero padding pads the input volume with zeros around the border. After implementing zero padding of two to the shape of 32×32×3, outcome reaches the shape of 36×36×3. It is one of the hyper-parameters of a network. By using this, input values of an image can be preserved with a higher chance. More importantly, it can be used to keep the height and width of an input same with the height and width of an output.

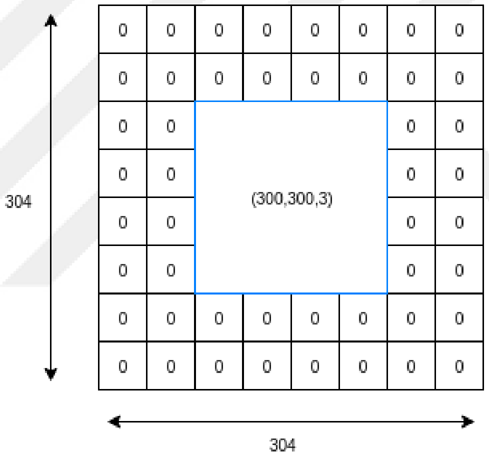
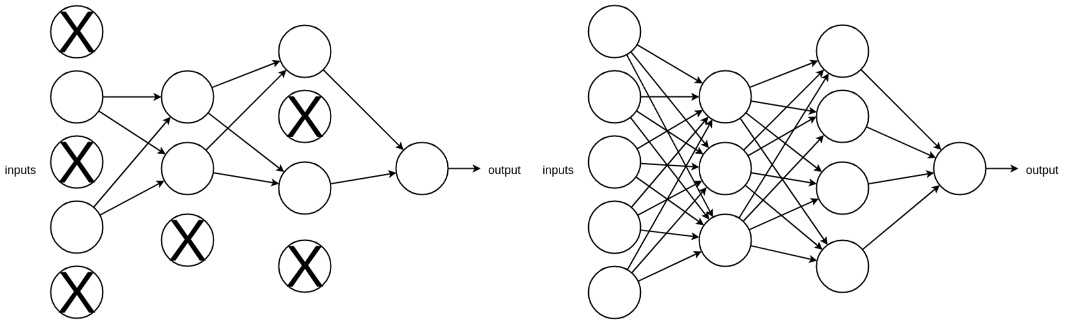


Figure 2.12. Zero padding

2.3.6 Dropout

Dropout (Srivastava et al. 2014) is one of the most used regularization techniques in deep learning. It prevents the model from over-fitting. It is first chosen as a probability value to keep the nodes. At each epoch in training, nodes are dropped out of the network with the probability of $(1-p)$. All connections that come to these nodes or go out from these nodes are also removed. In other words, nodes remain the same with a probability of p in every layer. In training, it is used with a certain probability value which generally equals to 0.5. However, the threshold should equal to 1 in testing. Figure 2.13 demonstrates the implementation of dropout.



(a) In training

(b) In test

Figure 2.13. Using Dropout in a model

Dropout encourages a model to become more robust by learning features from random situations. Dropout also occurs worse training loss error. However, it could also happen by using other regularization techniques. It is a trade-off between training performance and gaining more generalization. It is mainly used on the fully connected layers. It also may be used after the pooling layers.

2.3.7 Batch normalization

Typically, initial parameters are created and normalized so that it has zero mean and unit variance. However, training starts and the parameters are updated with a different range. Parameters which are created in this way occur to disrupt normalization. Consequently, the speed of training slows down and reaching to the global minimum gets harder. What batch normalization does is to re-normalize the weights for all batches in the layers in forward-propagation. Therefore, the combination of a bunch contains zero mean and unit variance. It also normalizes the updated weights after back-propagation. It deals with the outcome of high activation functions so that it reduces over-fitting. With the help of batch normalization, the learning rates with the high numbers can be used more commonly. Consequently, batch normalization allows the network to converge faster.

Besides the model becomes more robust for initializing the parameters. Thus, it helps with regularization (Ioffe and Szegedy 2015).

2.3.8 Adaptive moment estimation (Adam) optimizer

Choosing the learning rate is one of the vital hyper-parameters. Adam optimizer (Kingma and Ba 2014) is a method that computes adaptive learning rates for each parameter so that it changes throughout the training. Its power comes from using insights from Root Mean Square Propagation (RMSProp) optimizer and Momentum optimizer. RMSProp stores an exponentially decaying average of past squared gradients like v_t. Momentum keeps an exponentially decaying average of past gradients like m_t. Adam optimizer combines them. The decaying averages of past and past squared gradients m_t and v_t are computed in Equation 2.8. m_t is exponentially weighted moving average. β_1 and β_2 help to define the number of average to calculate. g_t is the value that comes from gradient descent. The initial values of m_t and v_t are equal to zero. Subscript t stands for time.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.8}$$

Values of m_t and v_t are needed to be tuned for bias correction as shown in Figure 2.9. It helps to get a better estimation result in the initial phases of the learning. $\hat{m}_t$ and $\hat{v}_t$ refer to the corrected values.

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.9}$$

Equation 2.10 shows the main formula of Adam optimizer. This formula also comes from RMSProp optimizer. ε prevents the errors from dividing by zero. α refers to a learning-rate value. In original paper, the authors propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ε. It generally outperform than other adaptive learning-method algorithms.

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.10}$$

2.3.9 Categorical cross entropy loss

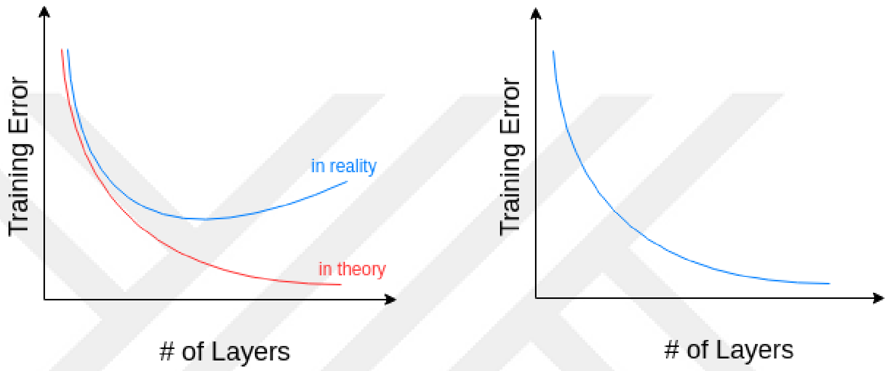
It is needed to have a loss of function to calculate the performance of the model for generalizing and prediction. Categorical cross entropy is one the most preferred loss functions. It is quite similar to Binary cross entropy. In Equation 2.11, J stands for the number of classes, N stands for the number of samples, $\hat{y}$ refers to the prediction of the model.

$$\mathscr{L}(w) = -\frac{1}{N} \sum_{i=0}^{N} \sum_{j=0}^{J} y_{ij}.log(\hat{y_{ij}}) + (1 - y_{ij}).log(1 - \hat{y_{ij}}) \tag{2.11}$$

Cross-entropy, in general, provides the model to measure the difference between predicted probabilities $\hat{y}$ and labels y. The model tries to change its parameters to decrease the cross-entropy loss. Minimizing cross-entropy loss allows the model to reduce the negative log-likelihood of the data. This loss function could work together with Softmax activation function which is introduced in the previous sections.

2.4 Residual Networks (ResNet)

From LeNet (LeCun et al. 1998) to ResNet (He et al. 2015) model, there has been many models that are proposed. They mainly suggest the changes in hyper-parameters like filter size, number of hidden layers, convolution, max-pooling, average-pooling, etc. The original paper of ResNet model shows that after reaching a significant amount of layers, the training error starts to increase. Although theoretically is not expected, it happens because of vanished or exploded gradients. As it is shown in Figure 2.14, the outputs of an activation function in a layer are multiplied by weights which connected to the next layer. Calculating this weights in a row in multiple time can cause those problems. For example, after random initialization of weights, some weights in a layer could have a large number. When multiplying this weight with other weights could cause a number to explode exponentially. The same could happen in vanishing which refers to reaching zero.



(a) Plain Network **(b)** ResNet

Figure 2.14. Training error performance between a plain network and a ResNet model (Ng 2017)

However, ResNet proposes a substantial change in logic adding by special connection among several layers. This model solves the problem of vanishing or exploding gradients. For instance, it remembers the input weights even if it vanishes or explodes on the next layers as shown in Figure 2.15. The main reason for this is because of the short-cut branch. It sums up with the weights which are applied to some layers. In the case of vanishing or exploding the weights of the main branch, it can still remember the weights that come from short-cut branch. ResNet is the winner of ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2015. It mainly uses batch normalization. Identity branch is also added after original work by same author (He et al. 2016). In the original paper of ResNet, it includes 150 layers in it.

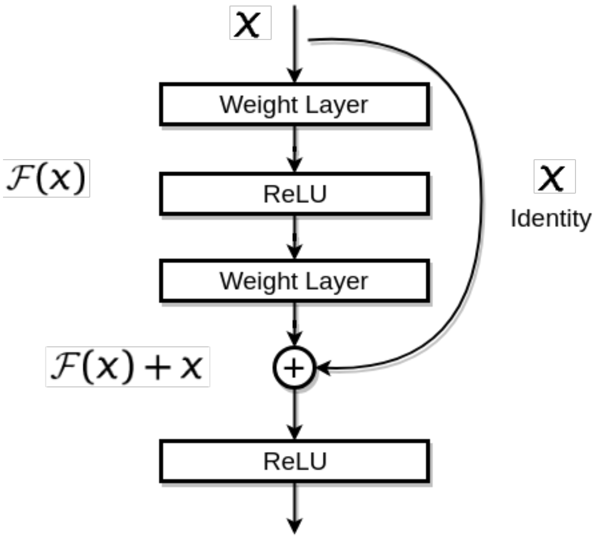
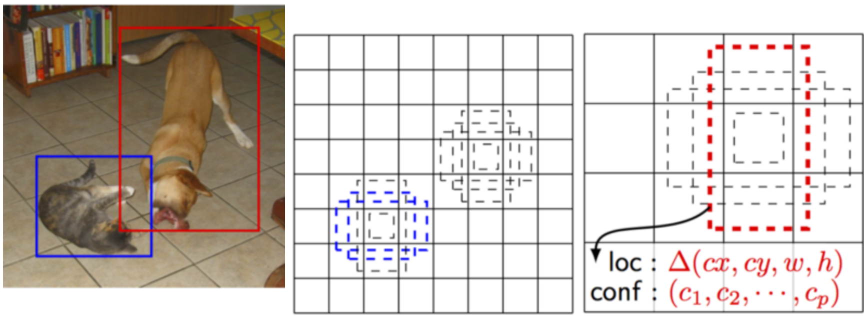


Figure 2.15. A ResNet block (He et al. 2015)

In this thesis, some tests are made to choose the number of the layer in the custom ResNet model. According to the practical results, a custom ResNet model with 30 layers is chosen. It is introduced in the following sections.

2.5 Single Shot MultiBox Detection (SSD)

Conventionally detection is made by the classification approach. However, this approach occurs some crucial problems. For instance, predicted bounding boxes generally don't fit the position of an object if the ideal bounding box is not a square. As an alternative solution, the detection is handled by the regression approach. Then, SSD is introduced. A custom multi-box bounding box system is developed in the SSD model using the previous works (Szegedy et al. 2014). In SSD, there are manually chosen default bounding boxes at various dimensions and aspect ratios. Those are associated with every feature map cell. By that, without a need for pre-training for the default box creation, SSD could become robust for generalizing of any input.


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

(a) Image with GT boxes

(b) 8×8 feature map

(c) 4×4 feature map

Figure 2.16. SSD framework [taken from (Liu et al. 2015)]

In traditional methods, SSD has its unique bounding box regression technique by using the multi-scale convolutional multi-box method (Szegedy et al. 2014). It includes many grid cells which can detect only one object in it. Every cell has default boxes with different aspect ratios, which can also be called anchor boxes, to detect the objects. In the figure that presented above, while cat matches with the default box in 8×8 feature map, dog matches with the default box in 4×4 feature map. The number of values can be calculated to pair with an object. Equation 2.12 shows the loss function which contains three parameters.

$$m_{loss} = c_{los} + \alpha \times l_{loss} \tag{2.12}$$

c_{los} refers to confidence loss. While it is used to categorical cross-entropy in the previous multi-box method, a simple softmax loss function is used to compute this loss in SSD. It produces a score for every object according to their chance of existence. l_{los} refers to location loss. While it is used to L2-norm in the previous multi-box method, a smooth L1-norm is used to compute this loss in SSD. Based on ground truth bounding boxes it calculates the closeness of predicted bounding boxes. L2-norm could help to predict perfectly shaped ground truth boxes. However, it is a trade-off between predicting excellently and performance. To balance the outcome of location loss α term is used. SSD architecture contains 8732 detections for every class at different feature map cell. Thus, the system becomes robust for detecting a various size of objects at multiple scales.

Intersection over Union is used to calculate the ratio between default boxes and the predictions. Default boxes are fixed size bounding boxes which roughly match the shape of the original ground truth boxes. It can also be used between ground truth bounding boxes and the predictions. Generally, default boxes are selected if *IoU* is greater than 0.5. It is shown in Figure 2.17.

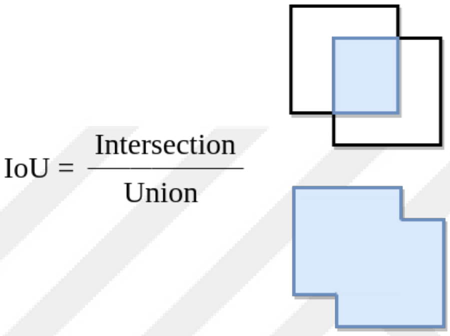


Figure 2.17. Intersection over union

Non-Maximum Suppression (NMS) is one of the helpful methods which is used in SSD (Hosang et al. 2017).

Algorithm 2.1 Non-Maximum Suppression

```
Require: IoU_threshold ← 0.5,
  probability_threshold_of_a_class ← 0.6,
  for every feature cell do
      make a prediction;
      if probability_of_a_class of feature cell ≤ probability_threshold_of_a_class then
        get rid of the boxes;
      end if
  end for
  choose the box with the largest probability_of_a_class;
  if IoU of chosen box ≥ IoU_threshold then
        get rid of the boxes;
  end if
```

One of the problems of object detection is that the algorithm could make multiple detections of the same object. Algorithm 2.1 ensure that Non-Maximum Suppression detects each object only once. In other words, the algorithm outputs the maximal probabilities of classification. However, it suppresses the close-by one that is non-maximal. There are two thresholds to filter bounding boxes that refer to the same object and select only the one which is most relevant. Confidence loss threshold eliminates the ones below than that score. This threshold could be 0.6 for instance. IoU threshold to eliminate the highest confidence score with others.

3 MATERIALS and METHODS

This section explains the methods that have been used in this thesis. They consist of the details corresponding to tools, environment, dataset generation, data annotation, pre-processing, classification model, detection model, experiments and proposed application. All the developments are made in Image Processing Laboratory and Computer Laboratory of the Computer Engineering Department at Uludağ University. Many computers are used simultaneously in the laboratories. The Graphics Processing Unit (GPU) card that is used is NVIDIA GeForce GT 730 with 2GB Random Access Memory (RAM). As a primary operating system, it is used Ubuntu 16.04LTS. Besides, cloud sources of Amazon and Google are also used for training. We use NVIDIA GRID K520 as a GPU with 4GB RAM in Amazon Web Service. We prefer to use NVIDIA Tesla K80 as GPU with around 12GB RAM providing by Google Colab since the training of SSD need potent computations. Moreover, Anaconda and Docker are used as virtual environments and containers respectively. Development of this thesis is written based on Python v3.5, and it is used Jupyter notebook as a development environment. Figure 3.1 shows how Jupyter notebook system works.

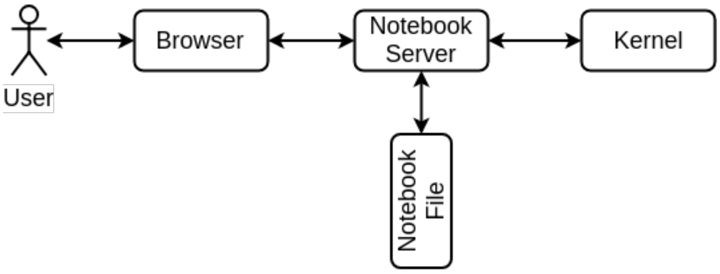


Figure 3.1. Jupyter notebook architecture

Open Source Computer Vision Library (OpenCV) version 3 is used for image processing, especially in data manipulation and augmentation purposes. It is worked with primarily TensorFlow version 1.3.0 and sometimes Keras version 2.1.3 as ML libraries. It supports to use GPU using Compute Unified Device Architecture (CUDA) acceleration. Such libraries are used for the development like scikit-learn, matplotlib, and numpy, etc.

3.1 Dataset Generation

This part explains the details about how all the images are gathered and become a dataset. All experiments are applied in this dataset. Thus, this dataset has a significant effect on the results. Images are collected through online sources using a script that acts as a web crawler. It is used some focused keywords to collect them class by class. Then, the ones that have inappropriate features such as showing the inside of the car, containing not the foremost part of the vehicle are eliminated. After collecting them class by class, they are manually labeled. The process of labeling manually takes quite a long time since it isn't used a method like Amazon Mechanical Turk.

Table 3.1 shows the distribution of the images in the dataset which includes seven classes. First six classes indicate Volkswagen Passat 1.6 TDi BlueMotion Comfortline 2015, Renault Fluence 1.5 dCi Touch 2016, Fiat Linea 1.3 Multijet Active Plus 2013, Volkswagen Polo 1.6 1999, Renault R12 Toros 2000, Fiat Dogan SLX 1996. Classes have numbers of images respectively 4024, 4293, 4234, 3208, 3783, 4183, 4162. Seventh class is composed of seven different make and models, except then the first six classes, for representing other cars. The shapes of the images vary from 300 to 600 pixels in both height and width. The motivation to select those make & models are related to statistical works of Turkish Statistical Institute (TurkStat) (Institute 2017). The Institute monthly states the number of vehicle brands that are registered to the traffic in that specific month in Turkey. It is made research for the reports released within the last two years. Finally, it is sorted by the number of vehicles that brands have.

Since there are limited time and human resources, it is needed to be chosen only some brands. Volkswagen is selected since it is on the top of the list that it is formed according to reports of TurkStat. Taking into account that Renault and Fiat have a manufacturing company in Bursa, where the Uludağ University is located, and they are also in the top 5th list; they are chosen to the dataset. It is also needed to indicate that Fiat Dogan SLX and Renault R12 Toros are produced in Turkey and mainly used in the country. So, they are mostly local models of their makes.

The statistical data of TurkStat is also taken into account for forming seventh class. It stands for make & models of other vehicle brands except than Volkswagen, Renault, and Fiat; Thus, all of these features make this dataset more country-specific. The dataset contains 27887 images of the vehicles.

Table 3.1. Class distribution of the dataset

Make	Model	Year	Feature	# of Images
Volkswagen	Passat	2015	1.6 TDi BlueMotion Comfortline	4024
Renault	Fluence	2016	1.5 dCi Touch	4293
Fiat	Linea	2013	1.3 Multijet Active Plus	4234
Volkswagen	Polo	1999	1.6	3208
Renault	Toros	2000	R12	3783
Fiat	Dogan	1996	SLX	4183
Other Class				4162

Table 3.2 states the distribution of the seventh class. It is composed of seven different make & models, except then the first six classes. It refers to the other cars in general.

Table 3.2. Inner distribution of the other class in the dataset

Make	Model	Year	Feature	# of Images
Toyota	Corolla	2016	1.4 D-4D Advance	663
Volvo	S60	2014	1.6 D Premium	707
Peugeot	206	2001	1.4 XR	468
Ford	Focus	2017	1.6 TDCi Trend X	693
Mercedes-Benz	C	2015	CLA 180d	608
Nissan	Micra	2016	1.2 Match	533
Audi	A3 Sedan	2017	1.6 TDI	490

Data annotation is essential regarding detection purposes. It generally requires lots of time to be done. Algorithm 3.1 is followed to decrease the annotation time. It defines the GTBB and classes of the images from the dataset with the help of predicted outcomes of pre-trained SSD model. In this case, we assume that the images usually include a vehicle which is larger than a certain size. Annotation takes a work day long when this algorithm is implemented in our dataset.

Algorithm 3.1 Annotating ground truth bounding boxes and classes

```
Require: certainSize ← a threshold value,
  classId ← zero,
  for all images of that class do
      read the image;
      pass it through the pre-trained SSD to detect only cars;
      carSize ← the size of the detected car;
      if carSize $\geq$ certainSize then
          classOfDetectedCar ← classId;
      else
          ask annotator to give a label or delete it;
      end if
       save the annotation to a .csv file;
  end for
  increase the classID by one if any and run again;
```

A script is written to implement this algorithm. The images in dataset stay under the folder of its related make & model of a class. The script takes only one folder of a class so that this help for automating. For example, it starts labeling with Fiat Dogan and gives an input to script assuming that the object as a class of Dogan. If the detected object higher than a certain threshold, it is accepted as a class automatically which is Fiat Dogan in that case. If the detected object is not big enough, ask for input the user. There are two main possible in that case. Firstly user can give a specific input saying this is a class of possible classes. On the other hand, the user can provide another individual input stating that omit the object not to save it.

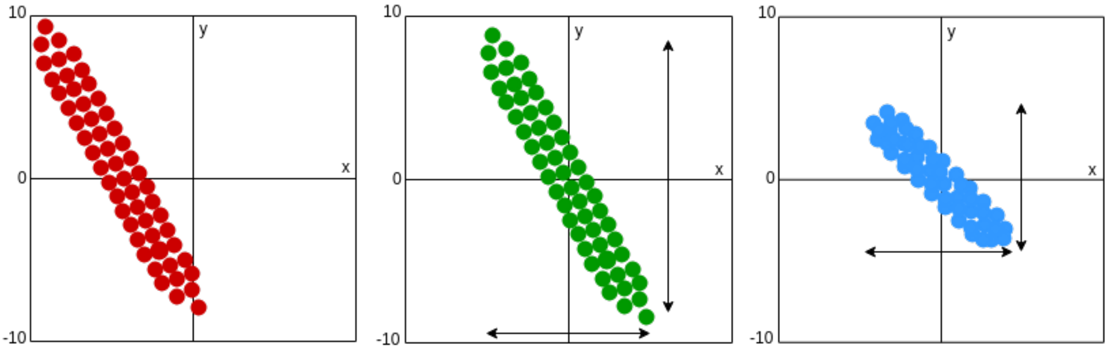
3.2 Pre-processing

There are plenty of pre-processing methods to make the data appropriate for the Neural Network. In this thesis, the data is normalized first. Then it is re-sized by adding zero padding. More importantly, data augmentation is also implied in it to reduce over-fitting.

Normalization of data by having zero mean and equal variance is essential. Calculating the mean and subtracting it per channel for every image serves to center the image at around zero-mean. By that, each image and feature have a similar range, so that prevents the gradients from exploding or shrinking. Since the gradient has a uniform distribution for each channel, it helps to learn faster. It also makes the model more robust on illumination changes. Equation 3.1 represents the Figure 3.2.

$$x' = \frac{x - \mu}{255} \tag{3.1}$$

In the thesis after calculating the mean of all channels, it is subtracted from the value. Since every pixel has ranged from 0 to 255, it is re-scaled to 0 between 1 by dividing 255.

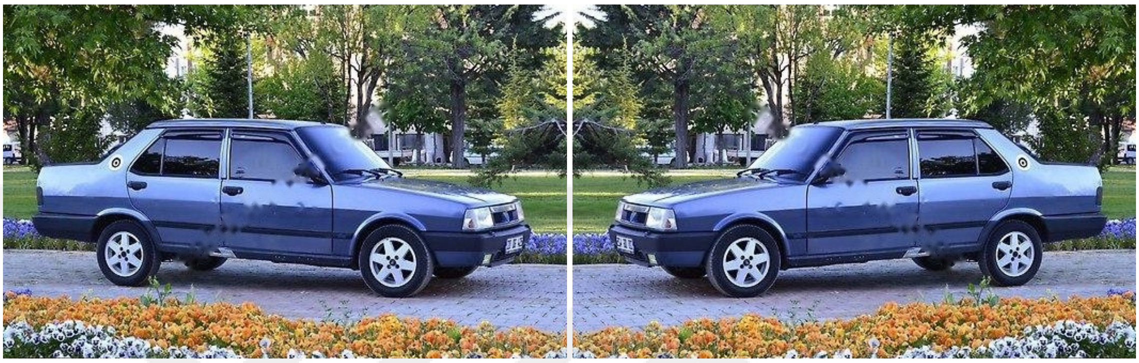


(a) Original data **(b)** Zero-centered data **(c)** Normalized data

Figure 3.2. Normalization of the data (Fei-Fei Li 2017)

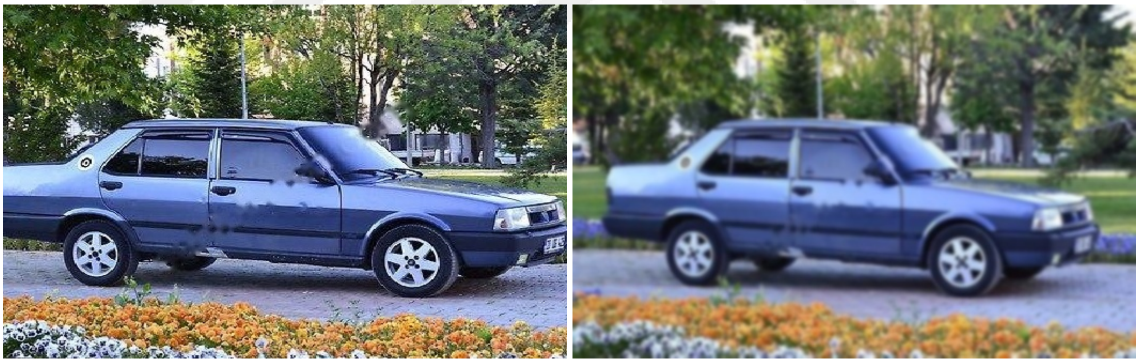
Data Augmentation

Because the dataset doesn't include a vast amount of data, it is necessary to make data augmentation with specific techniques. The data augmentation could prevent the model from over-fitting and encourage the model for performing a better generalization. Since the model could deal with various type of data, this process is quite crucial for the model to achieve its best in real-world problems.



(a) Gauss Noise

(b) Horizontally Flip



(c) Zoom In

(d) Gauss Blur

Figure 3.3. Some data augmentation techniques

Figure 3.3 shows some of the methods of the data augmentation. In the first example, it is added Gauss noise in such a way that it has zero mean and its sigma and variance equals 1. It is hard to understand by only observing; however, its pixels are slightly changed. In the second sample, it is merely flipped horizontally. In the third example, it is zoomed in with an absolute value. It is vital to keep the aspect ratio same here. On the last instance, it is added Gaussian Blur to simulate focusing problems of the cameras. OpenCV is used for implementation of data augmentation techniques.

Padding and Resize

Maintaining aspect ratio of the images is crucial when it comes to applying the result in real-world cases. The images in our dataset generally have rectangular shaped images. While their heights vary from 300 to 400 pixels, their widths range from 500 to 600 pixels. It is crucial to make them square size to make convolutions more efficient. The SSD model requires (300,300,3) shape images. Therefore, it is needed to add padding and re-sizing to the desired shape.

In our algorithm, it is first made a comparison to find whether height or width is smaller than the other. After noticing the smaller one, it is added the zero padding to that side so that the image becomes a shape of a square. Then, it is re-sized to the desired size. In this work, the images are all re-sized it to the form of (300, 300, 3) for all experiments so that it could be able to see the differences among all different models. Figure 3.4 displays a sample image from the dataset after adding padding on it.

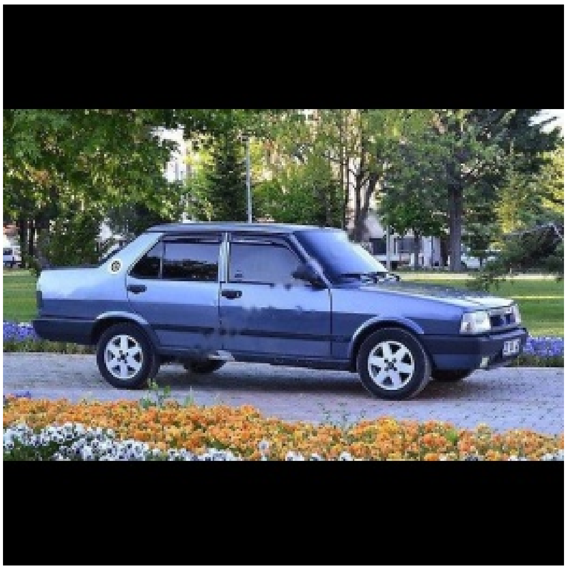
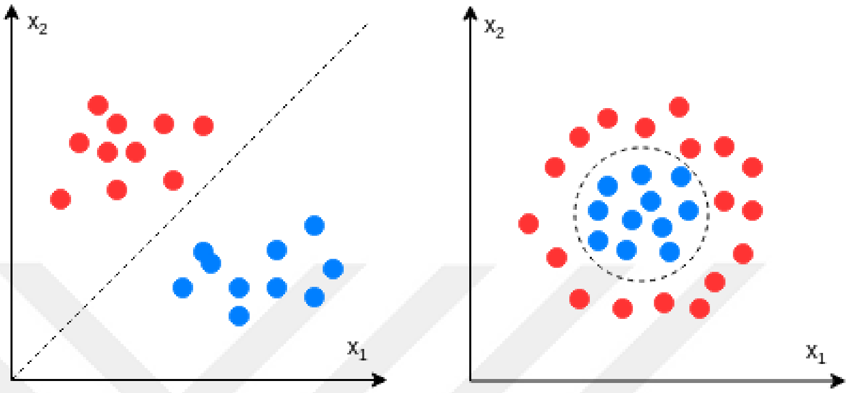


Figure 3.4. Adding zero-padding to an image

3.3 Vehicle Classification and Detection

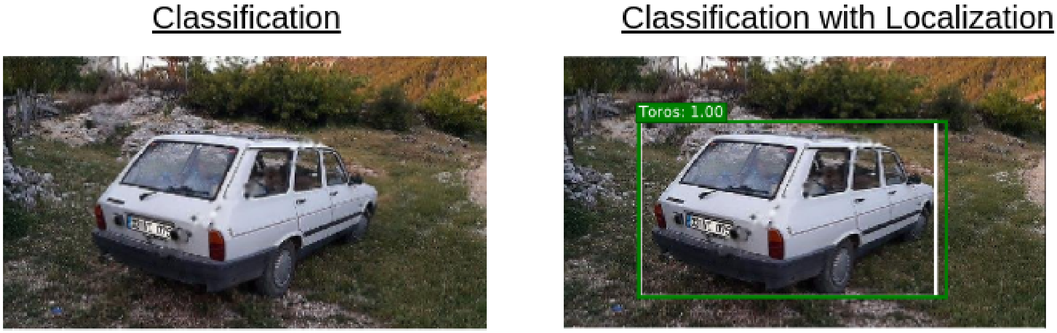
Figure 3.5 gives general intuition about linear and non-linear classification which can be used in object classification tasks. In this study, non-linear classification is made by Softmax layer which is introduced in the previous sections.



(a) Linearly separable **(b)** Linearly non-separable

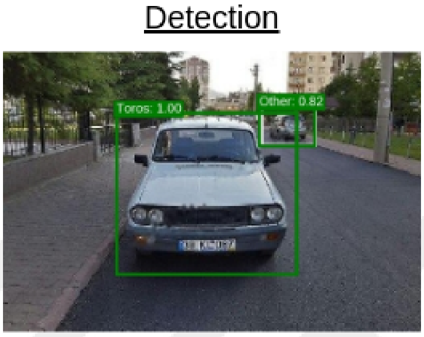
Figure 3.5. Linear and non-linear classification

Figure 3.6 demonstrate the differences among classification, localization, and detection. Classification represents whether if an image contains a certain object in it at somewhere. In the context of the thesis, it can be said that image classification is responsible for declaring whether the image contains a certain model of a vehicle. However, classification with localization is not only responsible for declaring if the image contains a specific model of a vehicle but also is responsible for drawing a bounding box around the position of it in the image. In other words, it is responsible for figuring out where the object is located in the picture. But, this can happen only for one object in an image. Detection is defined as having multiple objects which are implemented classification with localization on them. Therefore, it might be various objects in the picture, and it can be detected all. As a result, classification and classification with localization problems have one object in the middle of the image which is supposed to be recognized. In contrast, the detection problem deals with the multiple objects of different categories within a single image.



outcome: Renault Toros

outcome: Renault Toros + It's BB



outcome: Renault Toros + It's BB
Other Class + It's BB

Figure 3.6. Differences among classification, classification with localization and detection

The output of classification could be shown in Equation 3.2 corresponding to the context of the thesis. As an outcome, every class has its probability. The sum of the probabilities equals one when applying the softmax activation function to them. *class* stands for one of the seven categories which are introduced in Table 3.1. It is assumed that the image only has one object in the picture. *prob* shows how accurate the prediction of a class object. Probability can be in a range between 0 and 1.

$$y_{predict} = \Big[[class_1 : prob_1], \quad [class_2 : prob_2], \quad ..., \quad [class_7 : prob_7]\Big] \tag{3.2}$$

The output of classification with localization is shown in Equation 3.3. x_{min} and y_{min} refer to the most up-left point meaning to (0,0). x_{max} and y_{max} refer to the most down-right point meaning to their max pixels.

$$y_{predict} = \begin{bmatrix} prob, & class, & x_{min}, & y_{min}, & x_{max}, & y_{max} \end{bmatrix} \tag{3.3}$$

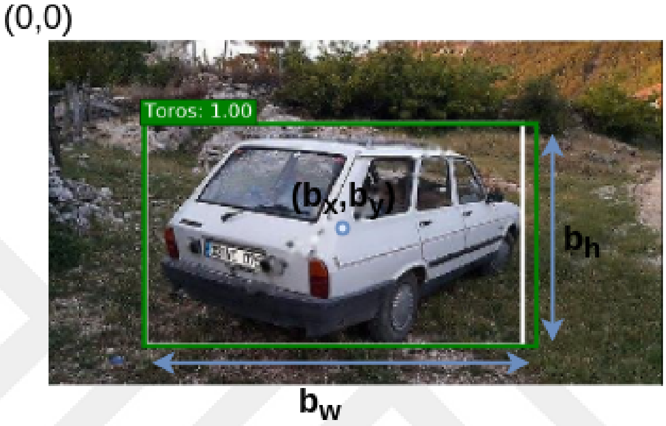


Figure 3.7. Showing a bounding box

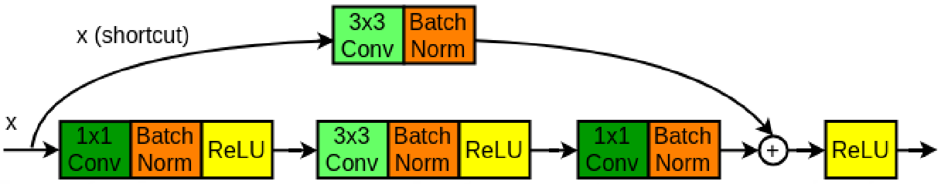
In detection, the models generally accept the images with GTBB in different notation. Besides, the image can contain multiple objects. Detection is just multiple ways of classification with localization. The output of detection is shown in Equation 3.4.

$$y_{predict} = \left[\underbrace{[prob, class, b_x, b_y, b_h, b_w]}_{\text{object 1}}, \underbrace{[...]}_{\text{object n}} \right] \tag{3.4}$$

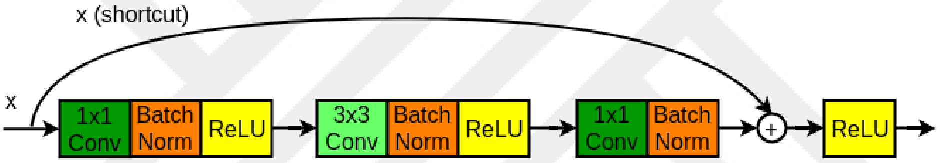
b_x and b_y represent the middle points of the car, as well as height b_h and width b_w of the bounding box: Values of b_h, b_w can vary from 0 to 300 referring to pixels. In the thesis, a custom designed ResNet model is used which is shown in Figure 3.8. It has 30 layers and is used in experiments of the thesis. Experimental details are introduced in the following sections. This model contains trainable parameters with the number of

1132775. In convolutional sections of Identical block implement (1,1) as a stride value. Therefore, the shapes of features preserve their size of heights and widths. The filter sizes also preserve their values. Equation 3.5 shows the output of the block.

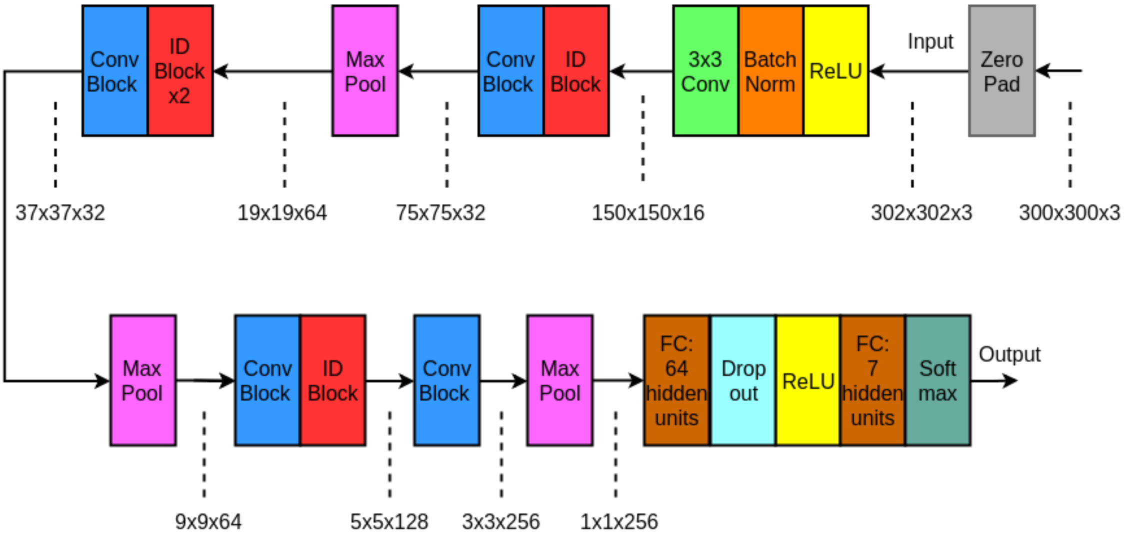
$$y_{identity\_block} = F(x, W_i) + x \tag{3.5}$$



(a) Convolutional block



(b) Identical block



(c) Main ResNet model

Figure 3.8. Custom ResNet architecture

The first section in the convolutional block has a stride of (2,2). Shortcut section contains the same stride value as well. Other sections include a stride of (1,1). Moreover, the first and second section of the main block has the same value of filter size. The third section on the main block has the equal value of filter size with the shortcut section. Equation 3.6 shows the output of the block.

$$y_{conv\_block} = F(x, W_i) + W_s x \; . \tag{3.6}$$

Detection model is shown in Figure 3.9 which shows the architecture of the SSD model. It contains a series of convolutional blocks. Detections are made on certain layers which equals to 8732 per class. After predicting these, an NMS method is implemented to have the most stable outcome.

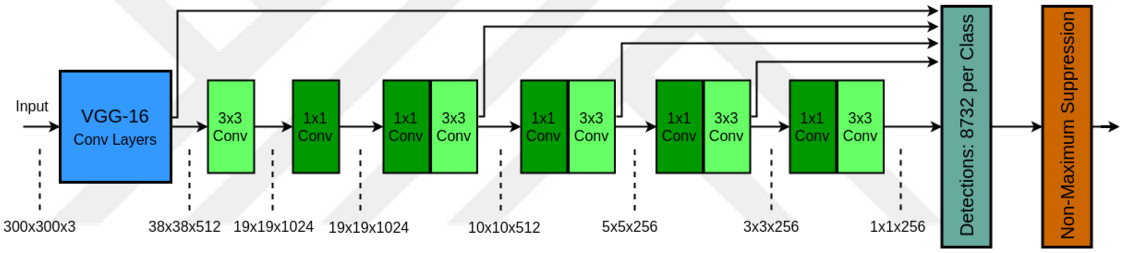


Figure 3.9. The architecture of SSD Model for Detection

It is originally used the same model with the original paper. The only difference is either it is used with the custom ResNet model or fine-tuned on our dataset. In this thesis, an implementation of the original paper is used for detection purposes (Ferrari 2017).

3.4 An application

This thesis also proposes an application which could detect the fraud on the license plates of the vehicles. The process of the application can be explained as follows. At first, reading and recognizing the license plate numbers is done by an open source project (Dahms 2016). Simultaneously, the proposed make & model classification method is used to predict the class of the vehicle. Both of them are transferred to the middleware. The middleware makes a pair of them and assigns them to a dictionary value. It is composed of a key value. As a result, a comparison is made with the contents of the dataset. In this application, the dataset is set manually at first. Every license plate number is matched with the class of a vehicle in the dataset. In this use case, fraud is detected if the license plate numbers are matched however make & models of a car not. It can be referred that the license plates are repetitive and the detected vehicle has an unauthorized license plate. Figure 3.10 explains how it works as a diagram.

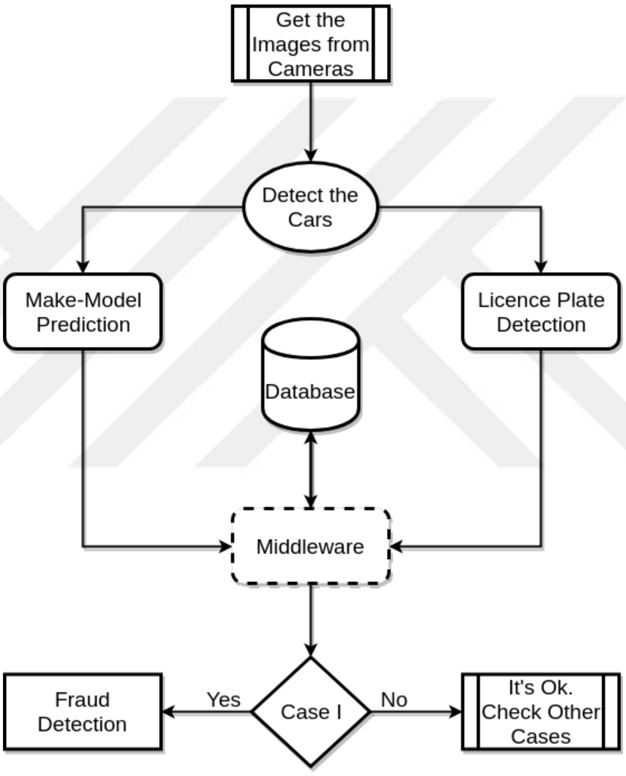


Figure 3.10. A use case diagram, Case I: license plates match however models don’t

4 RESULTS

Three main experiments are implemented in the thesis. Figure 4.1 demonstrates the differences among them. They make classifications using a custom ResNet model only, a pre-trained SSD with the ResNet model and a fine-tuned SSD only respectively. There is the main difference between Experiment I and II. The model gathers the images, then implements the pre-processing methods on them in Experiment I. Like it is shown in previous sections; normalization, zero padding, re-sizing and data augmentation are the main parts of pre-processing methods. In the pre-processing section, re-sizing to the shape of (300,300,3) is implemented on the images firstly. Later on, data augmentation is followed. It consists of flipping, adding Gaussian blur, adding Gaussian noise and zooming. Finally, the images become ready for being fed into the model for training.

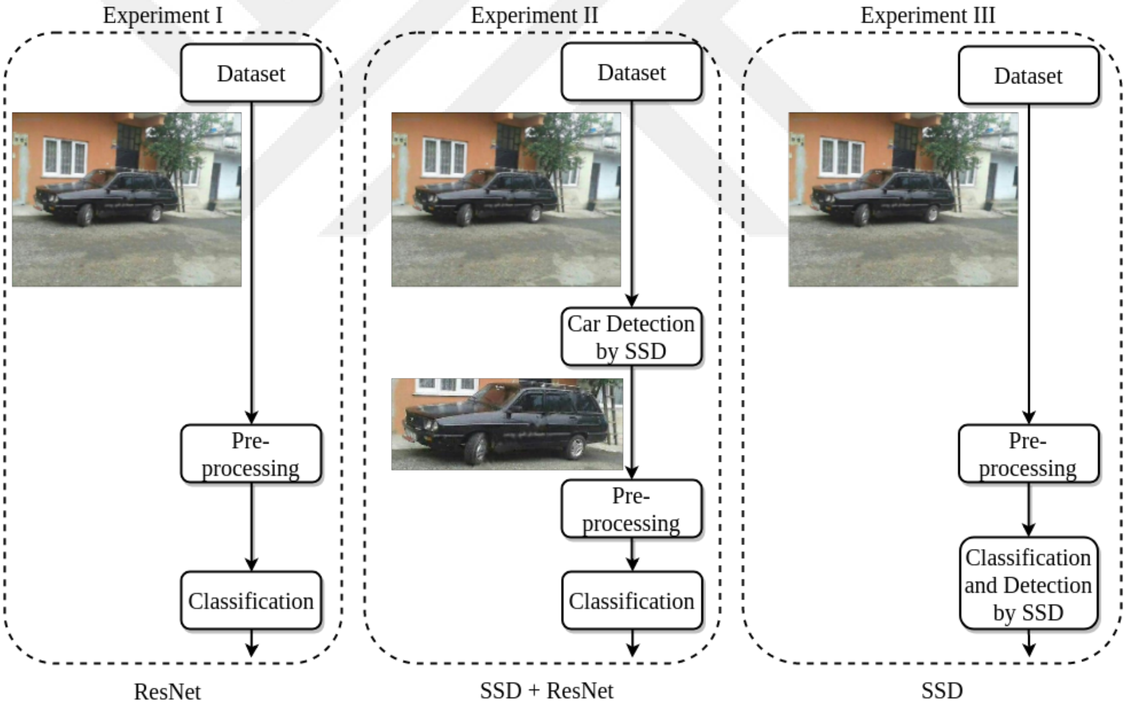


Figure 4.1. Overview of Experiments

The images are processed through an SSD model to only detect vehicles in Experiment II. The weights of the SSD model firstly pre-trained on MS COCO dataset. Then, they fine-tuned on PASCAL VOC07 & VOC12 dataset. After detecting vehicles, the same pre-processing methods are implemented on the images of that vehicles. As a result, the same ResNet model gathers that images for training. Thus, only the vehicles are given to the model in Experiment II instead of feeding the whole image to the model. In Experiment III, it is only used in an SSD model. However, the images are labeled with the help of the detection part of Experiment II. The same GTBB is used for every image. The same pre-processing methods are applied. This experiment aims to make detection as well as classification. The results of the experiments are introduced in the experimental result section.

This section also explains all the results of experiments which have been done on a test set in our dataset. Table 4.1 introduces all the conclusions of the analyses. Also, loss graphs, accuracy graphs, and confusion matrices express the detailed results. Besides, specific outcomes of the implementation of an experiment on test videos are shown. It is implemented 80%-10%-10% rule on training-validation-test sets distribution in Experiment I and II. As a GPU power, it is used NVIDIA GT 730 with 2GB RAM for Experiment I and II. However in Experiment III; it is implemented 80%-20% rule on training-test sets distribution. As a GPU power, Tesla K80 with the help of Google Colab is used for Experiment III since it needs a lot more computation than the other experiments. It is seen in the table that the result of the classical model in Experiment I and the result of the proposed model in Experiment II reach 91.27% and 95.10% classification accuracy scores respectively. Since the only difference between Experiment I and II is to have well-centered detected vehicles, it can be referred that this feature could help to the model to increase classification accuracy significantly.

Furthermore, it is seen that the result of the proposed model in Experiment III reaches slightly close to the achievement of the classical model in Experiment I even without using not perfectly shaped GTBB. Besides, it is also seen that a lot more data is needed to reach higher classification accuracy results by taking into consideration of train score in Experiment III. It can be referred from the significant difference between train and validation score that it is needed more data and a bit of change to extend in the model.

Table 4.1. Comparison of the experimental results

	Experiment I		Experiment II		Experiment III	
Method	ResNet		SSD + ResNet		SSD without fine-tuning	SSD with fine-tuning
Batch Size	32				32	
Epoch	128	96	128	96	29	34
Loss	Categorical Cross Entropy				Smooth L1 + Softmax	
Image Size	224	300	224	300	300	
Train score	0.9692	0.9635	0.9853	0.9836	0.9071	0.8916
Valid score	0.9185	0.9052	0.9394	0.9376	0.9057	0.8852
Test score	0.9239	0.9127	0.9524	0.9510		

4.1 Experiment I: Classification by a ResNet Model

The pipeline is implemented with the images which have the shape of (224,224,3) and (300,300,3) respectively. At first, the experiment is done with the images that have the shape of (300,300,3) to match the shape with the SSD model in Experiment III. Later, it is made with the images that have the shape of (224,224,3) to see if re-sizing would hurt the accuracy. As a result, re-sizing even shows a better result. It means that the model could obtain the essential features and generalize even if the shape of the image is re-sized to down. Figure 4.2 and 4.3 show the loss graph and accuracy graph of this experiment **with 224×224 sized images**. The number of 4 in the epoch line refers to 128 originally. It is seen that the more accuracy reaches a higher score or loss reaches a lower score, the more the lines reach to plateau.

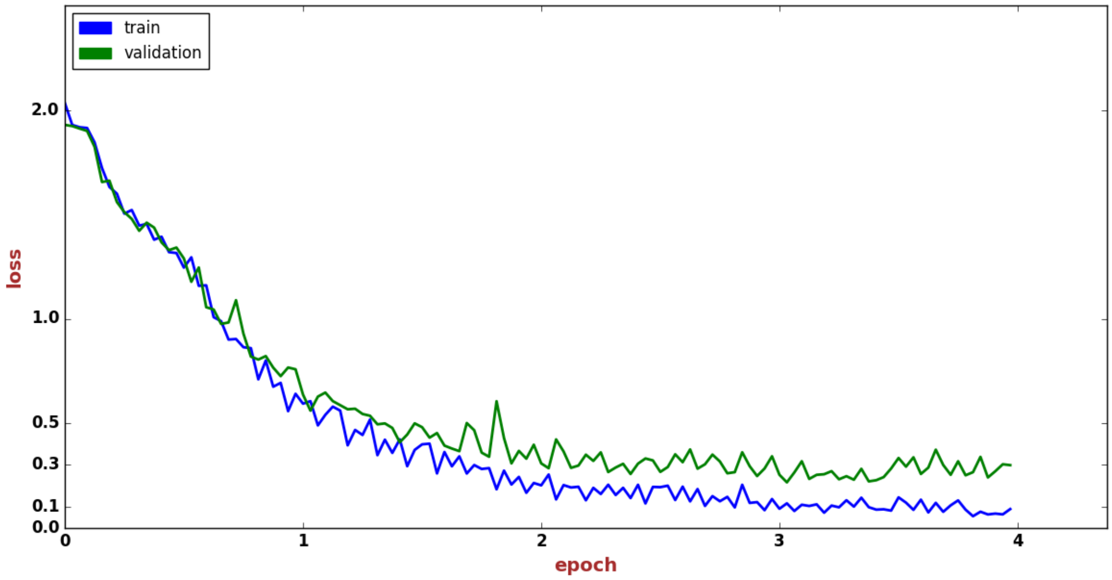


Figure 4.2. Loss graph of Experiment I on (224,224,3) shaped test set

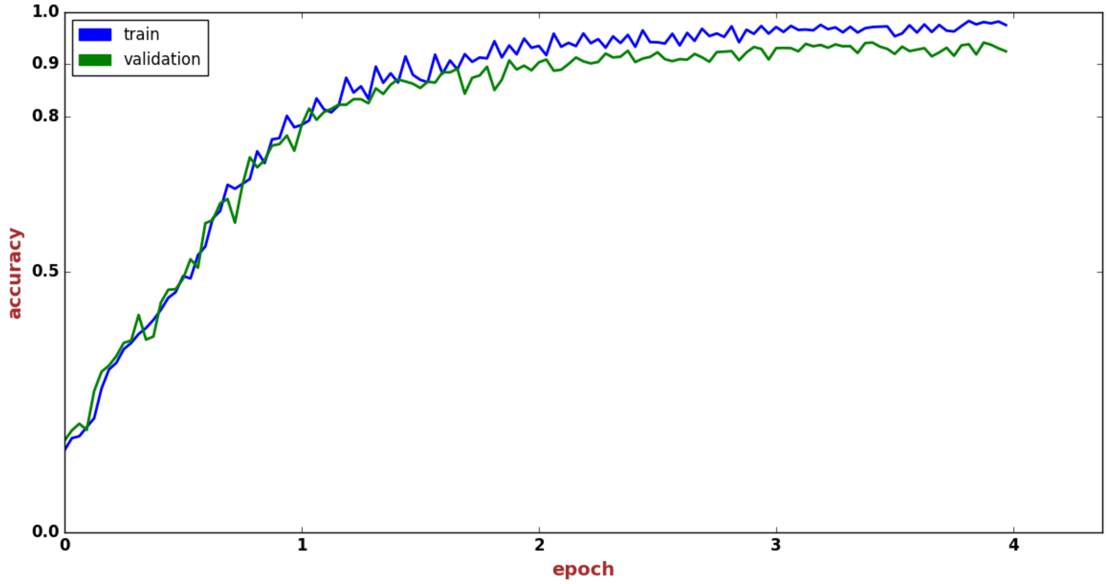


Figure 4.3. Accuracy graph of Experiment I on (224,224,3) shaped test set

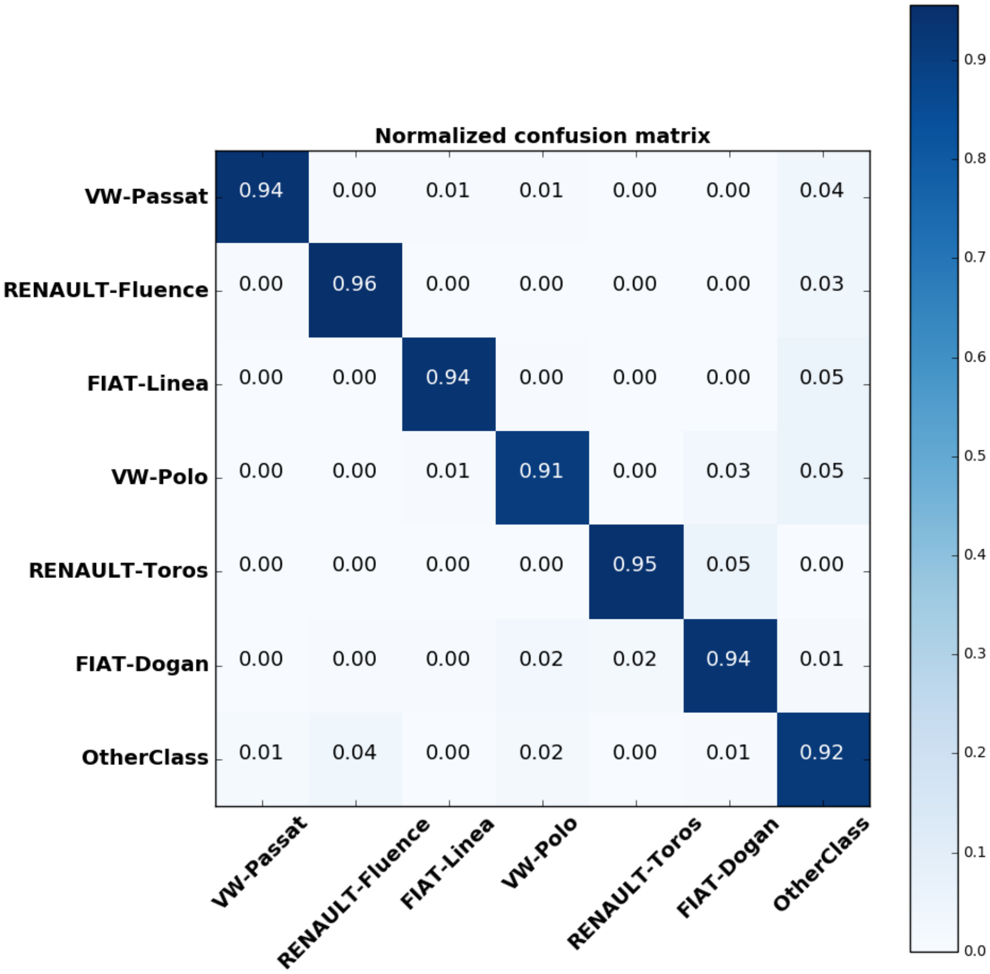


Figure 4.4. Confusion matrix of Experiment I on (224,224,3) shaped test set

Figure 4.4 displays the confusion matrix of this experiment. It reaches 93.71% overall accuracy result on the test set. It also reaches over 90% score for all classes. However, since the number of images of VW-Polo class is lower than the others in the dataset could affect the score of VW-Polo class. Figure 4.5 and 4.6 show the loss graph and accuracy graph of this experiment **with 300×300 sized images**. Like the result of the previous experiment, the scores reach a plateau after some epochs.

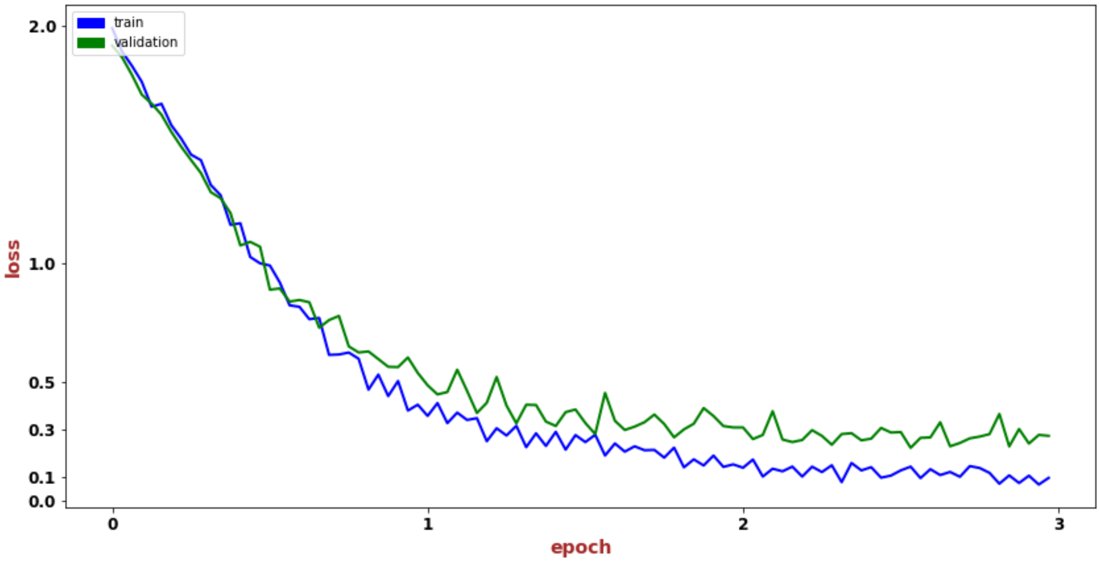


Figure 4.5. Loss graph of Experiment I on (300,300,3) shaped test set

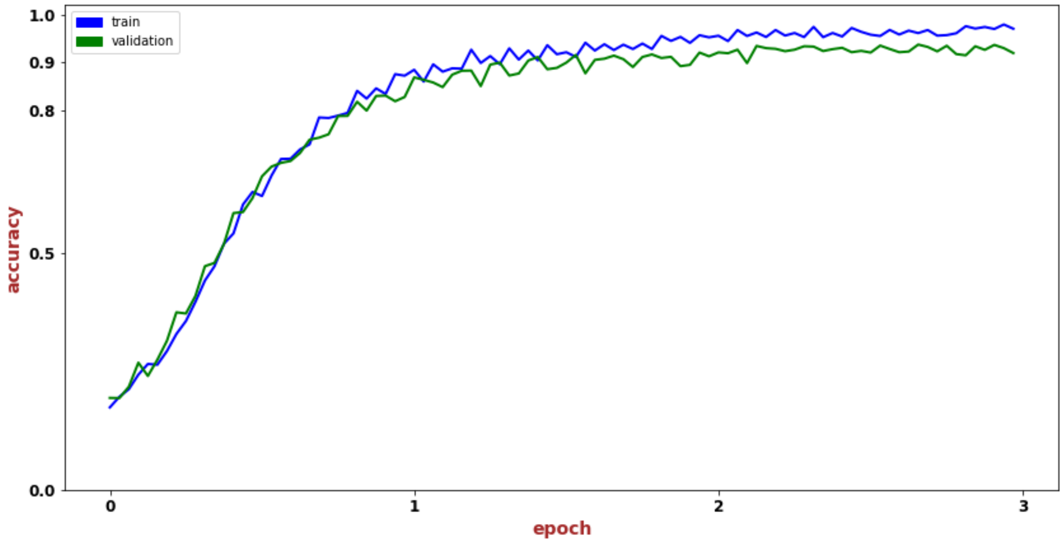
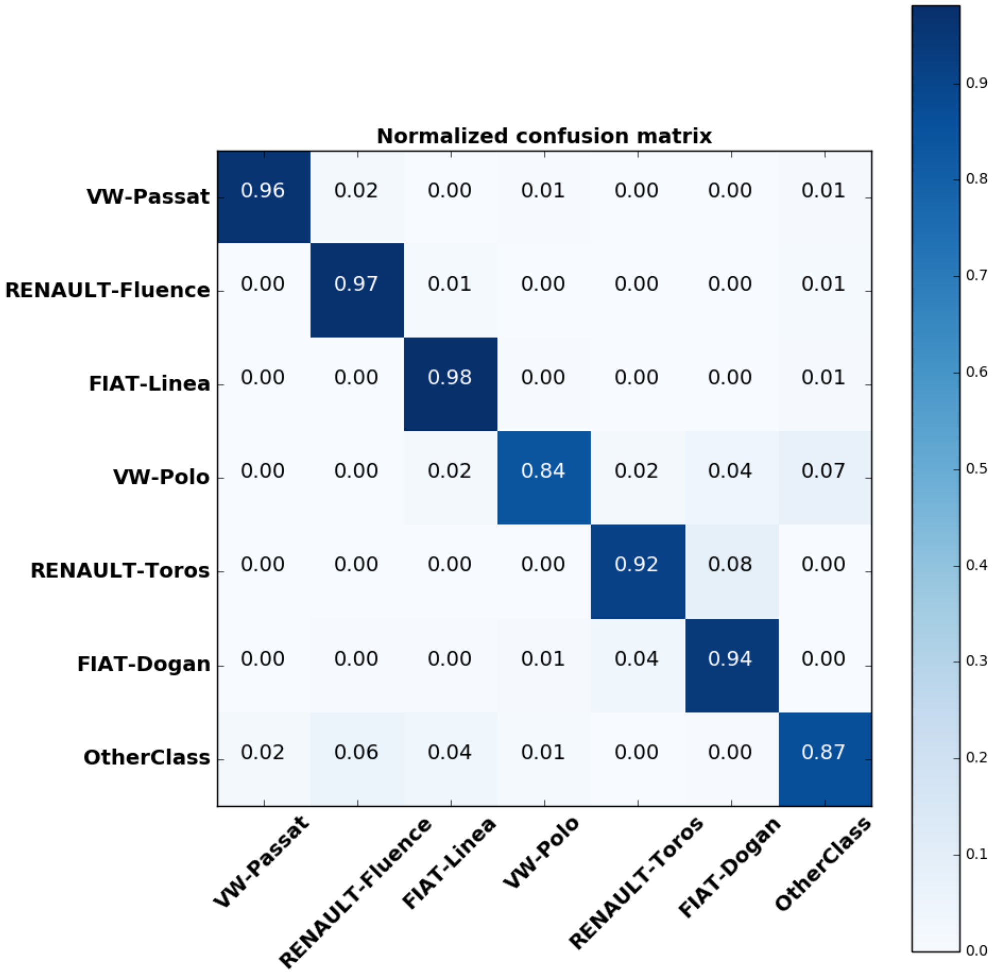


Figure 4.6. Accuracy graph of Experiment I on (300,300,3) shaped test set

Figure 4.7 displays the confusion matrix of this experiment. It reaches 91.27% overall accuracy result on the test set. It can be seen that it reaches a quite low score to generalize the other class as well as the class of VW-Polo. Increasing the number in the sub-classes of the other class could help to increase this particular accuracy.



Normalized confusion matrix

	VW-Passat	RENAULT-Fluence	FIAT-Linea	VW-Polo	RENAULT-Toros	FIAT-Dogan	OtherClass
VW-Passat	0.96	0.02	0.00	0.01	0.00	0.00	0.01
RENAULT-Fluence	0.00	0.97	0.01	0.00	0.00	0.00	0.01
FIAT-Linea	0.00	0.00	0.98	0.00	0.00	0.00	0.01
VW-Polo	0.00	0.00	0.02	0.84	0.02	0.04	0.07
RENAULT-Toros	0.00	0.00	0.00	0.00	0.92	0.08	0.00
FIAT-Dogan	0.00	0.00	0.00	0.01	0.04	0.94	0.00
OtherClass	0.02	0.06	0.04	0.01	0.00	0.00	0.87

Figure 4.7. Confusion matrix of Experiment I on (300,300,3) shaped test set

4.2 Experiment II: Classification by a ResNet + an SSD Model

This pipeline also is implemented with the images which have the shape of (224,224,3) and (300,300,3) respectively. Figure 4.8 and 4.9 show the loss graph and accuracy graph of this experiment **with 224×224 sized images**. With 224×224 sized images, it is needed to train with higher epoch number to reach better scores.

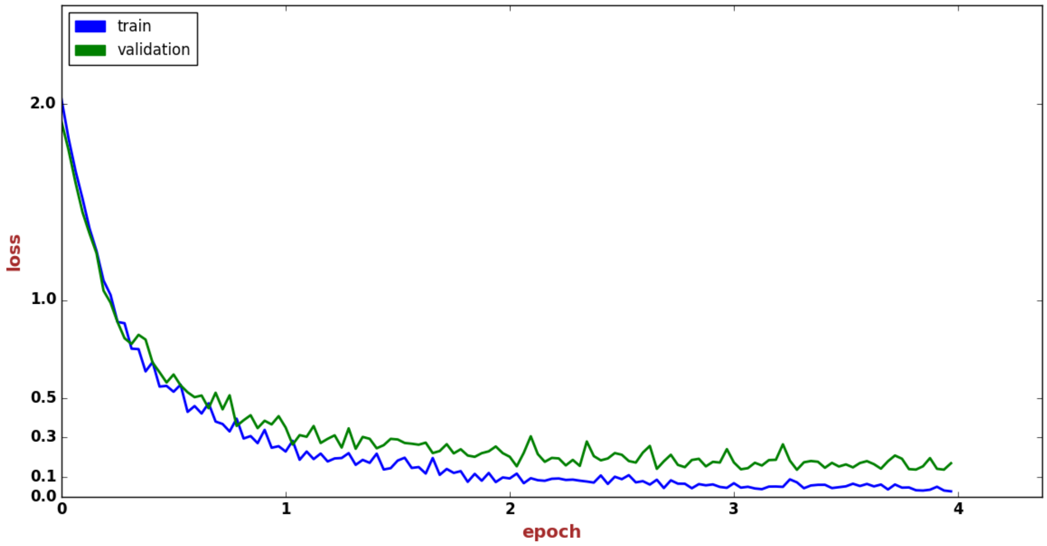


Figure 4.8. Loss graph of Experiment II on (224,224,3) shaped test set

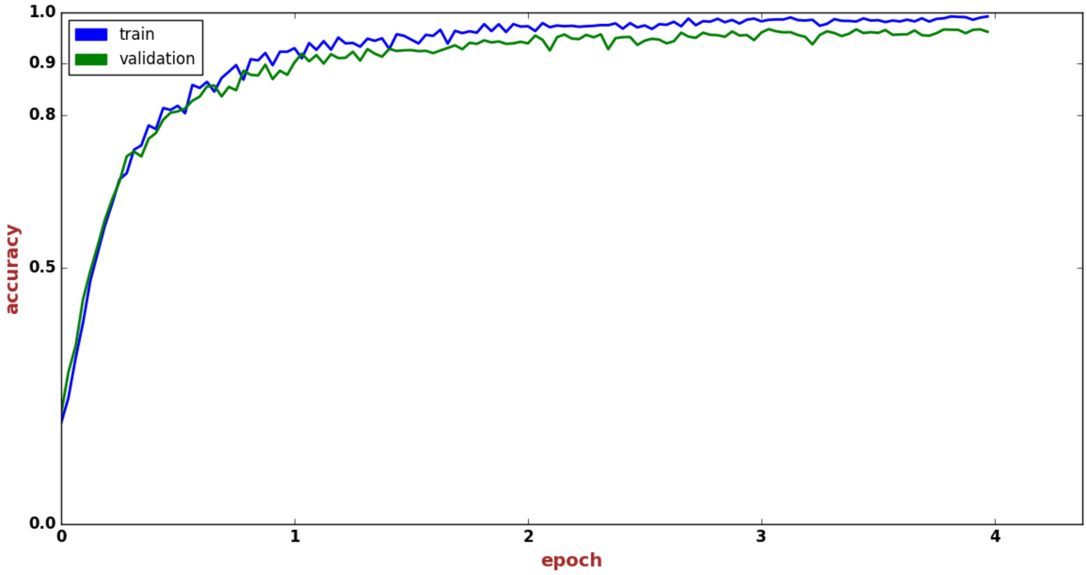
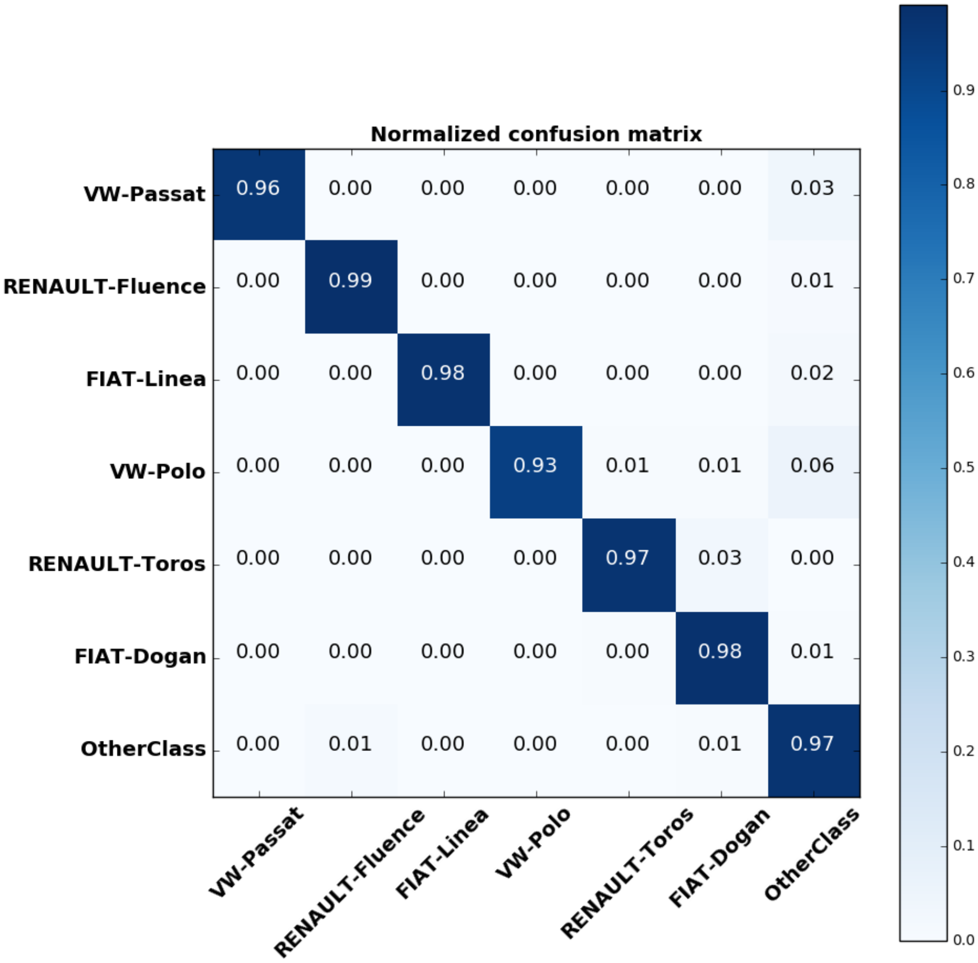


Figure 4.9. Accuracy graph of Experiment II on (224,224,3) shaped test set

Figure 4.10 displays the confusion matrix of this experiment. It reaches 96.85% overall accuracy result on the test set.



	VW-Passat	RENAULT-Fluence	FIAT-Linea	VW-Polo	RENAULT-Toros	FIAT-Dogan	OtherClass
VW-Passat	0.96	0.00	0.00	0.00	0.00	0.00	0.03
RENAULT-Fluence	0.00	0.99	0.00	0.00	0.00	0.00	0.01
FIAT-Linea	0.00	0.00	0.98	0.00	0.00	0.00	0.02
VW-Polo	0.00	0.00	0.00	0.93	0.01	0.01	0.06
RENAULT-Toros	0.00	0.00	0.00	0.00	0.97	0.03	0.00
FIAT-Dogan	0.00	0.00	0.00	0.00	0.00	0.98	0.01
OtherClass	0.00	0.01	0.00	0.00	0.00	0.01	0.97

Figure 4.10. Confusion matrix of Experiment II on (224,224,3) shaped test set

Figure 4.11 and 4.12 show the loss graph and accuracy graph of this experiment **with 300×300 sized images**. With 300×300 sized images, it is not needed to train longer as 224×224 shaped images since the lines reach to the plateau earlier.

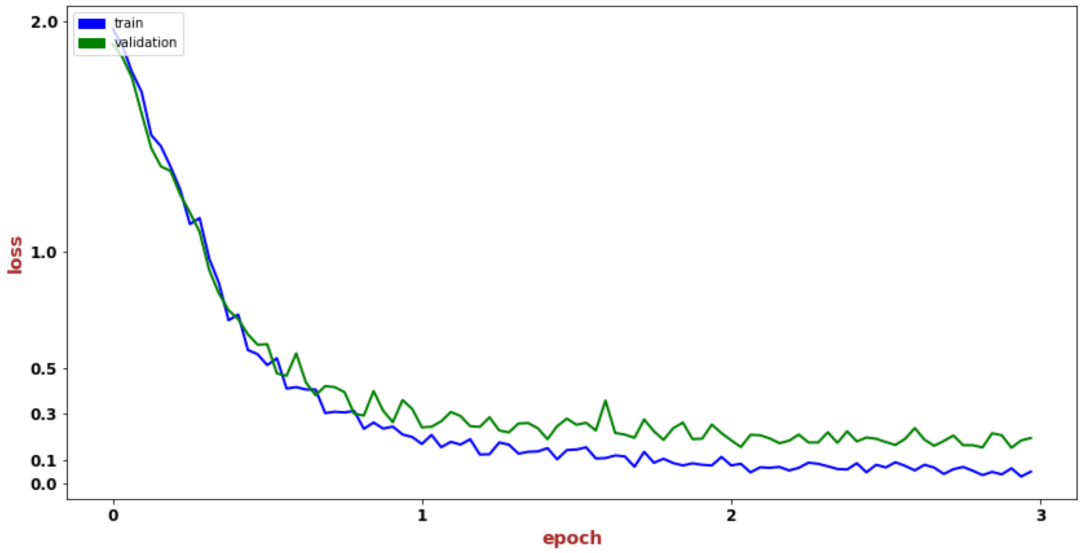


Figure 4.11. Loss graph of Experiment I on (300,300,3) shaped test set

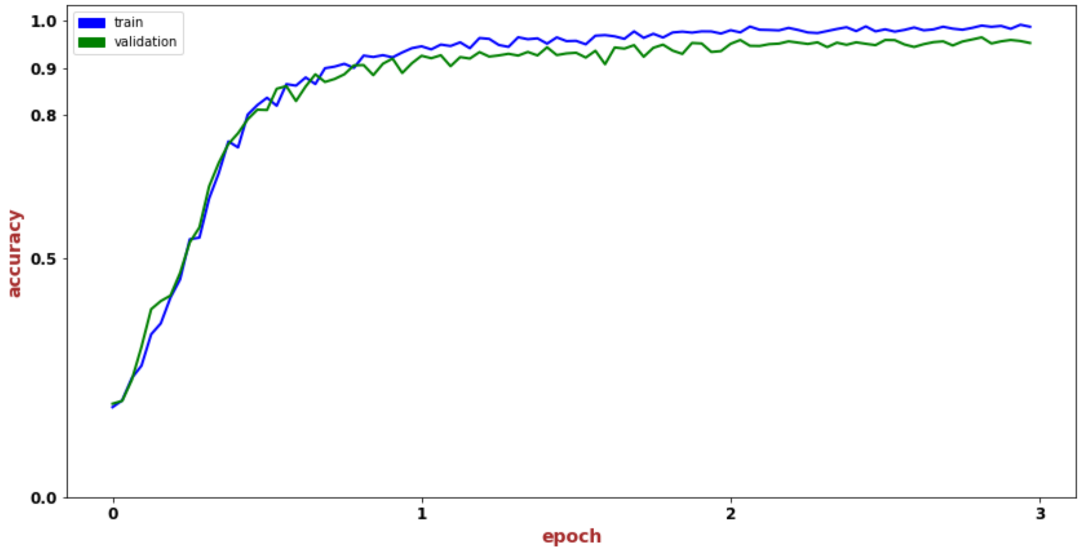
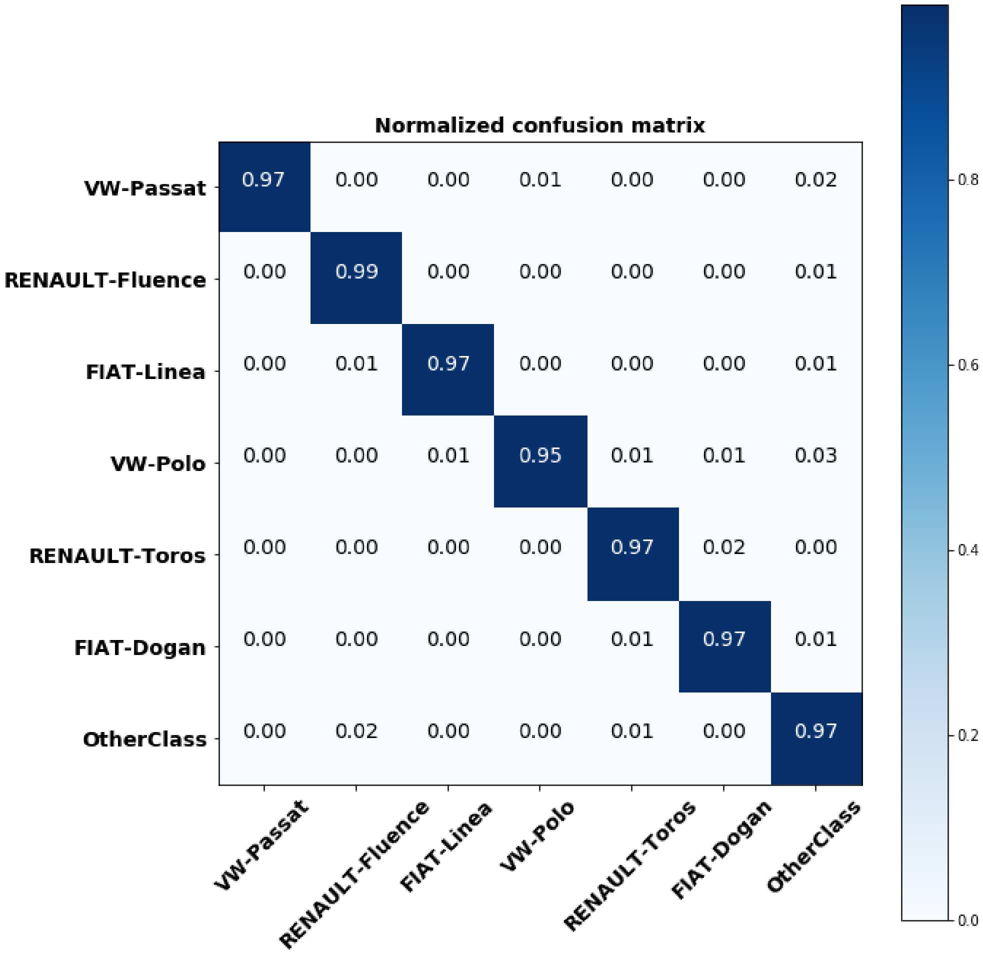


Figure 4.12. Accuracy graph of Experiment I on (300,300,3) shaped test set

Figure 4.13 displays the confusion matrix of this experiment. It reaches 95.10% overall accuracy result on the test set. It is also pleasing to see that Experiment II generalizes the Other Class 10% better. Fiat Polo Class has less amount of images comparing to the other classes. However, Experiment II also generalizes it 11% better than Experiment I.



	VW-Passat	RENAULT-Fluence	FIAT-Linea	VW-Polo	RENAULT-Toros	FIAT-Dogan	OtherClass
VW-Passat	0.97	0.00	0.00	0.01	0.00	0.00	0.02
RENAULT-Fluence	0.00	0.99	0.00	0.00	0.00	0.00	0.01
FIAT-Linea	0.00	0.01	0.97	0.00	0.00	0.00	0.01
VW-Polo	0.00	0.00	0.01	0.95	0.01	0.01	0.03
RENAULT-Toros	0.00	0.00	0.00	0.00	0.97	0.02	0.00
FIAT-Dogan	0.00	0.00	0.00	0.00	0.01	0.97	0.01
OtherClass	0.00	0.02	0.00	0.00	0.01	0.00	0.97

Figure 4.13. Confusion matrix of Experiment II on (300,300,3) shaped test set

4.3 Experiment III: Detection by an SSD Model

This pipeline is implemented with different weights. In the first experiment, it is only used with original base pre-trained weights. In the second experiment, it is performed with the weights which fine-tuned on MS COCO and PASCAL VOC datasets. In Experiment III, less number of epochs are chosen than the others. It is because of that the SSD model is originally pre-trained on ImageNet dataset at first by using a Visual Geometry Group (VGG) based network. Thus, it is no need to be trained more.

Figure 4.14 and 4.15 show the loss graph and accuracy graph of this experiment by **without doing Fine-tuning**.

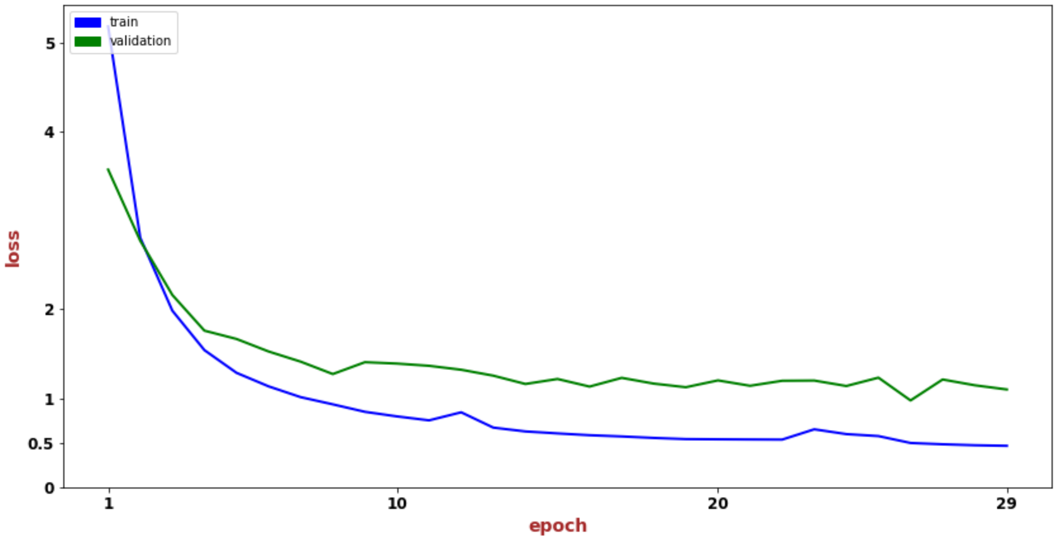


Figure 4.14. Loss graph of Experiment III with VGG based weights on the test set

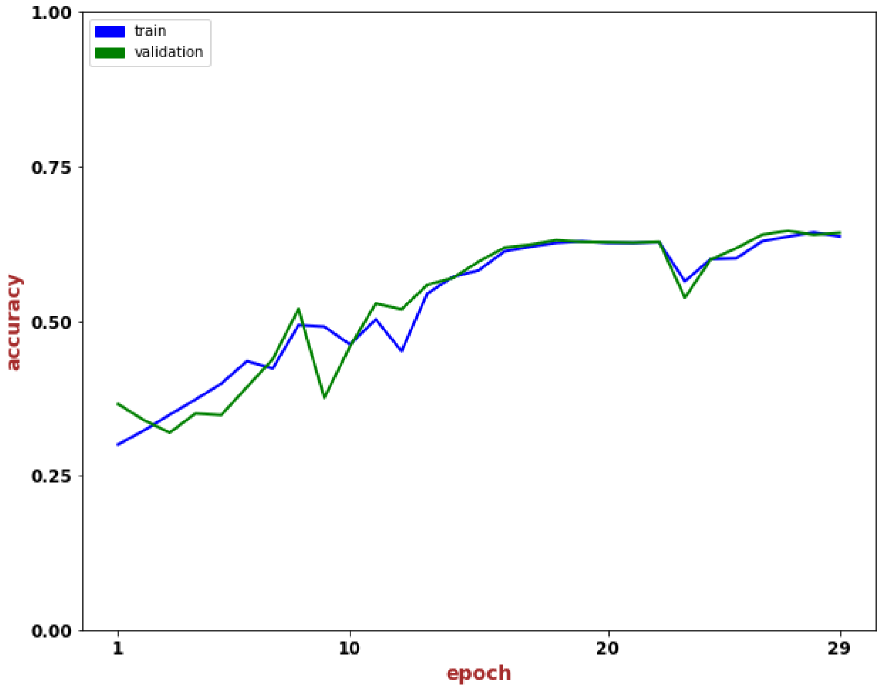


Figure 4.15. Accuracy graph of Experiment III with VGG based weights on the test set

It reaches 70.34% detection accuracy result on the test set. However, localization loss is also included in calculating this accuracy. Thus, the detection result achieves a lower score than the classification result. Using not perfectly shaped GTBB could occur this result regarding detection. Even though it detects reasonably well, since the reference value which is GTBB is not perfectly shaped, such results could happen. Figure 4.16 displays the confusion matrix of this experiment. It is reached 90.57% Mean Average Precision (mAP) score on the test set.

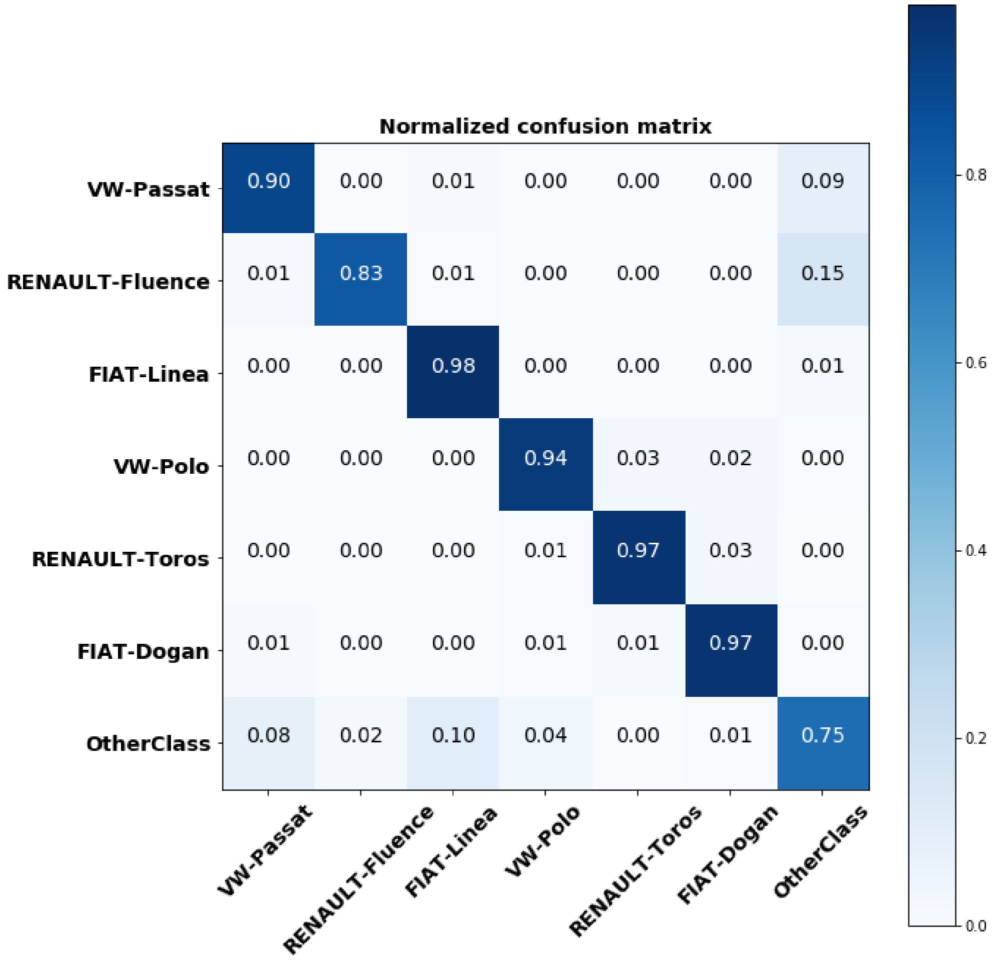


Figure 4.16. Confusion matrix of Experiment III with VGG based weights on the test set

This confusion matrix shows that Experiment III has almost the same and even better classification score on certain classes when comparing to Experiment II. It is also be noted that GTBB of the images is defined by an algorithm which uses a pre-trained SSD model. Besides, the Other Class reaches the lowest accuracy result for its class. It could be mean that generalizing vehicles with fewer images in this model is not reasonable when comparing to Experiment II.

Figure 4.17 and 4.18 show the loss graph and accuracy graph of this experiment by **with doing Fine-tuning**. It reaches 73.27% detection accuracy result on the test set. It can be said that the SSD model achieves approximately 3% higher detection accuracy result by fine-tuning. The same points can be referred like happened in the previous section by without fine-tuning. The main difference between this experiment with the previous one is that the SSD model uses a fine-tuned weights. However, they both also fine-tuned on our dataset.

Moreover, another important metric is the precision-recall curve to calculate the performance of the classification. The results of the precision-recall curve are shown in Appendix A. It shows the results depending on various IoU's. Using the values of True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) the precision-recall values are calculated. The more IoU threshold gets higher, the less result of accuracy in the model being reached. The precision implies how accurate the predictions of the model. However, the recall demonstrates how well to find the positive among all objects.

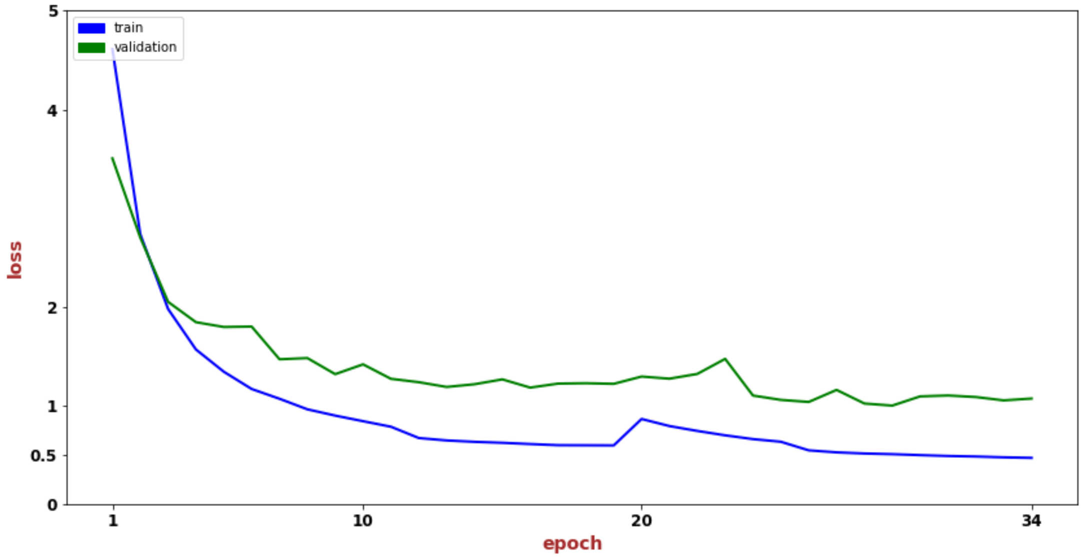


Figure 4.17. Loss graph of Experiment III with fine-tuned weights on the test set

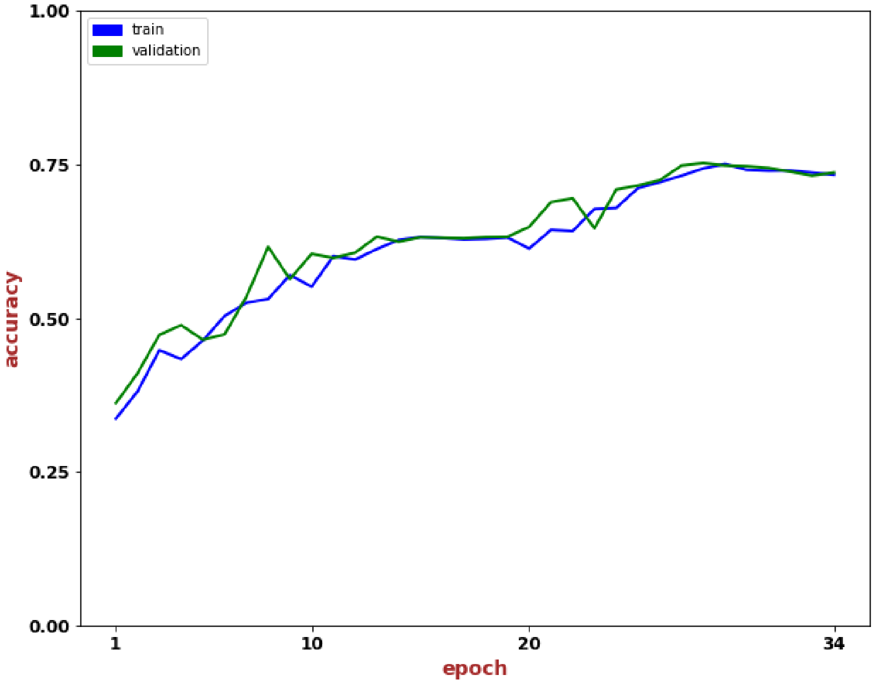
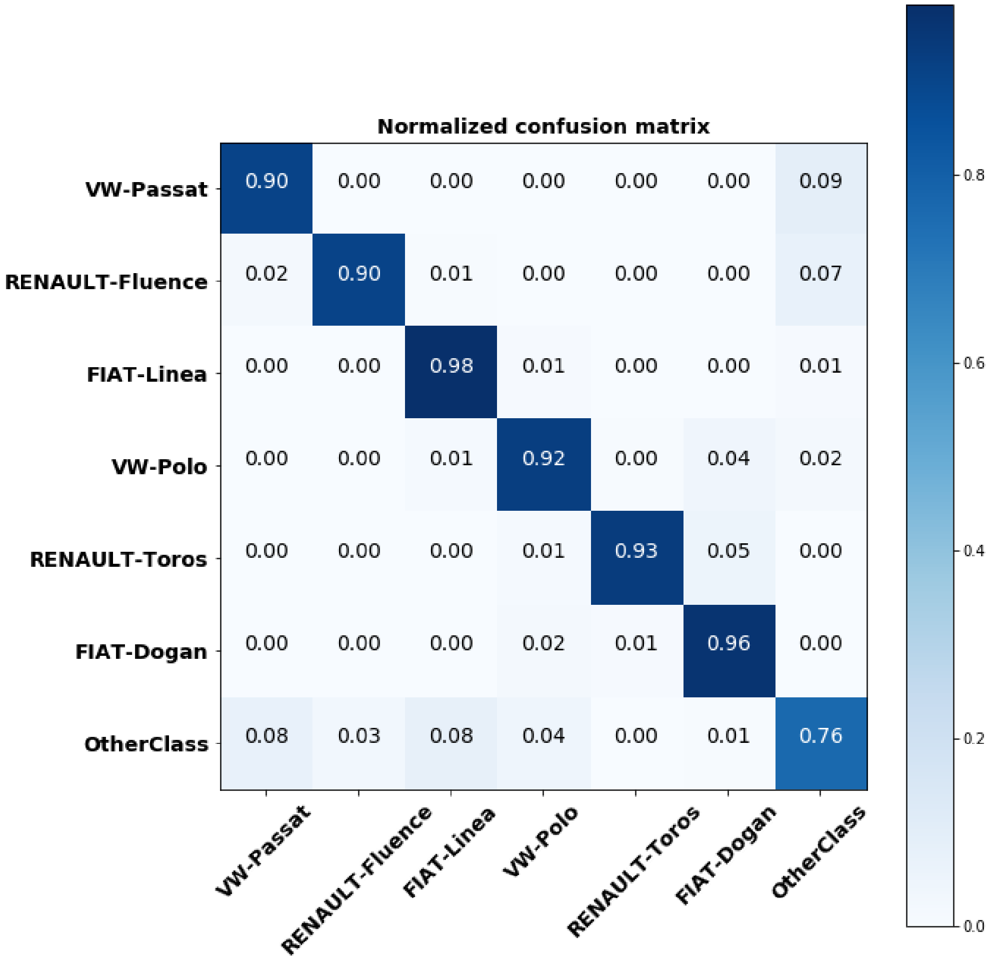


Figure 4.18. Accuracy graph of Experiment III with fine-tuned weights on the test set

Figure 4.19 displays the confusion matrix of this experiment. It is reached 88.52% mAP score on the test set using IoU with a threshold of 0,5.

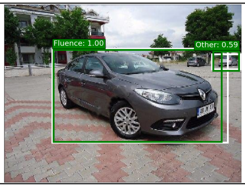
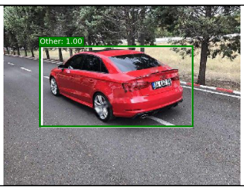
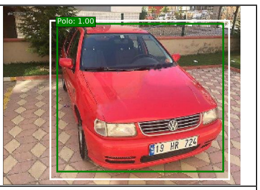
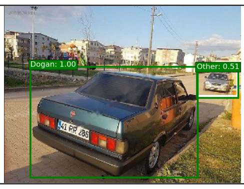
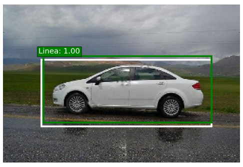
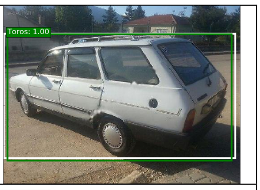


	VW-Passat	RENAULT-Fluence	FIAT-Linea	VW-Polo	RENAULT-Toros	FIAT-Dogan	OtherClass
VW-Passat	0.90	0.00	0.00	0.00	0.00	0.00	0.09
RENAULT-Fluence	0.02	0.90	0.01	0.00	0.00	0.00	0.07
FIAT-Linea	0.00	0.00	0.98	0.01	0.00	0.00	0.01
VW-Polo	0.00	0.00	0.01	0.92	0.00	0.04	0.02
RENAULT-Toros	0.00	0.00	0.00	0.01	0.93	0.05	0.00
FIAT-Dogan	0.00	0.00	0.00	0.02	0.01	0.96	0.00
OtherClass	0.08	0.03	0.08	0.04	0.00	0.01	0.76

Figure 4.19. Confusion matrix of Experiment III with fine-tuned weights on the test set

Table 4.2 shows some positive results which come from implementation of Experiment III pipeline with fine-tuned weights our fine-grained dataset. Green lines stand for the correct predictions. For example, it can be referred from the first sample that the model could predict the vehicle reasonably well although it doesn't have perfectly shaped GTBB. The second, the third, and the fourth image show that it could be crucial for the cars to have a well-centered position when it is compared with the results of false decisions.

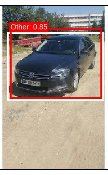
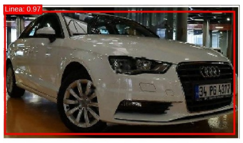
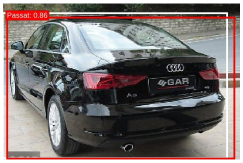
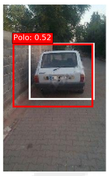
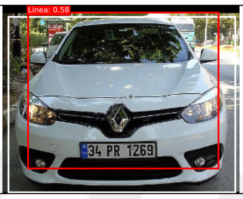
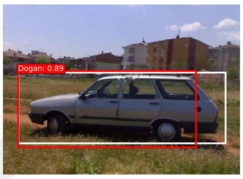
Table 4.2. True decisions on our fine-grained dataset

Sample			
Predicted Class	Renault Fluence	Other Class	VW. Polo
Probability	1.00	1.00	1.00
Sample			
Predicted Class	Fiat Dogan	Fiat Linea	Renault Toros
Probability	1.00	1.00	1.00

Ground Truth ▭ Predicted True 🟩

Table 4.3 shows some negative results which come from implementation of Experiment III pipeline with fine-tuned weights our fine-grained dataset. Red lines stand for the false predictions. For example, it can be referred from the first and second sample that when the vehicles are in the comparatively minor size in the image, it could occur to be detected as a false prediction. Reversely; The second, third, fourth image shows that when the vehicles are relatively large-sized in the photo, it could also cause false detection.

Table 4.3. False decisions on our fine-grained dataset

Sample			
Correct Class	VW. Passat	Other Class	Other Class
Predicted Class	Other Class	Fiat Linea	VW. Passat
Probability	0.85	0.97	0.86
Sample			
Correct Class	Renault Fluence	Renault Toros	Renault Toros
Predicted Class	VW. Polo	Fiat Linea	Fiat Dogan
Probability	0.52	0.58	0.89

Ground Truth ⬜ Predicted False 🟥

The SSD model pipeline is also tested on some commercial videos. It is reached 12 Frame Per Second (FPS) by using the Tesla K80 as a GPU. When it is compared with the one which used in the original paper, it is pretty plausible to reach this performance. This result could be explained because the GPU used in the original document has almost five times better performance quality than the one is used in this thesis. Thus, this is why it is approximately five times worse FPS score than the original study on SSD. The outcomes of the implementation of the pipeline on test videos can be found in the author's repository. https://goo.gl/EB6vyF

Table 4.4 and 4.5 also show the certain frames from the test video results. It can be said that the same reasons for previous false decision table also cause the false decisions in that case.

Table 4.4. True decisions on the test videos

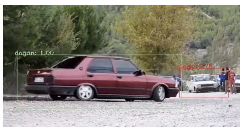
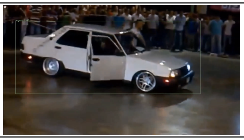
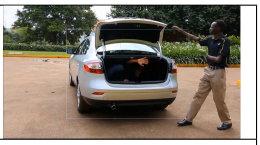
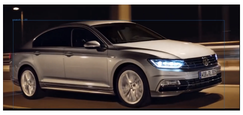
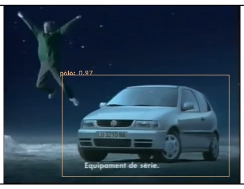
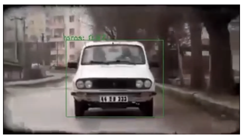
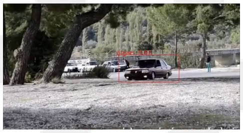
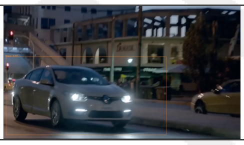
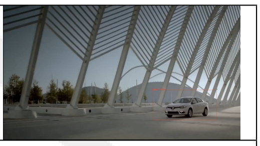
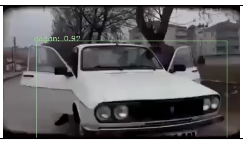
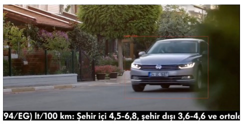
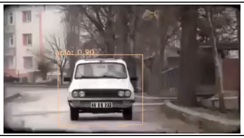
Sample			
Predicted Class	Fiat Dogan	Fiat Dogan	Renault Fluence
Probability	1.00	1.00	1.00
Sample			
Predicted Class	VW. Passat	VW. Polo	Renault Toros
Probability	1.00	0.97	0.92

Table 4.5. False decisions on the test videos

Sample			
Correct Class	Fiat Dogan	Renault Fluence	Renault Fluence
Predicted Class	Other Class	Fiat Linea	Other Class
Probability	0.86	0.97	0.54
Sample			
Correct Class	Renault Toros	Other Class	Renault Toros
Predicted Class	Fiat Dogan	Other Class	VW. Polo
Probability	0.92	0.64	0.90

5 DISCUSSION AND CONCLUSION

This thesis deals with the problems corresponding to vehicle make & model classification. To address those problems following actions are taken. First, a fine-grained dataset is created. Secondly, a model which combines ResNet and SSD is proposed to increase classification accuracy results. Thirdly, a pre-trained SSD model is used to reach high classification and detection results by fine-tuning on our dataset. It is used as an algorithm to automate the annotation process of the vehicles. Lastly, an application is proposed to detect fraud on unauthorized vehicles in a use case by using proposed classification models.

The fine-grained dataset is used in all the experiments. It contains a large number of samples per class which specific to Turkey. However, it took a month to collect them all and filter them from unwanted samples. Since the collecting data is tremendously expensive, the scope of this thesis contains only seven classes. For future work, it is planned to use Amazon Mechanical Turk with the help of a fund. It is also scheduled to have an agreement with online vehicle marketplaces and even government to extend the dataset. Moreover, different experiments are made to reach better classification results. The model which proposed in Experiment II indicates that combining a ResNet based model with an SSD model could increase the classification score significantly. It is projected to use one of the recent state-of-art architectures to reach even higher classification accuracy scores in future.

This thesis also uses a pre-trained SSD based model to detect vehicles. An algorithm is used for annotation which feeds the model with the images that contain GTBB. It reduces the annotation time dramatically by trading-off with perfect shaped bounding boxes. Especially it saves such an amount of time when it is compared with the traditionally drawing bounding box by a manual. Even though it is not used perfectly shaped GTBB, it is reached a reasonable classification and detection scores. For future work, a change in filter sizes of the SSD model architecture could be made to fix false detection results.

Finally, the thesis proposes that an implementation of this model on fraud detection of license plates is considerably possible in certain use cases. The performance of the application depends on how accurate and fast the detection and the prediction are made. A demonstration of this proposed application can be found on the author's repository. The application could be extended to further use cases for future works. For example, it is quite hard to detect an unauthorized vehicle when the license plates cannot be recognized to read. However, security providers easily improve their chance to catch the unapproved cars by filtering them related to their make & model. After reaching the data from surveillance cameras which especially recording from highways, the application could be more robust and reliable to use in the real world.

REFERENCES

Anonymous, 2017. Turkish Istatistical Institute, Road motor vehicles data of Turkey, 2017. https://goo.gl/svnzXN-(Date of access: 26 July 2018).

Anonymous, 2018. Sigmoid function. https://goo.gl/5qJofX-(Date of access: 26 July 2018).

Baran, R., Glowacz, A., and Matiolanski, A. 2015. The efficient real and non-real-time make and model recognition of cars. *Multimedia Tools and Applications*, 74(12):4269–4288.

Buch, N. E. 2010. Classification of Vehicles for Urban Traffic Scenes. *'PhD Thesis'*, Kingston University, London, UK.

Chang, S.-L., Chen, L.-S., Chung, Y.-C., and Chen, S.-W. 2004. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):42–53.

Dahms, C. 2016. Opencv 3 license plate recognition python. https://goo.gl/Wk6GFT-(Date of access: 26 July 2018).

Dehghan, A., Zain Masood, S., Shu, G., and Ortiz, E. G. 2017. View independent vehicle make, model and color recognition using convolutional neural network. arXiv:1702.01721.

Du, S., Ibrahim, M., Shehata, M., and Badawy, W. 2013. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. 2015. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

Fei-Fei Li, Justin Johnson, S. Y. 2017. Lecture 6: Training neural networks, part I lecture notes. https://goo.gl/AK1MNU-(Date of access: 26 July 2018).

Ferrari, P. 2017. A keras port of single shot multibox detector. https://goo.gl/R9dbuz-(Date of access: 26 July 2018).

Goodfellow, I., Bengio, Y., and Courville, A. 2016. Deep learning. MIT Press. http://www.deeplearningbook.org-(Date of access: 26 July 2018).

Gupte, S., Masoud, O., Martin, R. F. K., and Papanikolopoulos, N. P. 2002. Detection and classification of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):37–47.

He, K., Zhang, X., Ren, S., and Sun, J. 2015. Deep residual learning for image recognition. arXiv:1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. 2016. Identity mappings in deep residual networks. arXiv:1603.05027v3.

Hosang, J., Benenson, R., and Schiele, B. 2017. Learning non-maximum suppression. IEEE Conference on Computer Vision and Pattern Recognition, 21-26 July 2017, Honolulu, Hawaii.

Ioffe, S. and Szegedy, C. 2015. Batch normalization accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, 6-11 July 2015, Lille, France.

Kingma, D. P. and Ba, J. 2014. Adam: A method for stochastic optimization. International Conference on Learning Representations ICLR, 7-9 May 2015, San Diego, USA.

Krause, J., Jin, H., Yang, J., and Fei-Fei, L. 2015. Fine-grained recognition without part annotations. In 2015 IEEE Conference on Computer Vision and Pattern Recognition CVPR, 7-12 June 2015, Boston, USA.

Kriesel, D. 2007. A brief introduction to neural networks. http://www.dkriesel.com/en/science/neural_networks-(Date of access: 26 July 2018).

LeCun, Y. and Bengio, Y. 1998. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pp: 255-258. MIT Press, Cambridge, MA, USA.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. 1998. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, pp: 2278-2324.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. 2014. Microsoft COCO: Common Objects in Context. Computer Vision - 12th European Conference, ECCV 2014, 6-12 September 2014, Zurich, Switzerland.

Liu, D. 2016. Monza : Image classification of vehicle make and model using convolutional neural networks and transfer learning.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. 2015. SSD: Single shot multiBox detector. Computer Vision - 14th European Conference, ECCV 2016, 8-16 October 2016, Amsterdam, The Netherlands.

Mitchell, T. M. 1997. Machine Learning. McGraw-Hill, Inc., New York, USA, 414 pp.

Ng, A. 2017. Cs229: Machine learning lecture notes. https://goo.gl/WdmCRk-(Date of access: 26 July 2018).

Nielsen, M. 2015. Neural Networks and Deep Learning. Determination Press. http://neuralnetworksanddeeplearning.com-(Date of access: 26 July 2018).

Simonyan, K. and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv:1409.1556.

Singh, R. 2012. Vehicle model identification. *'Master's Thesis'*, National Institute of Technology, Rourkela, Odisha, India.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Szegedy, C., Reed, S., Erhan, D., Anguelov, D., and Ioffe, S. 2014. Scalable, High-Quality Object Detection. arXiv:1412.1441.

Tafazzoli, F., Frigui, H., and Nishiyama, K. 2017. A large and diverse dataset for improved vehicle make and model recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 874–881.

Yang, L., Luo, P., Change Loy, C., and Tang, X. 2015. A large-scale car dataset for fine-grained categorization and verification. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3973-3981.

Zhou, Y. and Cheung, N.-M. 2016. Vehicle classification using transferable deep neural network features. CoRR, abs/1601.01145.

APPENDICES

APPENDIX 1 Precision-Recall and Average Precision
APPENDIX 2 ResNet Code Block

APPENDIX 1

PRECISION-RECALL AND AVERAGE PRECISION

Table 1.1 displays the average precision of all classes for different IoU's.

Table 1.1. Average precision table

		AP							**mAP****
		Passat	Fluence	Linea	Polo	Toros	Dogan	Other	
IoU*	0.5	0.892	0.899	0.926	0.927	0.898	0.904	0.747	0.885
	0.6	0.892	0.899	0.926	0.888	0.898	0.902	0.740	0.878
	0.7	0.887	0.897	0.891	0.888	0.898	0.898	0.692	0.865
	0.8	0.750	0.767	0.864	0.874	0.885	0.726	0.518	0.769

IoU* refers to a parameter of detection. mAP** refers to a parameter of classification.

While Figure 1.1 shows the precision-recall curve of other class, Figure 1.2 shows the precision-recall curve of the first six classes. In the curve, the black line refers to the value of 0.8 for IoU threshold. The orange line refers to the value of 0.7; the red line refers to the value of 0.6, and the blue line refers to the value of 0.5 for IoU threshold.

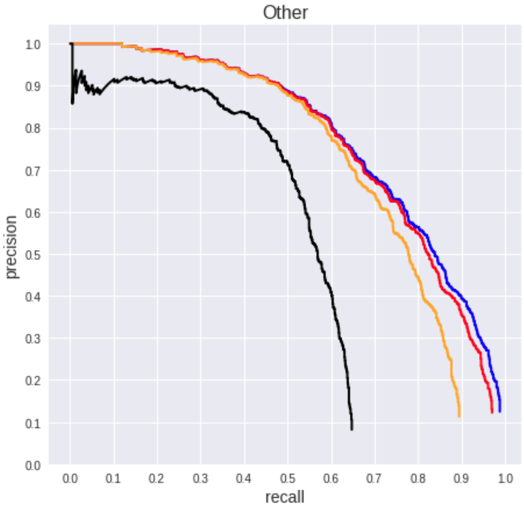


Figure 1.1. The precision-recall curve of the other class

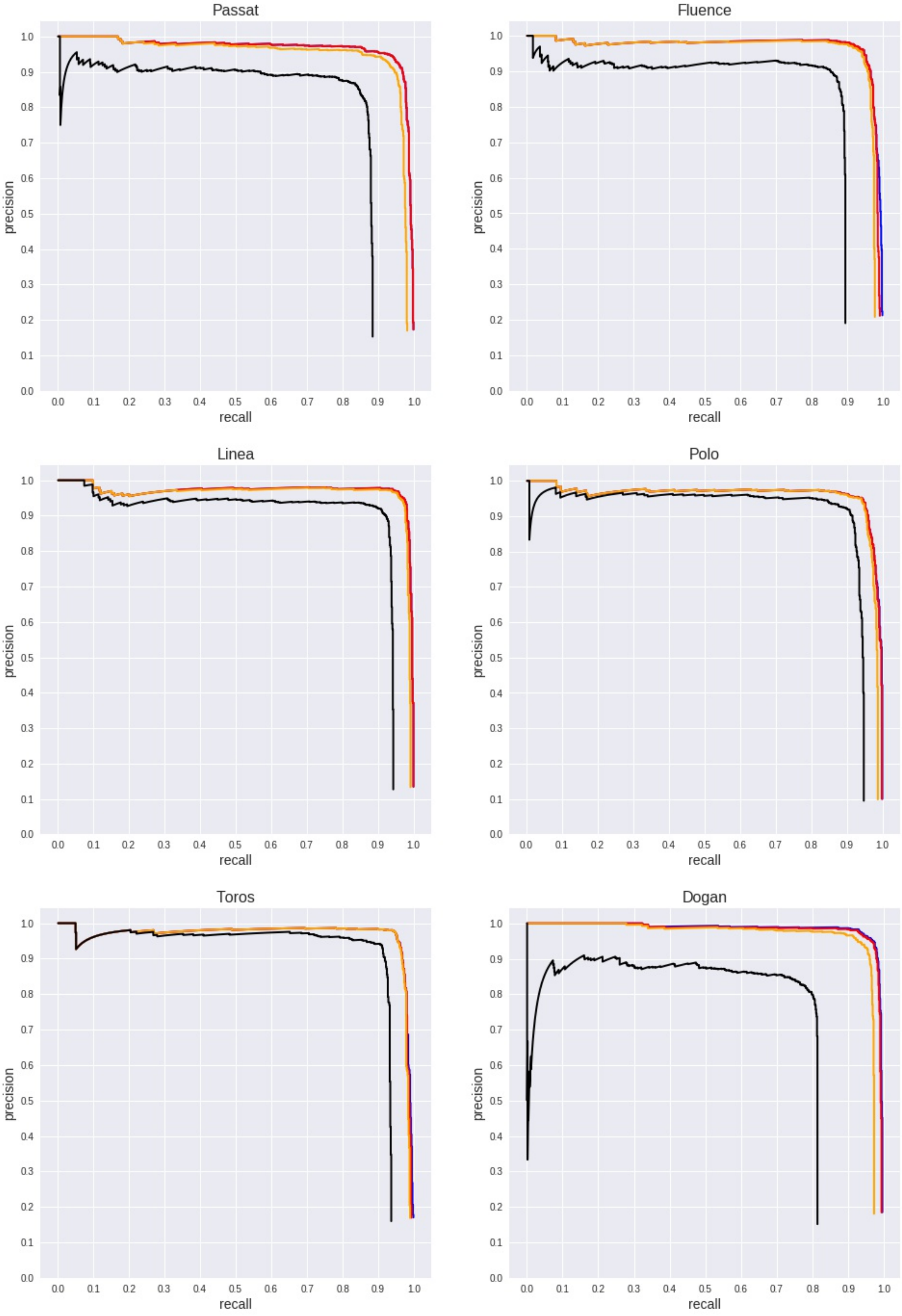


Figure 1.2. The precision-recall curve of the first six classes

APPENDIX 2

RESNET CODE BLOCK

In [1]:

```
### The Model
def ResNet50(input_shape = (300, 300, 3), classes = 7):
    # Getting the Input
    X_input = Input(input_shape)
    # Zero-Padding
    X = ZeroPadding2D((1, 1))(X_input)

    # Stage 1
    X = Conv2D(16, (3, 3), strides = (2, 2), name = ’conv1’,
    kernel_initializer = glorot_uniform(seed=0))(X)
    print(’Conv1a: ’ + str(X.shape))
    X = BatchNormalization(axis = 3, name = ’bn_conv1’)(X)
    X = Activation(’relu’)(X)

    # Stage 2
    X = convolutional_block(X, f = 3, filters = [16, 16, 32],
    stage = 2, block=’a’, s = 2)
    print(’Convblock2a: ’ + str(X.shape))
    X = identity_block(X, 3, [32, 32, 32], stage=2, block=’b’)
    print(’Identityblock3b: ’ + str(X.shape))
    X = MaxPooling2D( strides=(2, 2))(X)
    print(’MaxPool2a: ’ + str(X.shape))
```

In [1]:

```
# Stage 3
X = convolutional_block(X, f = 3, filters = [32, 32, 64],
stage = 3, block=’a’, s = 2)
print(’Convblock3a: ’ + str(X.shape))
X = identity_block(X, 3, [64, 64, 64], stage=3, block=’b’)
print(’Identityblock3b: ’ + str(X.shape))
X = identity_block(X, 3, [64, 64, 64], stage=3, block=’c’)
print(’Identityblock3c: ’ + str(X.shape))
X = MaxPooling2D( strides=(2, 2))(X)
print(’MaxPool3a: ’ + str(X.shape))

# Stage 4
X = convolutional_block(X, f = 3, filters = [64, 64, 128],
stage = 4, block=’a’, s = 2)
print(’Convblock4a: ’ + str(X.shape))
X = identity_block(X, 3, [128, 128, 128], stage=4,
block=’b’)
print(’Identityblock4b: ’ + str(X.shape))

# Stage 5
X = convolutional_block(X, f = 3, filters = [128, 128,
256], stage = 5, block=’a’, s = 2)
print(’Convblock5a: ’ + str(X.shape))
X = MaxPooling2D( strides=(2, 2))(X)
print(’MaxPool: ’ + str(X.shape))
```

In [1]:

```
    # Output layer
    X = Flatten()(X)
    X = Dense(64)(X)
    X = Activation(’relu’)(X)
    X = Dropout(0.5)(X)
    X = Dense(classes, activation=’softmax’, name=’fc’ +
    str(classes), kernel_initializer =
    glorot_uniform(seed=0))(X)

    # Create model
    model = Model(inputs = X_input, outputs = X,
    name=’ResNet50’)

    return model
```

In [2]:

```
\## Run the model
model = ResNet50(input_shape = (300, 300, 3), classes = 7)
model.compile(optimizer=’adam’,
    loss=’categorical_crossentropy’, metrics=[’accuracy’])
```

CURRICULUM VITAE

Full Name : Burak Satar
Birth Place and Date : Bursa, 1991
Languages : Turkish, English, Spanish, Italian

Education Status (Institute and Year)
High school : Bursa Anatolian Girl High School 2009
Bachelor : Uludağ University 2014

Workplaces and Year : TSV Energy 2014-2015
University of Valencia 2016-2017
Turkcell 2017-2018
E-mail : buraksatar@gmail.com
Publications :

Satar, B. 2016. An IoT based bus stop system design. National Conference on Electrical and Electronics Engineering ELECO, 1-3 December 2016 Bursa, Turkey.

Satar, B. and Dirik, A. E. 2018. Deep learning based vehicle make-model classification. 27th International Conference on Artificial Neural Networks ICANN, 4-7 October 2018, Rhodes, Greece.