MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

DRIVER PROFILING USING LONG SHORT TERM MEMORY (LSTM) AND CONVOLUTIONAL NEURAL NETWORK (CNN) METHODS

ASLIHAN CURA

MASTER THESIS

Electrical and Electronic Engineering

Thesis Supervisor

Prof. Dr. Haluk Küçük

ISTANBUL, 2019

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

DRIVER PROFILING USING LONG SHORT TERM MEMORY (LSTM) AND CONVOLUTIONAL NEURAL NETWORK (CNN) METHODS

ASLIHAN CURA

525014012

MASTER THESIS

Electrical and Electronic Engineering

Thesis Supervisor

Prof. Dr. Haluk Küçük

ISTANBUL, 2019

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

Aslıhan CURA, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended her thesis entitled "**Driver Profiling Using Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) Methods**", on June 28, 2019 and has been found to be satisfactory by the jury members.

Jury Members

Prof. Dr. Haluk Küçük (Advisor)
Marmara University .....(SIGN).....

Assoc. Prof. Bora ESMER (Jury Member)
Marmara University .....(SIGN).....

Assoc. Prof. Esin Öztürk IŞIK (Jury Member)
Bogazici University .....(SIGN).....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that Aslıhan CURA be granted the degree of Master of Science in Department of Electrical and Electronic Engineering on 10/07/2019 (Resolution no: 2019/14-05)

Director of the Institute

Prof. Dr. Bülent EKİCİ
Enstitü Müdürü

PREFACE / ACKNOWLEDGMENT

I would like to thank to Prof. Dr. Haluk Küçük for his guidance, support and patience throughout the course of this thesis work.

I would like to thank Otokar drivers for patience in data collection.

I would like to thank my husband Burak for his patience.

My mother and father Hatice and Mehmet were always supportive in every stage of my life. I would like to express my sincere gratitude them.

Thanks to my lovely sister Elif for adding joy and smile to my life.

Table of Contents

ÖZET

UZUN KISA SÜRE BELLEKLİ ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI YÖNTEMLERİ İLE SÜRÜCÜ PROFİLLEME

Sürücü araç kullanım şekli, trafik güvenliği, yakıt tüketimi ve gaz emisyonu konuları üzerinde son derece etkilidir. Bu çalışmada, trafik güvenliğini arttırmak için araçtan toplanan verileri yapay sinir ağları kullanarak sınıflandırmak ve bu sayede sürücünün davranış profilini çıkarmak amaçlanmıştır. Sürücü profillemesi üzerindeki yapılan çalışmalar incelendiğinde, akıllı telefonlardan toplanan sensör verileri, kamera görüntüleri ve aracın kendi verileri birlikte kullanılarak sürücü profili çıkarılma üzerine yoğunlaşıldığı görülmüştür. Bu çalışmadaki ise sadece aracın; hız, motor devri, gaz pedalı, fren pedalı, teker açısı ve ivmelenme gibi verileri kullanılarak sınıflandırma yapılmıştır. Sınıflandırmada iki farklı derin öğrenme metodu kullanılmıştır. Zaman bağlı veriler için sıklıkla kullanılan Uzun-Kısa Süreli Bellek (LSTM) ve görüntü işlemede kullanılan ancak zamana bağlı verilerde de tercih edilen CNN (Convolutional Neural Network) derin öğrenme metodu kullanılarak sınıflandırmadaki başarı oranları incelenmiştir. Çalışma sonucunda CNN'in daha yüksek başarı sonuçları verdiği gözlemlenmiştir.

ABSTRACT

DRIVER PROFILING USING LONG SHORT TERM MEMORY (LSTM) AND CONVOLUTIONAL NEURAL NETWORK (CNN) METHODS

Driver profiling has a major impact on traffic safety, fuel consumption and gas emission. The purpose of this work is to feed and train the neural network with the vehicle data and classify the driver behavior. When the driver profiling studies are examined, the majority of studies have classified the driver using sensor, image and vehicle data together. In this study, only the vehicle data such as engine speed, torque, steering wheel angle etc. were used. To classify driver, two different methods were implemented. One of them is Long Short Term Memory (LSTM) Neural Network which is usually for time series data classification and the other method is Convolutional Neural Network (CNN) which is frequently used for image classification but also can be used for time series classification. In the results section of this study, the success rates of two methods in classification were analyzed and the outcomes indicated that Convolutional Neural Network provided higher success rates.

SYMBOLS

σ : Sigmoid

C_t : Current cell state

C_{t-1} : Previous cell state

X_t : Input vector

h_t : Output of the current cell

h_{t-1} : Output of the previous cell

b :Bias vector

W : Weight

$\hat{C}_T$: Candidate vector

ABBREVATIONS

ANN : Artificial Neural Network

CNN : Convolutional Neural Network

DTW : Dynamic Time Warping

ECG : Electrocardiography

GMM : Gaussian Mixture Model

GPS : Global Positioning System

GRU : Gated Recurrent Neural Network

Tanh : Hyperbolic Tangent

HAR : Human Activity Recognition

IoT : Internet of Things

LSTM : Long Short Time Memory

NLP : Natural Language Processing

OBD : On Board Diagnostic

PHYD : Pay How You Drive

ReLU : Rectified Linear Unit

RNN : Recurrent Neural Network

SVM : Support Vector Machine

UBI : Usage Based Insurance

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1- INTRODUCTION

1.1. Motivation and Overview

Driving style has huge impact on traffic safety. Driver may be under the influence of alcohol, narcotics or can be drowsy. Some drivers may endanger the traffic willingly. In such cases, driving style becomes dangerous. According to Turkish Statistical Institutes 2017 road traffic accident data, in Turkey, 1,202,716 traffic accidents occurred. Among those accidents 1,020,047 resulted in material loss and 182,669 caused a death or injury. Looking at the 182,669 total faults causing accidents involving death or injury during in 2017, it was observed that 89.9% of faults were driver related, 8.5% of faults were pedestrian faults, 0.7% of faults were road faults, 0.5% were vehicle faults and 0.4% were passenger related [1].

As well as the traffic safety, driving style is also important criterion for fuel and energy consumption. By changing driver style, energy and fuel consumption can be reduced [2], [3]. Purpose of analyzing of vehicle data is to observe driver and determine driver behavior. The result of the analysis aims to measure the aggressiveness of the driver to help impose sanctions that may ultimately reduce aggressiveness.

In recent year, driver profiling has gained growing importance for insurance companies. Usage Based Insurance (UBI) also known as Pay How You Drive (PHYD) systems has recently been implemented [4]- [5]. The main idea is instead of a fixed cost, drivers are expected to pay according to their driving behavior. In this insurance system, driver who demonstrates aggressive breaking, line changing or over speeding, have to pay much more than other drivers. This system increases traffic safety due to its deterrence, preventing accidents and increasing the service life of the vehicle. At the same time, the driver's behavior leads to inevitable saving in fuel / energy consumption. In addition to insurance companies, the driver behavior profile is extremely important for companies that perform fleet management and intercity transport. The behavior of the driver can be ameliorated when the behavior of aggressive drivers is shared with them and the necessary warnings are made.

1.2. Purpose of the Thesis

The aim of this thesis is to classify driver behavior with two different methods which are LSTM and CNN deep learning methods. For this purpose, data was collected from the vehicle as shown in Figure 1.1 and fed to the network trained with TensorFlow library.

When driver behavior studies examined, it was seen that, different methods and different data types were used. Data was collected from three different data sources as below.

- OBD-II
- Smart Phones
- Cameras

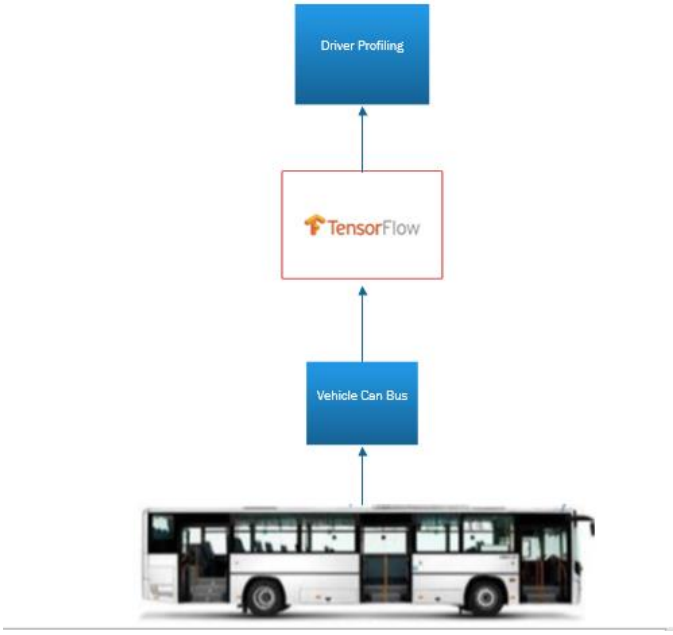


Figure 1. 1 Steps for creating a driver profile

Some studies, used all data sources together, whereas some other used just one data source. On Board Diagnostic (OBD) is a versatile electronic system which communicates with car electronic system and collects vehicle parameters such as brake pedal position, fuel consumption. The data from sensors such as magnetometer, acceleration, gyroscope and GPS in the phone can be used with the smartphone fixed in the vehicle. In the same way,

the driver's behavior can be examined with the camera mounted in the vehicle and the distance to other vehicles can be performed by monitoring the path and the aggression.

Examining the literature, it was observed that machine learning and artificial neural network techniques were applied to driver behavior profiling. Sensor data such as magnetometer, acceleration sensor and gyroscope of smart phones have been used to identify behaviors such as sudden acceleration, deceleration, hard turn and sudden lane change [6]. Dynamic Time Warping (DTW) and Bayesian Classification have been used to reveal the risky behavior of the driver. With smartphone data, GPS and video camera data was also used to classify driver behavior using DTW [7] method. To reveal the eco driving behavior model in electric vehicles, besides the acceleration data from the smartphone, speed data from vehicle Can Bus was collected for neural network training [8]. In their study using acceleration data collected from mobile phone, Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM) and Gated Recurrent Neural Network (GRU) neural network methods were used to classify driver behavior. Compared to three different methods, the highest success rate was obtained from GRU [9]. In their study, acceleration, engine speed information were obtained from OBD and were used to determine the anomaly in the behavior of the driver using the Markov model, K-Means clustering and Adaboost machine learning algorithms [10].

Some studies have been based on OBD and visual data from vehicle camera. In one of the studies examined, lane tracking and vehicle tracking distance information was obtained from the camera and vehicle speed and engine speed information was taken from CAN-bus [11]. This information was modeled by using Gaussian Mixture model (GMM) and converted into feature vectors. These vectors were classified using Support Vector Machine (SVM) to detect aggressive driving behavior. In [12]; torque, engine revolutions per minute, vehicle speed, torque, acceleration pedal position, throttle pedal position, intake manifold pressure, accelerometer and GPS data provided by OBD device has been used to identify the drunk driver. Collected data was interpreted with Logistic Regression algorithm and classification was achieved with an accuracy of 82%. [13] used the same data source and almost the same data but implemented GMM algorithm to perform driver behavior analysis and further identification of aggressive behavior. In [14], acceleration, deceleration and rotation data was used with SVM and K-means clustering machine

learning algorithms to identify driving style. Quintero et al. [15] proposed a method to detect erratic driving behavior using GPS, OBD-II and other localization sensor data supplied to a Feedforward Neural Network. To estimate the acceleration intend, Long Short Term Memory Neural Network and Feedforward neural networks [16] were compared which revealed that LSTM indicated favorable accuracies. In another study by using GPS, OBD-II and car camera data, LSTM has been used to predict the subsequent movement of a driver [17]. The difference of this study from the others discussed in the literature is using LSTM method trained with the onboard data recorded by can-bus of the car.

Convolution Neural Network (CNN or ConvNet) is a deep learning method which has proven its success in image classification. CNN is also used to interpret time-dependent data collected from a variety of domains including natural language processing (NLP), medicine, and the Internet of Things (IoT).

In order to attain a meaningful information regarding the data, it is necessary to make optimization with attribute inferences. The connections between neurons and layers and the parameters learned result in very large computational difficulties with a classic neural network model. Convolutional neural networks have been developed by Yann LeCun as a solution to this challenge.

One-dimensional (1D) CNN is used for time-dependent data while 2-dimensional (2D) CNN is used for image classification. CNN has been observed to be highly successful in detecting arrhythmias from various data sources including electrocardiography (ECG) or human motion detection using sensor data obtained from a cellular phone [18].

With the widespread use of wearable technologies, studies on the data collected from these devices have achieved an accelerated pace with a considerable use in disease diagnostics. Due to the elevated risk of cardiovascular complications, the importance of automatic classification on live data has increased. Kiranyaz et al., with the help of wearable devices, collected EEG data from each patient and trained 1-D CNN network for a real time heart monitoring and anomaly detection [19] application. The other study examined, normal beats, supraventricular ectopic beats, ventricular ectopic beats, fusion beats classified with a 1D CNN [20] yielding a 97% accuracy rate. ECG data from the MIT/BIH arrhythmia database were used for both studies. Unlike previous studies

examined, there were additional efforts using 2-D CNN to classify ECG signals yielding better results than the success rate of Kiranyaz et al [21].

Biometrics is an automated system that measures the physical or behavioral uniqueness of an individual and identifies it by comparing it with existing records. In other words, instead of using personnel identification cards, magnetic cards, keys or passwords, biometrics can be used to determine the individual's fingerprints, face, iris, handprints, signature, DNA and retina with easy and convenient verification methods. One study includes the design of a biometric system that classifies ECG signals using deep learning methods [22]. The ECG can be used as a biometrics for verification purposes because it provides detailed information about the electrical operation of the heart and this information is extremely personalized. Four different ECG dataset (MITDB, FANTASIA, NSRDB and QT) were used and SVM, KNN and 1-D CNN algorithms were compared. CNN based algorithm achieved an accuracy of 81.33%, 96.95%, 94.73% and 92.85% on the MITDB, FANTASIA, NSRDB and QT datasets respectively.

1-D CNN is frequently used in human activity recognition (HAR) studies. The data collected from the acceleration sensor of a mobile phone was used to classify movements as falling, sitting, jumping, running, walking, walking upstairs and walking downstairs [23]- [24]- [25]. CNN provided higher success rates than the machine learning methods such as SVM [26].

Examining the studies using CNN in driver profiling [27]- [28]- [29] to classify the driver behavior based on driver images taken in the vehicle, the unsafe behavior during driving condition could be detected. Gao et al. [30] studied to detect dangerous driving situation using video information captured from the vehicle camera. Wang et al. [31] proposed two methods, one using smartphone accelerometer and gyroscope sensor, they forecasted vehicle speed with an LSTM network, while traffic light, stop lines and crosswalks were detected using CNN method. Combining all this information, they focused on unsafe driving detection. In another study [32], vehicle movement direction was predicted using a hybrid combination of CNN and LSTM networks together. Acceleration data obtained from a mobile phone was classified as braking, turn, acceleration and mixed indicating a 90.07% accuracy. Although there are similarities in our study vehicle data was used to train LSTM and 1D-CNN networks separately by comparing their performance indices

independently. Examining the literature, no study 1D-CNN driver profiling based on vehicle data has been found.

1.3. Scope of Thesis

In this thesis, the experimental work was based on the vehicle data collected from vehicle can bus system. The studies have been performed in collaboration with Otokar Otomotiv ve Savunma Sanayi A.Ş a bus and military vehicle manufacturer [33]. After creation of test scenario, train, test and validation dataset were collected from Otokar test buses. Driving tests were pursued for five days and the data was collected with five different Otokar drivers sent to the test track. A neural networks model was developed using TensorFlow [34] library which is an open source machine learning and neural network framework. Computer with Intel Core i7-6600U CPU @2.60GHz was used. In the literature reviewed there are studies using LSTM for driver profiling but no work on CNN use for profiling has been encountered. In this thesis both LSTM and CNN methods were used for driver profiling. CNN method provides a lot of inspiration due to its prominent success in human activity recognition. In one study [18] three dimensional acceleration data from the smartphone sensor was used providing 91% accuracy in classifying human walking, steady, and running activities. Both human activity and driver profiling deal with a time series data, making CNN method has been an interest of this study besides the LSTM method. The comparison of two methods revealed that CNN handles the data much faster providing a higher success rate than the LSTM network.

CHAPTER 2

2. MATERIALS AND METHODS

Speed, fuel consumption and similar data were collected from vehicle' Can Bus communication system according to predetermined test scenarios. The start and end times of the driving behavior were recorded during the test. Data were filtered according to the start and end times of the driving behavior followed by a preconditioning before feeding into the neural network.

Data obtained from the vehicle is time dependent, therefore Long-Short Term Memory (LSTM) and 1D Convolutional Neural Network was studied. Python 3.5 and TensorFlow framework was used throughout this study.

Detailed information about data collection and neural networks will be explained in the following sections.

2.1 Data Collection

In order to determine aggressive, mild and gentle driving, four different test scenarios were created which were deceleration, engine speed, corner turning and lane change. Deceleration, engine speed and corner turning tests were performed on the test track, which was concrete floor of Otokar's Sakarya Arifiye campus illustrated in Figure 2.1.

Figure 2. 1 Otokar Arifiye test track

Line changing tests has been performed at Sakarya Karasu road which can be seen in Figure 2.2. The pavement of the highway is asphalt. During the test days, weather was clear and sunny and tests were completed with Otokar' five different drivers.

Figure 2. 2 Sakarya Karasu Road

66 deceleration tests, 36 engine speed tests, 3 corner turning tests and 5 lane change tests have been completed successfully. One test was repeated five times in deceleration and speed tests. For example, a deceleration test from 30 km speed to 0 km speed in 0-3 seconds was repeated five consecutive times. One test was repeated ten times in corner turning and lane changing tests.

In the deceleration and engine speed tests, the measure of aggressive, mild and gentle behavior was determined by the time-lapse for aggressive, mild and gentle driving (Table 2.1). In the deceleration test, if the vehicle drops to a speed of 0 km in 0-3 seconds while driving at a speed of 30 km, the behavior is labelled as aggressive. If the same test is performed in 4-7 seconds it is labeled as mild driving, whereas a deceleration in 7-10 seconds is labeled as gentle driving (Table 2.1 and Table 2.2).

Table 2. 1 Criteria for determining aggression and gentle in deceleration and speed tests

Test Scenario	**0 sec -3 sec**	**4 sec -7sec**	**7sec -10 sec**
Deceleration	Aggressive	Mild	Gentle
Engine Speed	Aggressive	Mild	Gentle

The sample test scenario for deceleration is described in Table 2.2.

Table 2. 2 Sample test scenario for deceleration

Test No	First Speed(km/sec)	Last Speed(km/sec)	Time (Sec)	Number of Test
1	50	0	0-3	5
2	50	0	4-7	5
3	50	0	8-10	5
4	70	20	0-3	5
5	70	20	4-7	5
6	70	20	8-10	5

In the acceleration tests, the speed parameters and driving time intervals were interpreted using the performance graphics. The speed of the vehicle was determined by looking at the engine speed graphics. In the corner turning and lane change tests, time intervals were determined with the graphs of the lateral acceleration parameter and the wheel angle parameter. The messages acquired from the vehicle and the period of these messages are shown in Table 2.3.

Table 2. 3 Messages and messages period

Messages	Period
Wheel Based Vehicle Speed	100 ms
Actual Retarder Percent Torque	100 ms
Brake Pedal Position	100 ms
Accelerator Pedal Position	50 ms
Percent Load At Current Speed	50 ms
Actual Engine Percent Torque	engine speed dependent
Engine Speed	engine speed dependent
Lateral Acceleration	10 ms
Longitudinal Acceleration	10 ms
Steering Wheel Angle	10 ms
Inlet Air Mass Flow Rate	50 ms
Fuel Rate	100 ms

The duration of aggressive driving data to be entered into artificial neural networks is 3.5 seconds, mild driving is 7 seconds and gentle driving is 10.5 seconds. Therefore, data which takes less than 3 seconds was augmented to 3.5 seconds, this process was done by repeating the last element up to 3.5 seconds. Likewise the mild driving data, which lasts less than 7 seconds, was augmented to 7 seconds. Driving data was supplied in stacks of 3.5 seconds and hence mild driving data was fed to the network in two, gentle driving data in three pieces. 70% of the collected dataset was allocated for training while 30% was used for testing the trained network. The driving data was labelled as “1” for

aggressive, “2” for mild and “3” for gentle driving. The format of the messages obtained from Otokar can be seen in Figure 2.3.

1.3.1 Cruise Control/Vehicle Speed: CCVS

0x00FEF1								PGN Hex
65,265								PGN
100 ms								Rep. Rate
Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8	Byte No
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1					Bit No
Not used in Bus FMS-Standard	Wheel based speed 1/256 km/h Bit gain 0 km/h offset SPN 84	Wheel based speed 1/256 km/h Bit gain 0 km/h offset SPN 84	Clutch switch 00 = pedal released 01 = pedal pressed SPN 598	Not used in Bus FMS-Standard	Not used in Bus FMS-Standard	Not used in Bus FMS-Standard	Not used in Bus FMS-Standard	Name values values values values values SPN
Parking Brake Switch 00 = Parking brake not set 01 = Parking brake set SPN 70			Brake switch 00 = pedal released 01 = pedal depressed SPN 597					Name values values values values values SPN
			Not used in Bus FMS-Standard					
			Cruise control active 00 = switched off 01 = switched on SPN 595					Name values values values values values SPN

Figure 2. 3 Can-Bus message format

Can Bus messages from Otokar were recorded in the .asc file extension format (Figure 2.4) which contains the ID, content and the length of the incoming message.

```
0.053590 CAN 1 Status:chip status error active
0.053590 CAN 2 Status:chip status error active
0.062354 1  CFE6CEEx        Rx   d 8 0A 00 C0 C1 A0 03 66 04  Length = 0 BitCount = 0 ID = 218000622x
0.063779 1  18FEF100x       Rx   d 8 F7 E4 03 10 CC 00 1F C0  Length = 0 BitCount = 0 ID = 419361024x
0.064803 1  CF00400x        Rx   d 8 21 9D 9D C1 17 00 0F 9A  Length = 0 BitCount = 0 ID = 217056256x
0.068367 1  18F0090Bx       Rx   d 8 DD 80 7F CB 7D 57 7D 83  Length = 0 BitCount = 0 ID = 418384139x
0.069511 1  18F0010Bx       Rx   d 8 00 00 C3 FF FF 01 0B FF  Length = 0 BitCount = 0 ID = 418382091x
0.078109 1  CF00A00x        Rx   d 8 00 00 56 09 FF FF FF FF  Length = 0 BitCount = 0 ID = 217057792x
0.084586 1  CF00400x        Rx   d 8 51 9E 9D 1F 18 00 0F 99  Length = 0 BitCount = 0 ID = 217056256x
0.089326 1  18F0090Bx       Rx   d 8 D9 80 7F CB 7D 57 7D 82  Length = 0 BitCount = 0 ID = 418384139x
0.093732 1  18FEF600x       Rx   d 8 FF 08 61 FF FF FF FF FF  Length = 0 BitCount = 0 ID = 419362304x
0.094300 1  18FE5BE8x       Rx   d 8 57 FF FF FF B6 41 01 D9  Length = 0 BitCount = 0 ID = 419322856x
0.099133 1  18F00029x       Rx   d 8 FF FF FF FF FF FF FF FF  Length = 0 BitCount = 0 ID = 418381865x
0.103497 1  CF00300x        Rx   d 8 D0 4E 42 FF FF 0F 72 80  Length = 0 BitCount = 0 ID = 217056000x
0.104622 1  CF00400x        Rx   d 8 71 9E 9D 8A 18 00 0F 99  Length = 0 BitCount = 0 ID = 217056256x
0.109086 1  18F0090Bx       Rx   d 8 D5 80 7F CB 7D 57 7D 82  Length = 0 BitCount = 0 ID = 418384139x
0.109659 1  18FEF200x       Rx   d 8 7C 00 3F 01 01 04 FA FF  Length = 0 BitCount = 0 ID = 419361280x
0.110819 1  18F0010Bx       Rx   d 8 00 00 C3 FF FF 01 0B FF  Length = 0 BitCount = 0 ID = 418382091x
0.112671 1  CFE6CEEx        Rx   d 8 0A 00 C0 C1 A0 03 66 04  Length = 0 BitCount = 0 ID = 218000622x
0.124617 1  CF00400x        Rx   d 8 11 9E 9E 18 19 00 0F 99  Length = 0 BitCount = 0 ID = 217056256x
0.128016 1  18F0090Bx       Rx   d 8 CF 80 7F CB 7D 57 7D 82  Length = 0 BitCount = 0 ID = 418384139x
0.128601 1  CF00A00x        Rx   d 8 00 00 D0 09 FF FF FF FF  Length = 0 BitCount = 0 ID = 217057792x
0.133284 1  18F00100x       Rx   d 8 FF FF FF FF FF FF FF FF  Length = 0 BitCount = 0 ID = 418382080x
0.141728 1  18F00010x       Rx   d 8 00 7D 7D F3 10 FF 00 56  Length = 0 BitCount = 0 ID = 418381840x
0.144741 1  CF00400x        Rx   d 8 41 9E 9E 8C 19 00 0F 98  Length = 0 BitCount = 0 ID = 217056256x
0.148072 1  18F0090Bx       Rx   d 8 C7 80 7F CB 7D 57 7D 82  Length = 0 BitCount = 0 ID = 418384139x
0.148668 1  18F0010Bx       Rx   d 8 00 00 C3 FF FF 01 0B FF  Length = 0 BitCount = 0 ID = 418382091x
0.153206 1  CF00300x        Rx   d 8 D0 4F 43 FF FF 0F 74 80  Length = 0 BitCount = 0 ID = 217056000x
0.162363 1  CFE6CEEx        Rx   d 8 0A 00 C0 C1 D0 03 99 04  Length = 0 BitCount = 0 ID = 218000622x
0.163748 1  18FEF100x       Rx   d 8 F7 64 04 10 CC 00 1F C0  Length = 0 BitCount = 0 ID = 419361024x
```

Figure 2. 4 Can Bus records of the vehicle

Data parser code was written with python3 to separate incoming messages into meaningful components such as speed, acceleration, fuel consumption. With this code, messages read in .asc file was separated according to their IDs; as shown in Figure 2.5, parsed into parameters such as speed, acceleration, fuel consumption, and brake pedal position.

Time	ID	Message	Wheel Based Vehicle Speed	Actual Retarder Percent Torque	Brake Pedal Position	Accelerator Pedal Position	Percent Load At Current Speed
0.00178	CF00400x	08 E1 B1 BA 51 00 0F B1	0	0	0	0	0
0.010089	CF00300x	D4 F7 64 FF FF 0F 82 84	0	0	0	0.988	1
0.011257	18F0010Bx	00 00 C3 FF FF 01 0B FF	0	0	0	0.988	1
0.020499	18FEF100x	F7 64 1C 50 CC 00 1F C0	0.28390625	0	0	0.988	1
0.021047	18F0090Bx	19 8C 7F FB 84 1F 8A 86	0.28390625	0	0	0.988	1
0.021776	CF00400x	08 E1 B0 F1 51 00 0F B0	0.28390625	0	0	0.988	1
0.035065	CF00A00x	68 06 F4 34 FF FF FF FF	0.28390625	0	0	0.988	1
0.040632	18F0090Bx	C7 8B 7F EF 84 1F 8A 86	0.28390625	0	0	0.988	1
0.041689	CF00400x	28 E1 AF 19 52 00 0F AF	0.28390625	0	0	0.988	1
0.048811	18F00010x	00 7D 7D F3 10 FF 00 1E	0.28390625	0	0	0.988	1
0.051112	18F0010Bx	00 00 C3 FF FF 01 0B FF	0.28390625	0	0	0.988	1
0.060514	CF00300x	D0 D7 63 FF FF 0F 7D 84	0.28390625	0	0	0.86	0.99
0.061054	18F0090Bx	75 8B 7F DF 84 17 8A 86	0.28390625	0	0	0.86	0.99
0.061799	CF00400x	58 E1 AE 38 52 00 0F AF	0.28390625	0	0	0.86	0.99
0.065412	18FEF200x	B1 02 A7 01 38 03 FA FF	0.28390625	0	0	0.86	0.99
0.081304	18F0090Bx	21 8B 7F CF 84 0F 8A 86	0.28390625	0	0	0.86	0.99
0.081853	CF00400x	78 E1 AD 6C 52 00 0F AE	0.28390625	0	0	0.86	0.99
0.085553	CF00A00x	2D 07 96 34 FF FF FF FF	0.28390625	0	0	0.86	0.99
0.091678	18F0010Bx	00 00 C3 FF FF 01 0B FF	0.28390625	0	0	0.86	0.99
0.101084	18F0090Bx	CD 8A 7F BB 84 FF 89 86	0.28390625	0	0	0.86	0.99
0.102216	CF00400x	38 CF AD D1 52 00 0F AD	0.28390625	0	0	0.86	0.99
0.110109	CF00300x	D0 B5 64 FF FF 0F 78 84	0.28390625	0	0	0.724	1
0.121051	18F0090Bx	79 8A 7F A7 84 EF 89 85	0.28390625	0	0	0.724	1
0.121616	18FEF100x	F7 98 1C 10 CC 00 1F C0	0.2859375	0	0	0.724	1
0.12218	CF00400x	48 BF AC E0 52 00 0F AD	0.2859375	0	0	0.724	1
0.132548	18F0010Bx	00 00 C3 FF FF 01 0B FF	0.2859375	0	0	0.724	1
0.135673	CF00A00x	B0 07 7D 33 FF FF FF FF	0.2859375	0	0	0.724	1
0.140849	18F0090Bx	25 8A 7F 93 84 D7 89 85	0.2859375	0	0	0.724	1
0.141789	CF00400x	68 AF AB 04 53 00 0F AC	0.2859375	0	0	0.724	1
0.148703	18F00010x	00 7D 7D F3 10 FF 00 1D	0.2859375	0	0	0.724	1

Figure 2. 5 Messages separated and converted to csv file format

In the same code, as shown in Figure 2.6, the necessary parameters such as speed, engine rpm, etc. were plotted. With the help of these graphs, the time intervals of aggressive and gentle driving were determined and the start and end times were recorded as in Table 2.4.

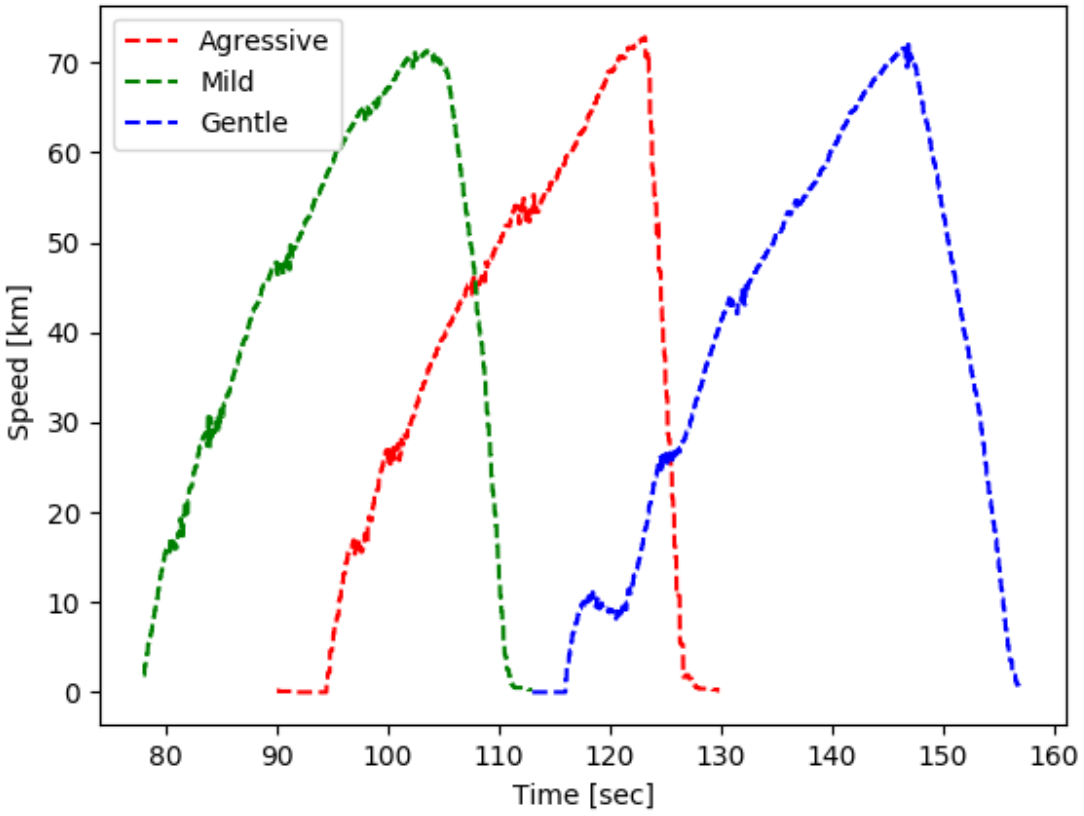


Figure 2. 6 Speed graph of vehicle in deceleration test

Table 2. 4 Deceleration start and end times

Test No	Initial Speed (km/sec)	Final Speed (km/sec)	Time (sec)	Number of test	Start Time	Stop Time	Elapsed Time (sec)
1	20	0	0-3	5	13,71	15,85	2,14
2	20	0	4-7	5	11,97	18,51	6,53
3	20	0	8-10	5	18,83	29,00	10,16
4	20	10	0-3	5	12,80	15,60	2,79
5	20	10	4-7	5	14,38	20,14	5,76
6	20	10	8-10	5	28,51	41,86	13,34
7	30	0	0-3	5	10,33	13,93	3,59
8	30	0	4-7	5	12,00	19,03	7,03
9	30	0	8-10	5	13,38	23,36	9,97
12	40	0	0-3	5	17,40	20,76	3,36
13	40	0	4-7	5	12,20	18,38	6,18
14	40	0	8-10	5	7,53	17,53	10,00

In the acceleration tests, graphics were interpreted by looking at the speed parameter and determining the driving period intervals. In the engine speed test, time intervals were determined by looking at the engine speed of the vehicle; in the corner turning and lane change tests, the graphs of the lateral acceleration parameters together with the wheel angle parameter were illustrated and the time intervals were designated.

2.2 Neural Networks

Using the previously realized examples of an event, the relationship between the input and output of the event to learn, and then predict the output of events that occurred is referred to artificial neural networks.

Artificial neural networks store information obtained during learning by weighted connection between neuron cells. The information is stored in these networks, not in the database. Learning can be defined as calculating weights between connections.

It is possible to classify artificial neural networks according to their structures and learning algorithms. In general there are 3 different learning strategies. These;

- Supervised Learning
- Reinforcement Learning
- Unsupervised Learning

2.2.1 Supervise Learning

In supervised learning, which is the most commonly used learning method in artificial neural networks, an expected output is provided as an example case to the artificial neural network and then the predicted output produced by the network was compared with the real output by considering the difference between the two as an error. Usually, the random picked weights are iteratively modified in a loop until the network error is minimized.

2.2.2 Reinforcement Learning

In this system, the expert helps the learner system. Instead of showing the output set that should be for each input set, the teacher expects the system to generate output for the

input shown to it, and generates a signal indicating that the output generated is true or false. The system continues the learning process by taking this signal into consideration.

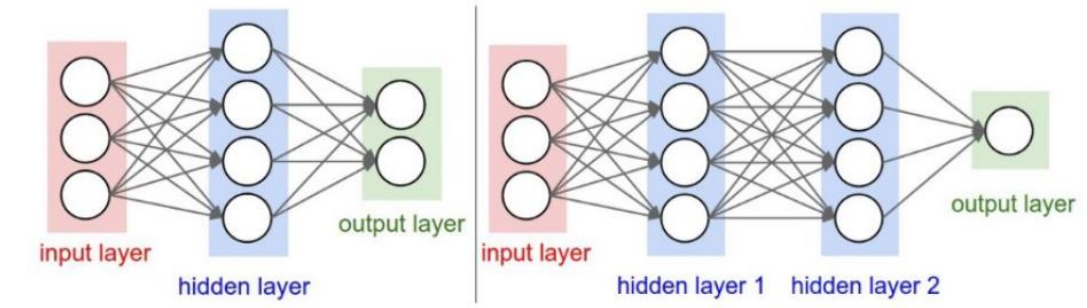
2.2.3 Unsupervised Learning

There is no supervision for this kind of learning that helps the system to learn. Only input values are supplied to the system. The relationship between the parameters of the samples are expected to be learned by the system itself.

Artificial neural networks are split into two according to the directions of the connections between the neurons; these are feed forward and recurrent neural networks.

2.3 Feedforward Neural Network

In feedforward networks, the processor elements are usually divided into layers. Signals are transmitted from the input layer to the output layer in one-way. In the feed-forward ANN, the cells are arranged in layers and the outlets of the cells in a layer which become an input to the next layer via weights. The input layer transmits the information it receives from the external environment to the cells in the middle layer without any modification. The information is processed in the middle and output layers and the network output is determined as shown in Figure 2.7.



Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. **Right:** A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

Figure 2. 7 Different neural network architectures [35]

Input layer: Each neuron in the input layer represents a different property in the data set. It takes the inputs and transmits them to the next layer.

Hidden layer: A series of neurons in which each neuron has a weight assigned to it. It takes the input from the previous layer and scales the inputs with weights, applies the activation function, generates the result, and then transfers the data to the next layer.

Output layer: Generates the prediction for given inputs.

2.4 Recurrent Neural Networks

In a Recurrent Neural Networks, the output of at least one cell is given as input to itself or to other cells, and usually the feedback is produced over a delay element. The feedback can be between cells, between layers or can also be between cells in a layer. Therefore, according to the method the feedback is generated, repetitive architectures can be obtained in various structures and behaviors. A basic RNN given in Figure 2.8 shows that neurons can receive values from both input neurons and the output of the hidden layers (Figure 2.8).

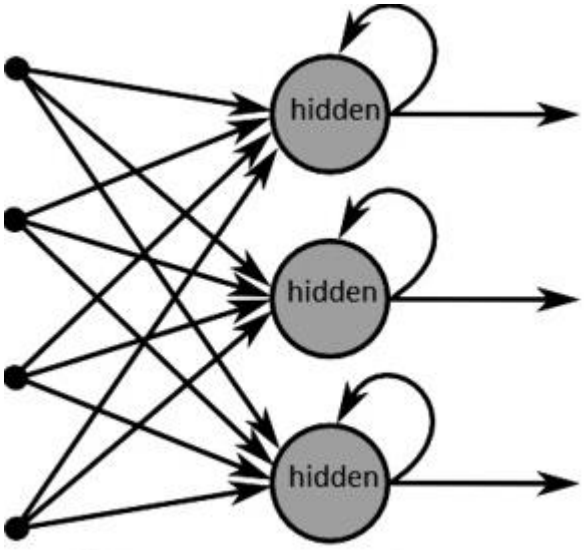


Figure 2. 8 Recurrent Neural Network [36]

2.5 Long Short Term Memory (LSTM)

There is an approach based on previous information usage in Feedback (RNN) architectures. For example, it is easy to predict the word ground in the sentence that states trees grow in ground. But when the gap between contexts increases, it is very difficult for the RNN to use information from the past. For example, “I grew up in England ……., I can speak English fluently.” While guessing the “English” word in this text, it can be estimated that it will be a language name based on the context of the sentence, but it is necessary to keep the sentence at the beginning of the text in memory to predict that the correct word is “English”. The long-term dependencies that are possible in theory have been observed to lead to major problems in practice. In order to solve this problem, as a special type of RNN which can learn long term dependencies, Long Short Term Memory Networks (LSTM) can be used. Introduced by Hochreiter & Schmidhuber (1997) [37], the network works well on a large variety of problems.

One of the methods of feedback artificial neural networks, LSTM neural networks; are used for sentence completion, natural language processing, sensor data, i.e time-dependent, sequential sensor data, understanding, classification and estimation purposes.

In driver behavior identification, the profile is intended to be extracted by using the vehicle operational data recorded during a specific time interval. Since our data is time dependent, Long Short Term Memory (LSTM) artificial neural network is more suitable for our study. In the LSTM topology shown in Figure 2.9, the network was fed with 3.5 seconds data with 12 different components including speed and acceleration measurements.

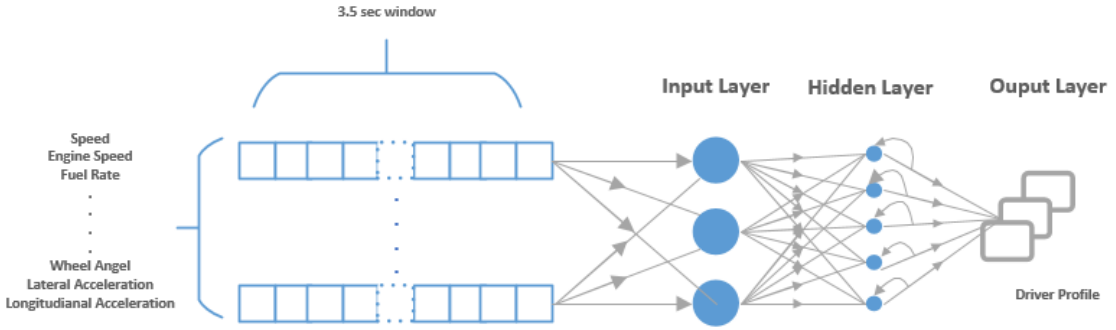


Figure 2. 9 Driver Profiling with LSTM topology

The 3.5-second data set can be considered as a matrix of 12 columns and 684 lines. The learning algorithm was developed using the TensorFlow library. There are parameters that determine the success of an artificial neural network and the success rate of the training can be increased by changing the hyper parameters. These are variables such as batch size, learning rate, training time (epoch), optimization algorithm, activation function, number of layers and number of neurons. In order to increase the success rate, changes were made both to hyper parameters and in data series with detailed information given in the following sections.

2.5.1 Long Short Term Memory Structure

In LSTM, there are two important parameters which are gates and cell states. Gate is a way of transmitting on demand information. LSTM is organized with structures called doors that can add or remove information to the cell state (Figure 2.10).

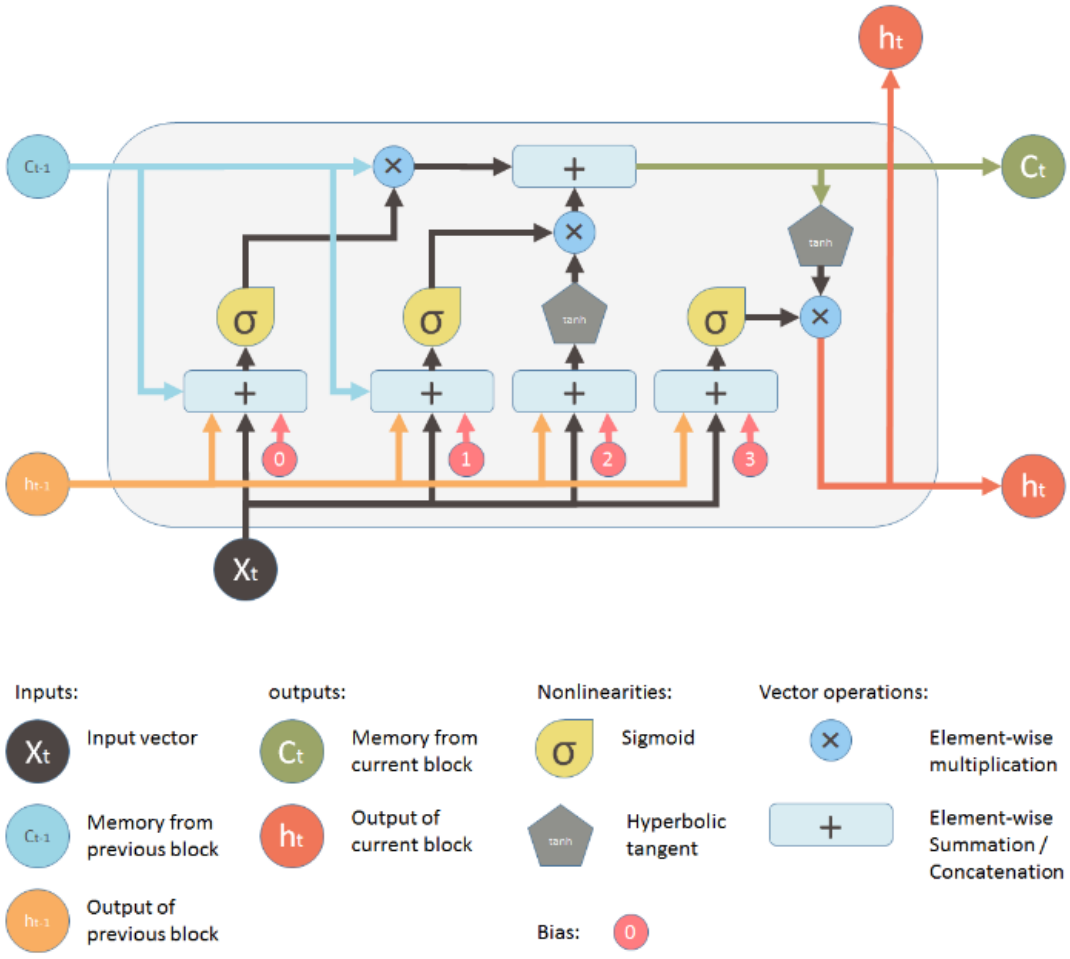


Figure 2. 10 Internal structure of LSTM [38]

The first step in an LSTM algorithm is to decide what information to discard from the cell state. As shown in Figure 2.11, this decision is made by sigmoid layer which is called “forget gate layer”.

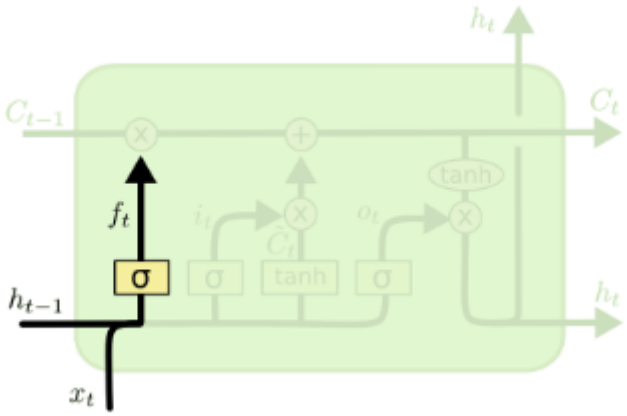


Figure 2. 11 Forget gate [39]

Forget gate looks at h_{t-1} and x_t and give output to the cell state c_{t-1}. Output is a number between 0 and 1, where 1 stands for "keep this information", 0 is for "don't store information", "W" is weight and b is the bias term in Eq. 2.1.

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right) \tag{2.1}$$

The next step is to decide which new information is stored in the cell. There are two parts. First, input gate layer which contains a sigmoid layer, decides the values to be updated. Then, a tanh layer creates an applicant vector $\hat{C}_T$ which will be added to the cell state in Eq. 2.2. As shown in Figure 2.12, the last stage is to combine sigmoid and tanh layers and update the cell status.

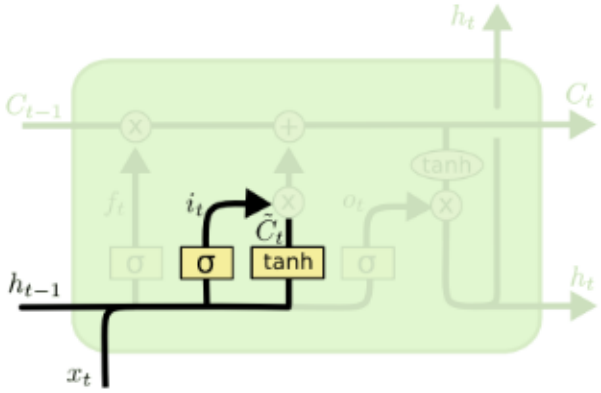


Figure 2. 12 Decision block for selection of new information to store in the cell [39]

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\hat{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.3}$$

Then the old cell state C_{t-1}, is updated into the new cell state C_t. As shown in Figure 2.13, the old state C_{t-1} is multiplied by f_t and then $i_t * \hat{C}_T$ is added in Eq. 2.4. This is the new applicant value scaled by how much it is desirable to update each state value. In the previous language example, this is where the information about the old state is forgotten and the new information is added.

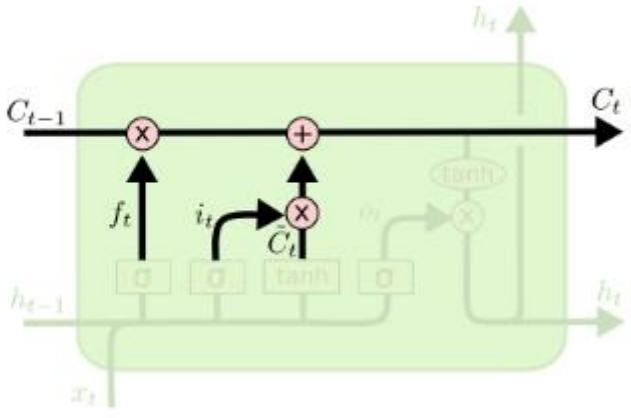


Figure 2. 13 Forgetting the old information and adding the new information [39]

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.4}$$

Finally, comes the decision stage (Figure 2.14) based on filtered cell state. First, a sigmoid layer is executed which decides the components of the cell state to extract. Then, the cell state is fed through tanh and multiplied by the output of the sigmoid gate, producing the output describing the decision in Eq. 2.5 and in Eq. 2.6. For the language model case, this might be the output information about a verb that comes in the following step due to a new topic.

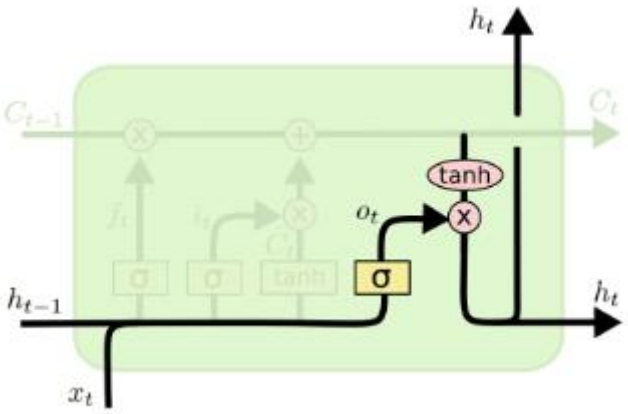


Figure 2. 14 Output information [39]

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t * tanh(C_t) \tag{2.6}$$

2.6 Classification with CNN

CNN consists of a primary part for attribute subtraction and a secondary part for classifying (Figure 2.15).

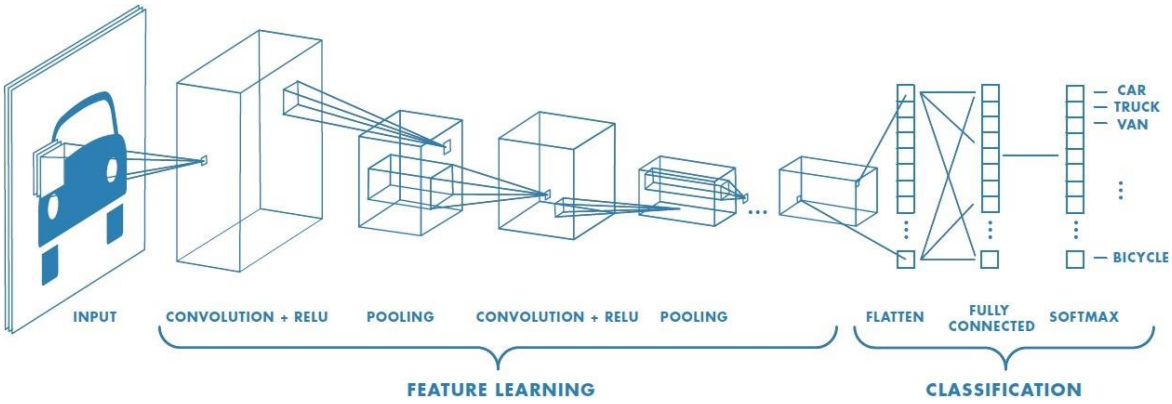


Figure 2. 15 2-D convolution network for image classification [40]

The primary part consists of convolution and pooling layers to extract meaningful attributes (feature extraction) from the raw data. With the convolution layer, feature extraction is performed. This layer is actually a simple filtering process. Some sources use the word kernel instead of a filter. As the depth in the convolution layer increases (Figure 2.16), the complex features of the data are extracted more accurately.

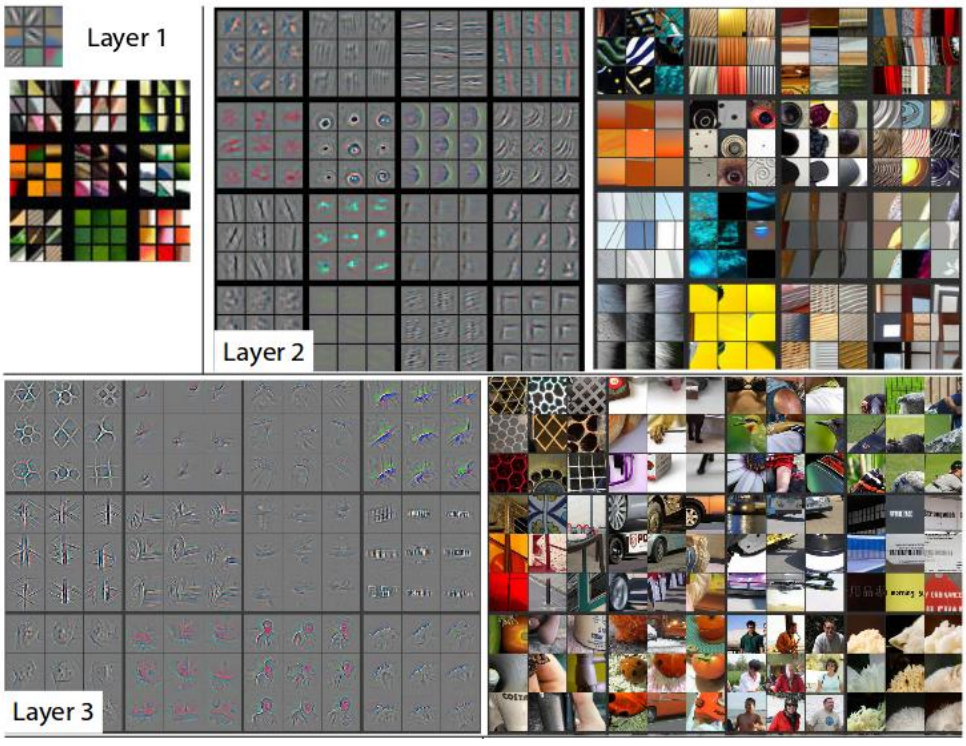


Figure 2. 16 Feature extraction [41]

Figure 2.17 shows the sub-sampling of the pooling layer. This layer minimizes the process load by reducing the attributes which were determined in the convolution layer. However, according to Hinton's capsule theory, due to loss of some important data, performance is a compromise.

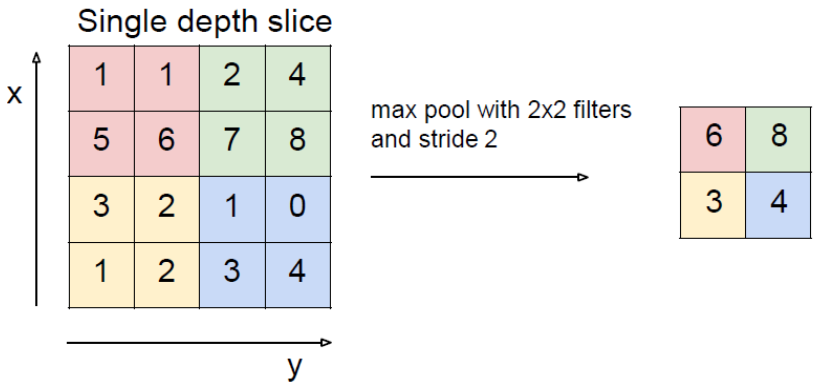


Figure 2. 17 Dimension reduction with pooling layer [35]

In the second part shown in Figure 2.15, the extracted attributes are transferred to the Multi-Layer Perceptron neurons for classification task as in the classic layer of a neural network.

2.6.1 Differences Between 1D CNN and 2D CNN

Although the definitions in the previous section imply that convolution neural networks are more for image classification purposes, the same logic can be applied to one-dimensional convolution neural networks to classify time-dependent data. 1D CNN indicates the same features for 2D and 3D, with only the biggest difference in the direction of the input data and the filters scanning the data.

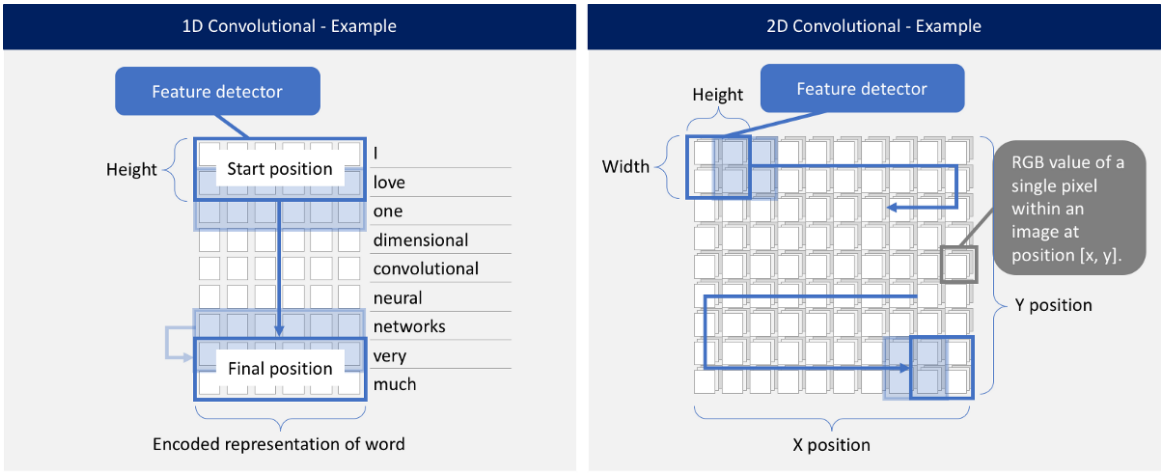


Figure 2. 18 1D versus 2D convolution neural network [42]

In the example for natural language processing (Figure 2.18), each of the 9 words represents a vector. The attribute extractor scrolls in one direction over each word throughout the sentence. In the 2-D convolution sample, the 2x2 size filter moves both horizontally and vertically on the image.

2.6 Neural Network Parameter Optimization

Artificial neural networks incorporate parameters that affect the success of the network. which can be increased by changing the hyper-parameters described in detail below.

2.6.1 Batch Size

The amount of the data set in artificial neural networks increase the success of the learning process. At the same time, the amount of the data set increases the time spent for training and the size of the model obtained at the end of the learning. During the learning phase of the network, learning all the data in the data set at the same time is important in terms of learning time and memory. In each iteration of learning, gradient descent is calculated in the network with backpropagation process to update the weights

accordingly. The higher the number of data in this computation, the more time it takes to complete the process. To solve this problem; the data set is divided into small batches and the training is performed with these small groups.

2.6.2 Learning Rate

In deep learning, the weights are updated with the backpropagation process. In this process, the update of the weights is calculated by finding derivate multiplied by the learning rate and subtracted from the previous weight parameters. Higher learning rates cause an fluctuation, whereas weights updated with a small learning rate prolongs the duration of learning (Figure 2.19).

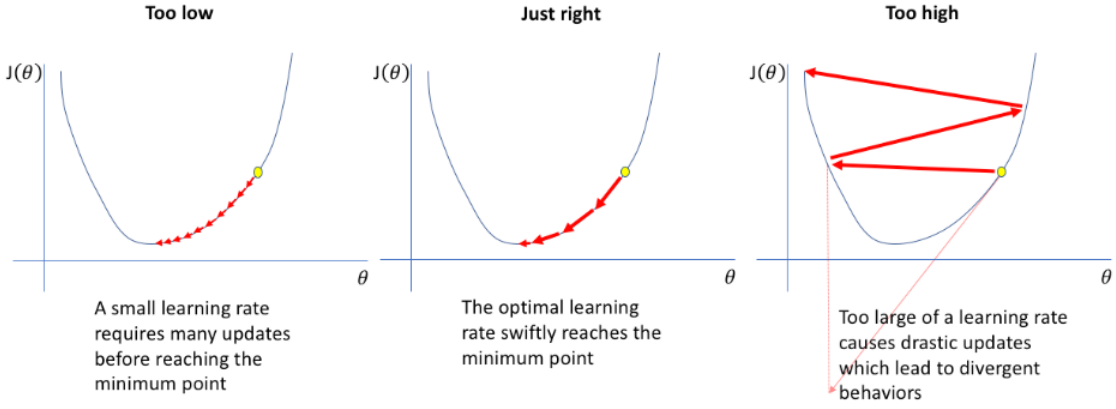


Figure 2. 19 Effect of learning rate on gradient descent [43]

Effect of learning rate is shown in Figure 2.20, where the loss refers to the error and the epoch refers to number of repetitions of the period.

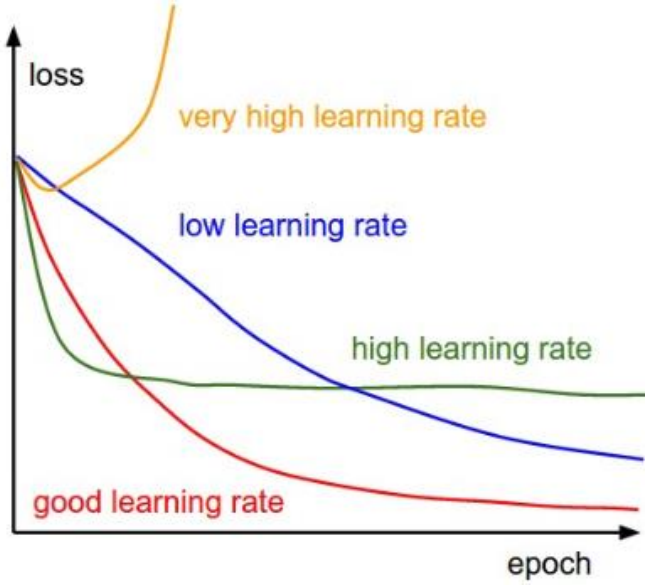


Figure 2. 20 Effect of learning rate on loss [35]

When the training is initiated, the learning rate usually is assigned an initial value of 0.01, and the most appropriate value can then be determined by testing 0.001 and 0.0001 lower values and the effect on accuracy.

2.6.3 Training epoch

While the model is trained, not all of the data are included in the training at the same time. The network takes data in pieces, the first part is trained, the performance of the model is tested and the weights are updated using backpropagation. Then the model is re-trained with the new data set and the weights are updated again. This process is repeated in each training step to calculate the most suitable weight values for the model. Each of these training steps is called an epoch.

As the number of epoch increases, the performance of the model increases significantly. Since performance will increase in very small units after a certain epoch value, then at this point training can be terminated.

2.6.4 Selection of Optimization Algorithm

Learning process in deep learning applications is basically an optimization problem utilizing a variety of methods to find the optimal values for usually nonlinear problems. Stochastic gradient descent, adagrad, adadelta, adam, adamax are widely utilized optimization algorithms in Deep learning applications. There are differences in terms of performance and speed between these algorithms and as an example the performance of different algorithms for MNIST data set is shown in Figure 2.21.

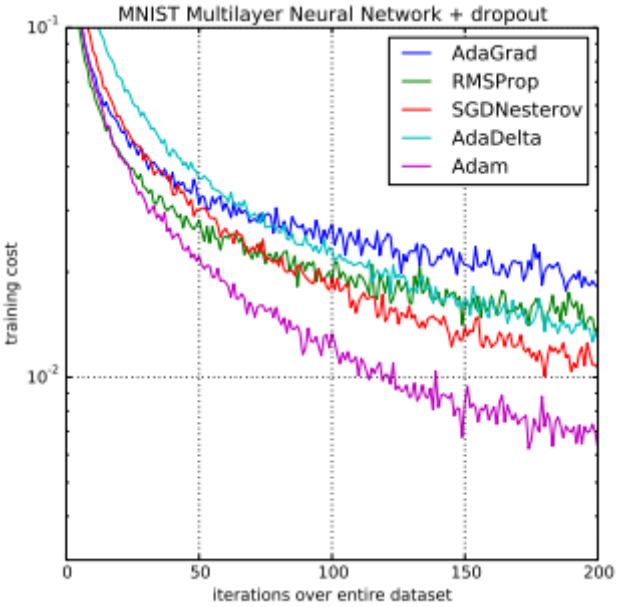


Figure 2. 21 Performance of different optimization algorithms for MNIST data set [44]

Gradient Descent is a general optimization algorithm to find the most suitable solutions for a wide variety of problems. The general idea of Gradient Descent is to set parameters repeatedly to minimize a cost function (Figure 2.22). Learning rate is an important hyperparameter for Gradient Descent. Batch Gradient Descent uses the entire training set to calculate gradients at each step which slows the iteration down when the training set is too large.

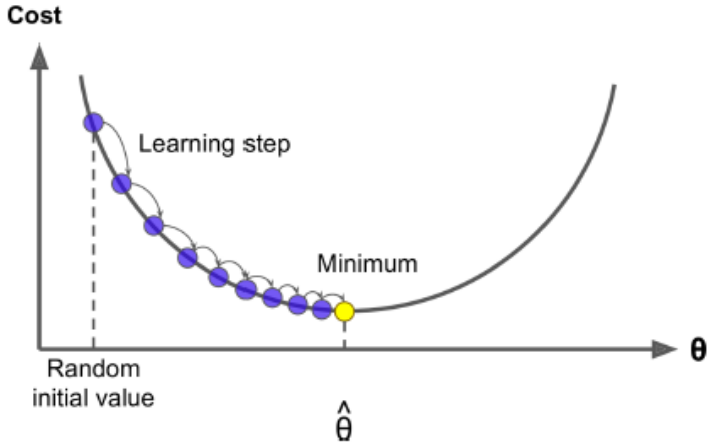


Figure 2. 22 Gradient Descent [45]

The Stochastic Gradient Descent selects a random sample at each step in the training set and calculates gradients based on only one instance. This makes the algorithm faster because there is little data to use in each iteration. It also receives large sets of training data and only one instance should be in the memory during each iteration. Instead of gradually decreasing to the minimum level, the cost function jumps up and down around an average (Figure 2.23). In time, it approaches to the minimum level, but when it gets there it oscillates without settling to a terminal point.

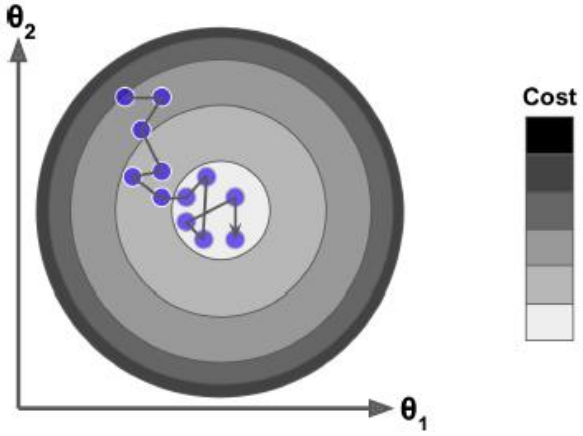


Figure 2. 23 Stochastic Gradient Descent [45]

Adagrad is a gradient-based algorithm. It adapts the learning rate to parameters, makes smaller updates for frequent parameters when making major updates for sparse parameters. It is used for natural language and computer vision problems.

Adadelta is an extension of Adagrad which attempts to reduce the rate of aggressive, tediously reduced learning rate. Adadelta limits the accumulated previous gradient window to a fixed size, instead of accumulating the total previous square gradients.

RMSprop was designed by the Geoffrey Hinton in a Coursera Class lecture [46]. The RMS prop eliminates the need to adjust the learning rate and does this automatically. Moreover, RMSProp selects a different learning rate for each parameter. RMSProp generates its parameter updates using a momentum on a rescaled gradient.

Adam optimizer combines Adagrad and RMSprop. It is an adaptive learning rate method. Rather than adjusting the parameter learning rates to the average initial baseline in the RMSProp, Adam makes updates using the average of the first and second moments of the gradients. Adam is an optimizer which provides fast and good results and it is frequently used in deep learning studies.

2.6.5 Activation Function

Activation functions are used for nonlinear transformation processes in multilayer artificial neural networks. The output of hidden layers is normalized by some activation functions to calculate the gradient in hidden layers (Figure 2.24).

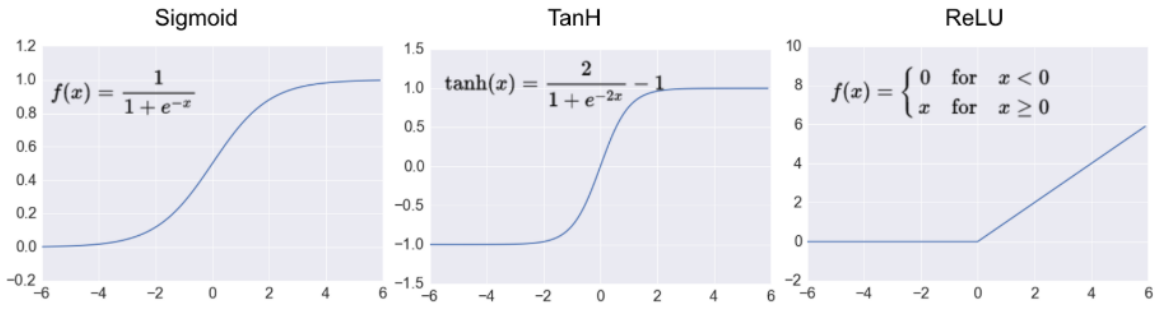


Figure 2. 24 Common activation functions [47]

The sigmoid function range between 0 and 1, tanh function range between -1 and 1, Rectified Linear Unit (ReLU) function ranges from 0 to infinity.

ReLU is currently the most used activation function in the deep learning studies and specifically in almost all Convolutional neural networks studies.

Activation functions are used with gradient descent to update weights. The purpose is to obtain easy derivatives which are shown in Figure 2.25.

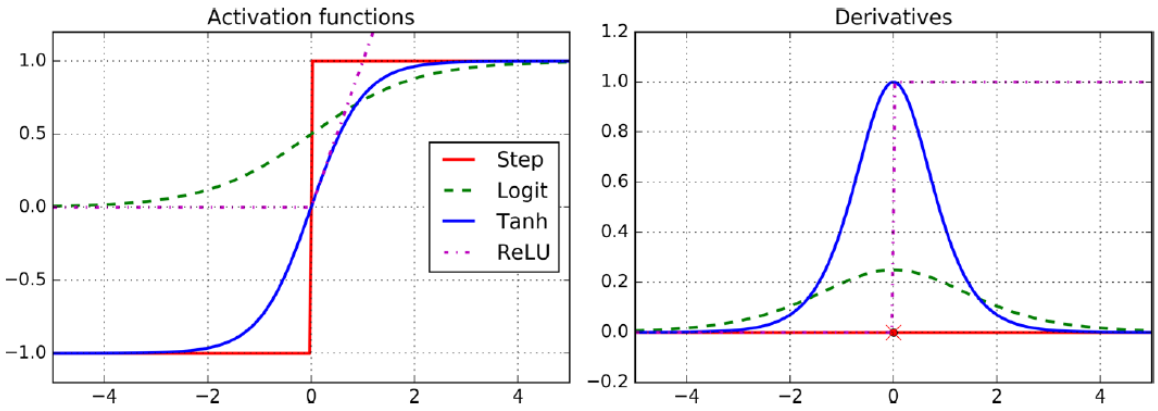


Figure 2. 25 Activation functions and their derivatives [45]

2.6.6 Number of Layers and Number of Neurons

There is no method to determine the number of layers in a network or the number of neurons in the layers. Starting with a single layer or two layers, it is possible to increase the success rate by making changes in hyper parameters. The number of layers and the number of neurons can then be increased as needed.

2.6.7 Overfitting & Underfitting

If neural networks are trained with a small number of epochs, the model cannot learn well, reducing the success of the training and test processes indicating an under-fitting. If trained with too many epochs, model memorizes the training sets leading to overfitting. When this occurs, training set pose higher accuracy (it is almost 1.0 or so close to 1.0), whereas validation loss starts to increase while the test set settle to a lower accuracy. The

accuracy of the network is determined by the ratio of the correctly classified cases to all of the samples.

Accuracy and loss graphics of training and test sets are usually plot against each other to comprehend the overfitting or underfitting problems. If the difference between training accuracy and test accuracy is increasing (Figure 2.26), the model starts overfitting and the training is terminated.

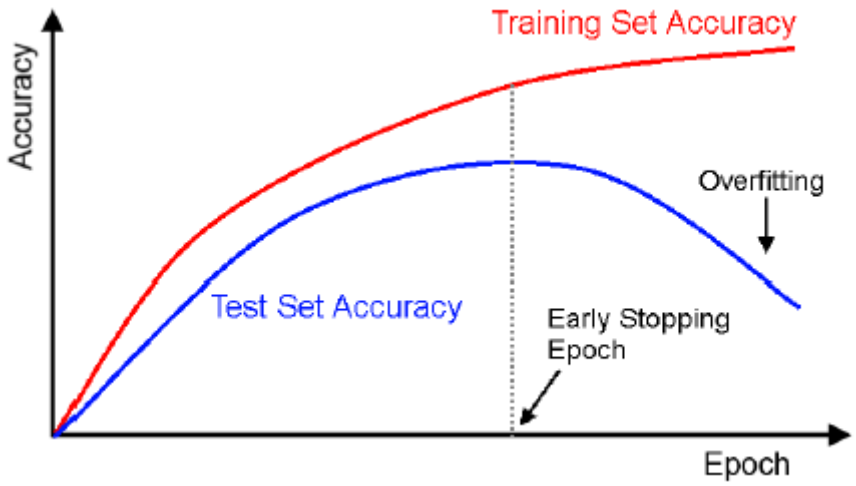


Figure 2. 26 Training and Test accuracy [48]

Similarly, overfitting occurs due to the increase in the value of validation loss (Figure 2.27).

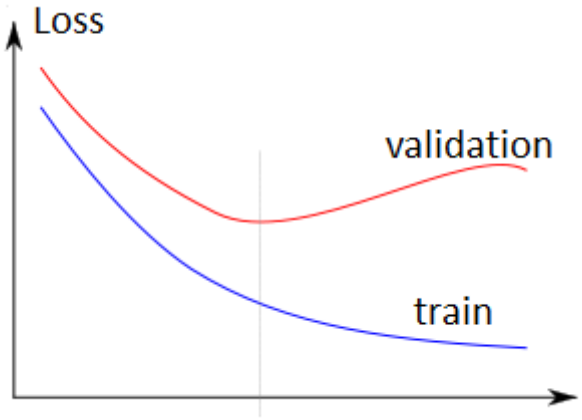


Figure 2. 27 Validation loss [49]

In order to prevent overfitting during the training process, model capacity can be reduced or dropout technique can be used [50].

Dropout technique was proposed by Srivastava, at al. [51] where randomly selected neurons are ignored during training (Figure 2.28). In the dropout technique p hyperparameter is determined ($0 \leq p \leq 1$) to determine the probability of which outputs of layers will be dropped out or which outputs of layers will be kept in the model. 1.0 indicates no dropout, while 0.0 means no outputs from the layer. A common value is a probability of 0.5. Dropout makes the network less sensitive. In this way, the network becomes insensitive to very fine details, such as noise in the data alleviating the problem of overfitting.

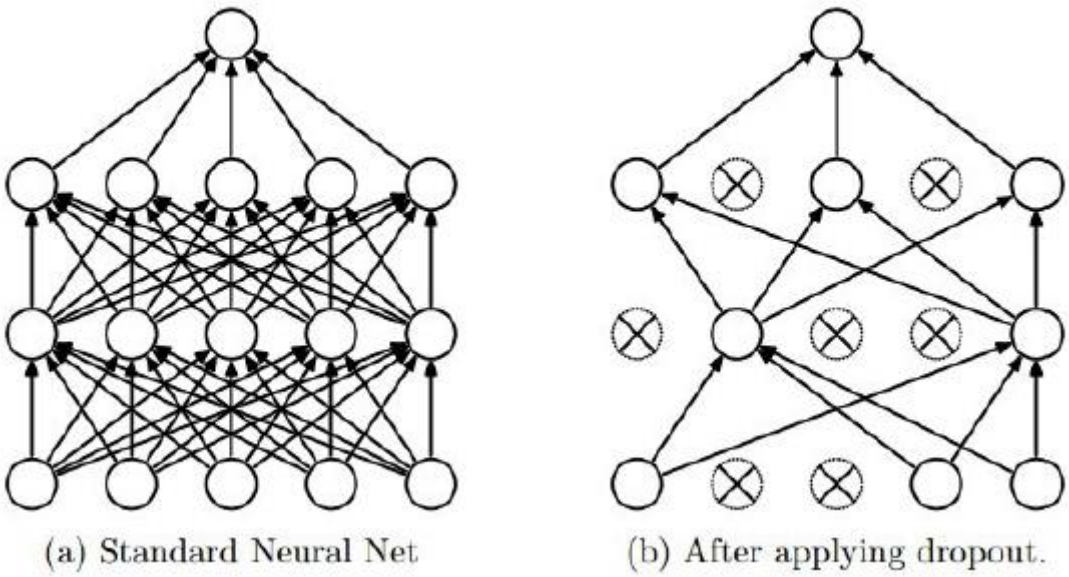


Figure 2. 28 Dropout application [51]

Figure 2.29, figure 2.30 and figure 2.31 show the number of parameters for models with different size using Keras framework which is a high-level neural network API, written in Python and capable of running on top of TensorFlow [52].

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                160016
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 160,305
Trainable params: 160,305
Non-trainable params: 0
```

Figure 2. 29 Baseline model [50]

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 4)                 40004
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 20
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 5
=================================================================
Total params: 40,029
Trainable params: 40,029
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. 30 Smaller model [50]

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 512)               5120512
_________________________________________________________________
dense_7 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 513
=================================================================
Total params: 5,383,681
Trainable params: 5,383,681
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. 31 Bigger model [50]

As shown in Figure 2.32, blue dotted line represents baseline network validation loss, blue straight line represents train loss, yellow dotted line shows smaller networks

validation loss, yellow straight line indicates training loss, green dotted line represent bigger networks validation loss, green straight line represent training loss. As shown in Figure 2.32, bigger network almost begins overfitting after just one epoch. Bigger capacity means less training loss, but also less accuracy for the test set.

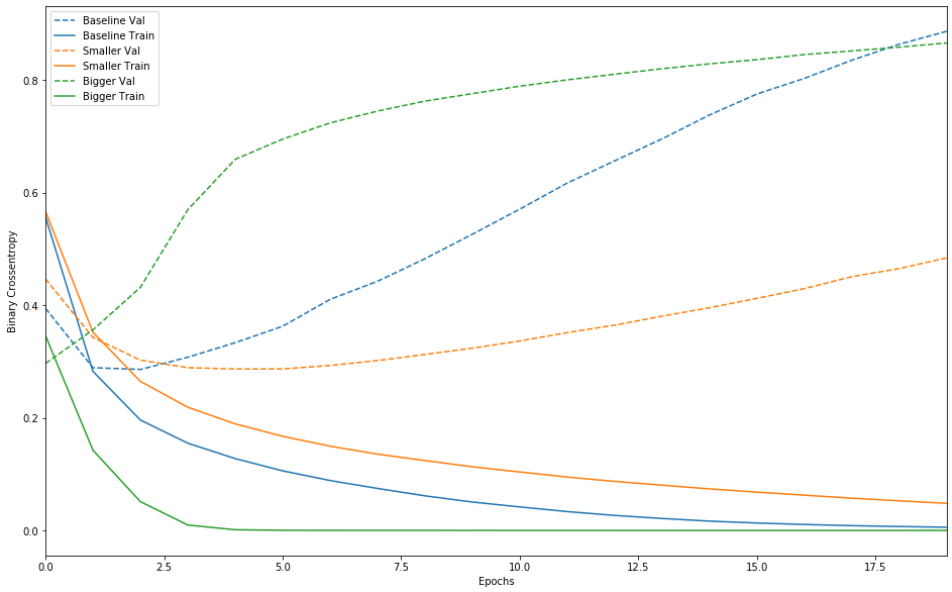


Figure 2. 32 Loss graph of different size networks [50]

2.6.8 Feature Scaling

It was observed that the performance of the network increases when pre-processing is applied to the data before training. Machine learning algorithms do not perform well when numerical inputs are on different scales. There are two common ways in which all features pose the same scale: min-max scaling (normalization) and standardization. Normalizing is, finding the minimum and maximum values of the data and rescaling the data to be between 0 and 1 or -1 to 1 if there are negative values in Eq. 2.7.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.7}$$

Standardization rescales the data to possess a mean (μ) value of 0 and standard deviation (σ) of 1 (unit variance) in Eq. 2.8.

$$X_{changed} = \frac{X - \mu}{\sigma} \tag{2.8}$$

In our study we supplied the data through a standardization procedure before feeding to the network.

CHAPTER 3

3. Results and Discussion

3.1 LSTM Training

There are 12 input parameters and 3 outputs. Input parameters are components as speed, engine speed, fuel consumption. The output parameter is an indicator with a value of 1 for aggressive driving, 2 for mild driving and 3 for gentle driving.

Data from the vehicle were split into two stacks with 603 training series and 402 test series. Each series contains 684 parameters; this indicates that our time series corresponds to 3.5 seconds. Data was standardized before feeding to the network. The process was programmed by using Python language and Numpy library [53].

Adam optimizer for cost minimization, and ReLU for activation function was used in each LSTM neuron. Dropout regularization with a value of 0.5 was added to prevent overfitting. As a result of the testing trials, learning rate was determined as 0.0001 providing the best result at this value. Gradient clipping was added to prevent exploding gradients during back propagation. A gradient threshold was added with a gradient clipping with a value between -1 and 1, and then the gradients that exceed this threshold were minimized to match the norms.

Table 3. 1 The effect hyperparameters and datasets on the success rate

Features	0-3 sec	4-7 sec	8-10 sec	Batch Size	Learning Rate	Hidden Layers	Neurons	Epoch	Accuracy
12	Exist	Exist	Exist	100	10^{-3}	2	32	800	**70.2%**
12	Exist	Exist	Exist	50	10^{-3}	5	20	1000	**69.1%**
9	Exist	Exist	Exist	100	10^{-4}	2	32	1000	**66.6%**
4	Exist	Exist	Exist	100	10^{-3}	4	32	500	**63.6%**
12	Exist	Non	Exist	100	10^{-4}	2	32	400	**91.8%**
12	Exist	Exist	Non	100	10^{-4}	2	32	400	**81.1%**
12	Non	Exist	Exist	100	10^{-4}	2	32	400	**71.2%**

In the initial tests, the network was fed with 0-3 seconds, 4-7 seconds and 8-10 seconds data. The success of the network was improved by altering the hyperparameters as seen in Table 3.1 with a maximum rate of 70%.

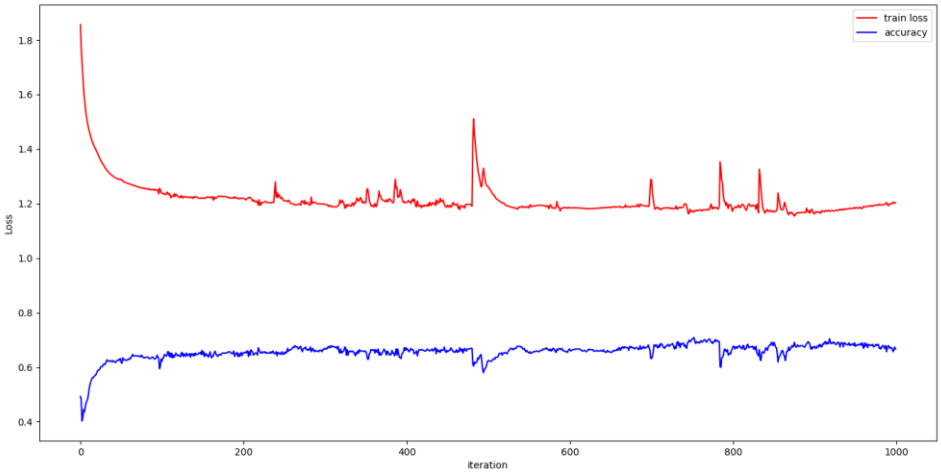


Figure 3. 1 Network performance with three data sets.

In Figure 3.1, red line correspond to training loss, blue line corresponds to training accuracy. In the graph (Figure 3.1) the training loss refers to the difference between the estimated result and the actual result, which the network attempts to reduce during training. The rate indicates the accuracy of the estimates made using the test data denoting that it can classify the driving data supplied to it correctly. As can be seen in the Figure 3.1, the success rate could not be improved to the desired rates. Training the network with different batch size, epoch value, number of hidden layers and number of neurons did not provide any improvement with the success rate.

To overcome this problem, the network was modified to binary classification, by feeding the training datasets in three different combinations as aggressive-gentle, aggressive-mild and mild-gentle. Aggressive - gentle dataset contains 546 train, 234 test set. Aggressive - mild dataset contains 430 train, 185 test set. Mild – gentle dataset contains 577 train, 247 test sets. It was observed that the success rate increased when the network was fed with aggressive and mild dataset or aggressive and gentle dataset.

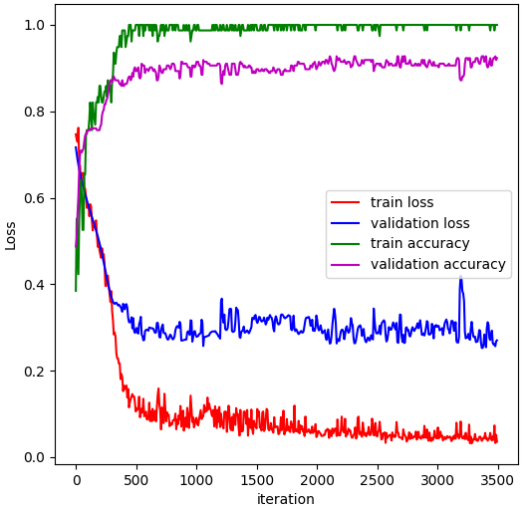
Network was fed with aggressive and gentle driving data yielding a 92.3% accuracy in validation dataset and 88.5% accuracy in test dataset with a 2 layer, 27 neurons network configuration. At the same time, aggressive and mild driving data was supplied to the

network to observe its response to the untrained aggressive and gentle driving data and 75 % accuracy was obtained (Table 3.2).

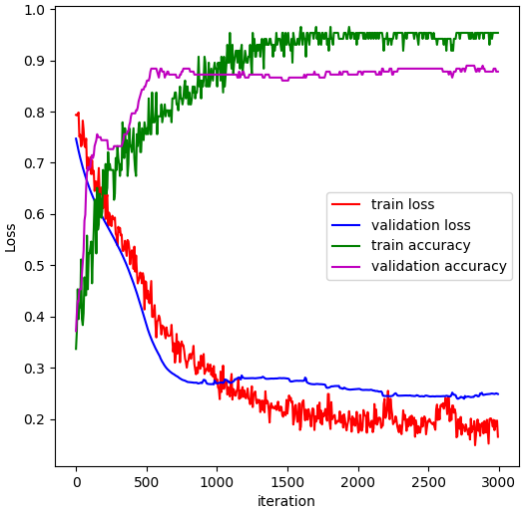
Loss and accuracy graphs of the network which was fed with aggressive and gentle driving data shown in Figure 3.2. As seen in above Figure 3.2.c and Figure 3.2.f, overfitting occurred in the network trained with 1000 epoch.

Table 3. 2 Outputs of the network trained with aggressive & gentle dataset (* overfitting)

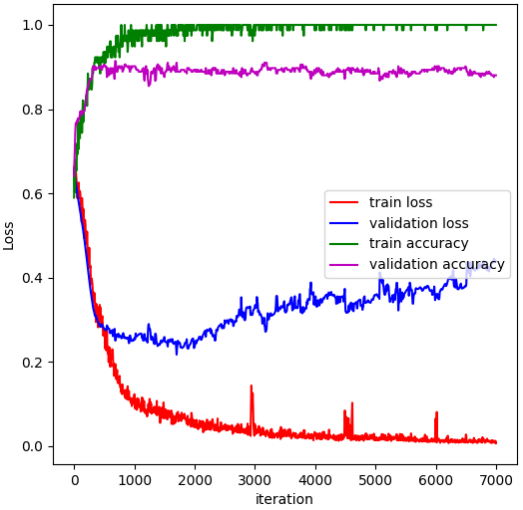
Number of Neurons	Number of Layers	Epoch	Validation Loss	Validation dataset accuracy	Aggressive & gentle test dataset accuracy	Aggressive & mild test dataset accuracy
27	2	500	0.27	92.3%	88.5%	75%
20	1	500	0.249	87.7%	81.2%	72.2%
30*	1	1000	0.438	88%	84.6%	79%
20	2	500	0.311	89.7%	83.3%	75%
20	4	500	0.228	92.3%	78.2%	73%
27*	2	1000	0.423	91.4%	85.8%	79.1%



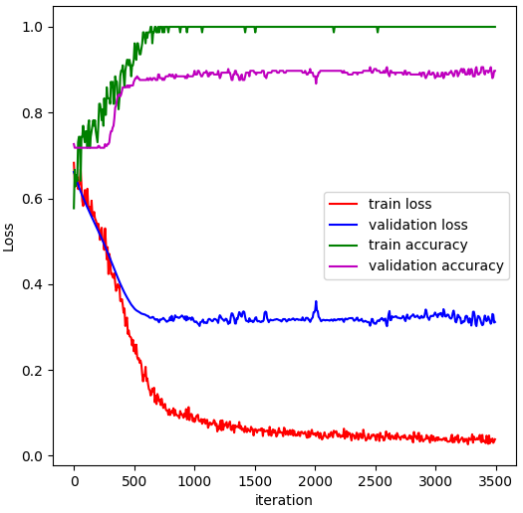
a. 27 neurons, 2 layers



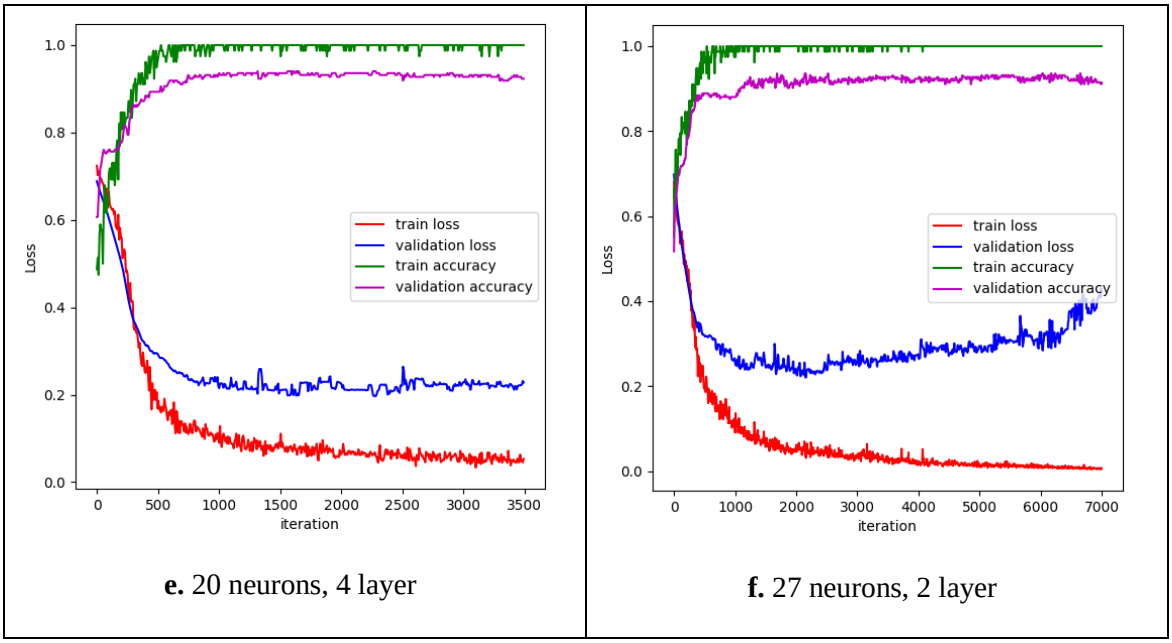
b. 20 neurons, 1 layer



c. 30 neurons, 1 layer



d. 20 neurons, 2 layer



e. 20 neurons, 4 layer

f. 27 neurons, 2 layer

Figure 3. 2 Loss and accuracy graph of network fed with aggressive & gentle

As shown in Figure 3.3, loss and accuracy graph of network trained with 2 layer, 27 neurons and 500 epoch. Figure 3.3 exhibits higher accuracy rates.

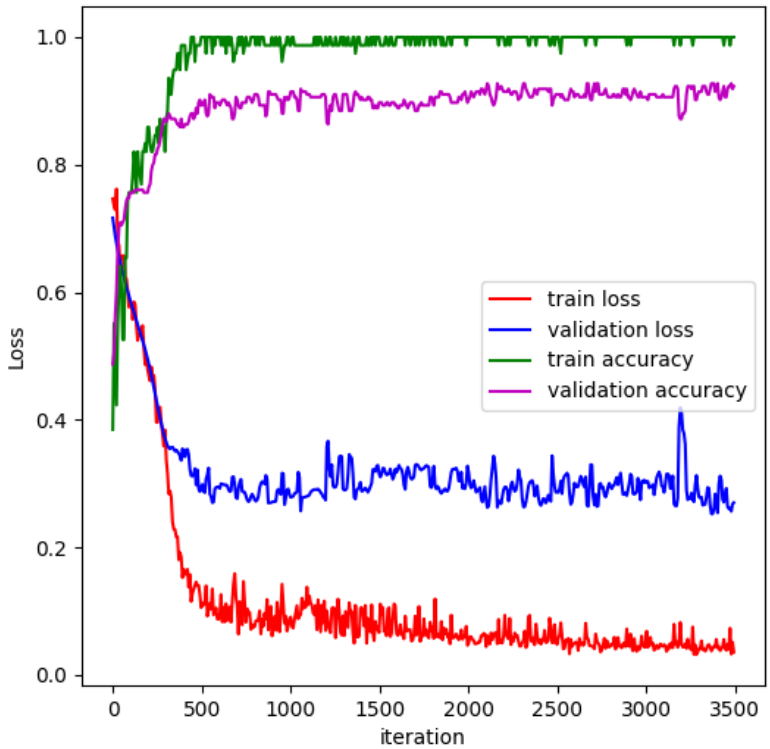
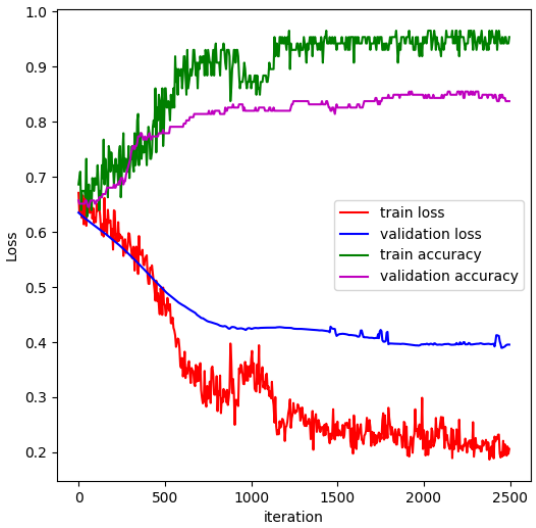


Figure 3. 3 Loss and accuracy graph of network with improved accuracy

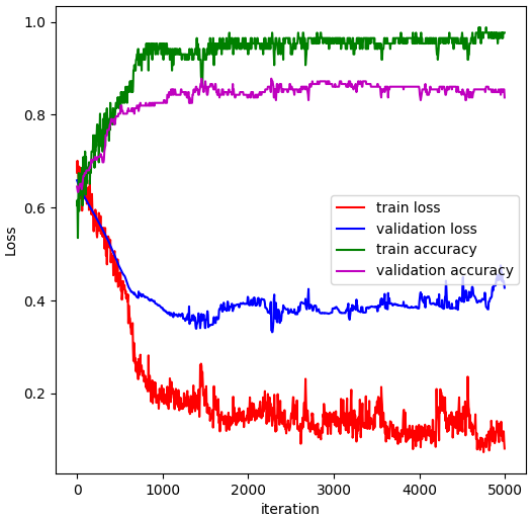
When network was fed with aggressive and mild data together, the outcome was generated with lower accuracy, as shown in Table 3.3. Loss and accuracy graphs of the network which was fed with aggressive and mild driving data as shown in Figure 3.4.

Table 3. 3 Outputs of the network which was trained with aggressive & mild

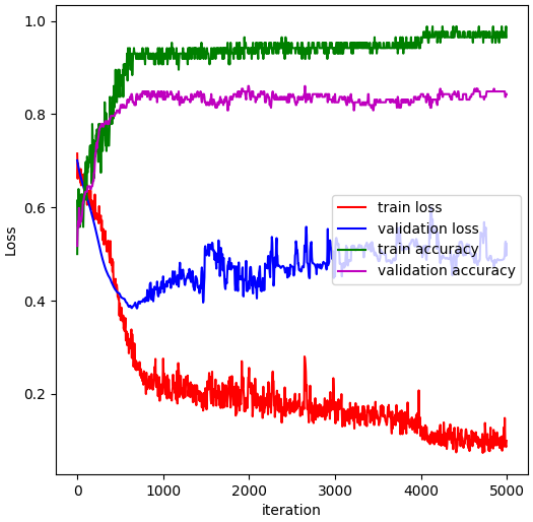
Number of Neurons	Number of Layers	Epoch	Validation Loss	Validation dataset accuracy	Aggressive & mild test dataset accuracy	Aggressive & gentle test dataset accuracy
20	1	500	0.395	83.7 %	61.1%	70.8%
30	1	1000	0.427	85.3%	68%	80.2%
27	2	1000	0.521	84.3%	65.2%	64%
27	2	500	0.395	87.2%	59.7%	64.5%
20	3	500	0.389	84.8%	68%	66%
20	3	1000	0.391	85.2%	69.5%	73.9%



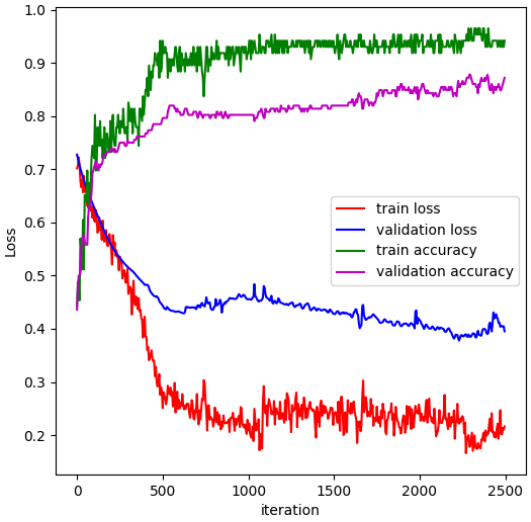
a. 20 neurons, 1 layer



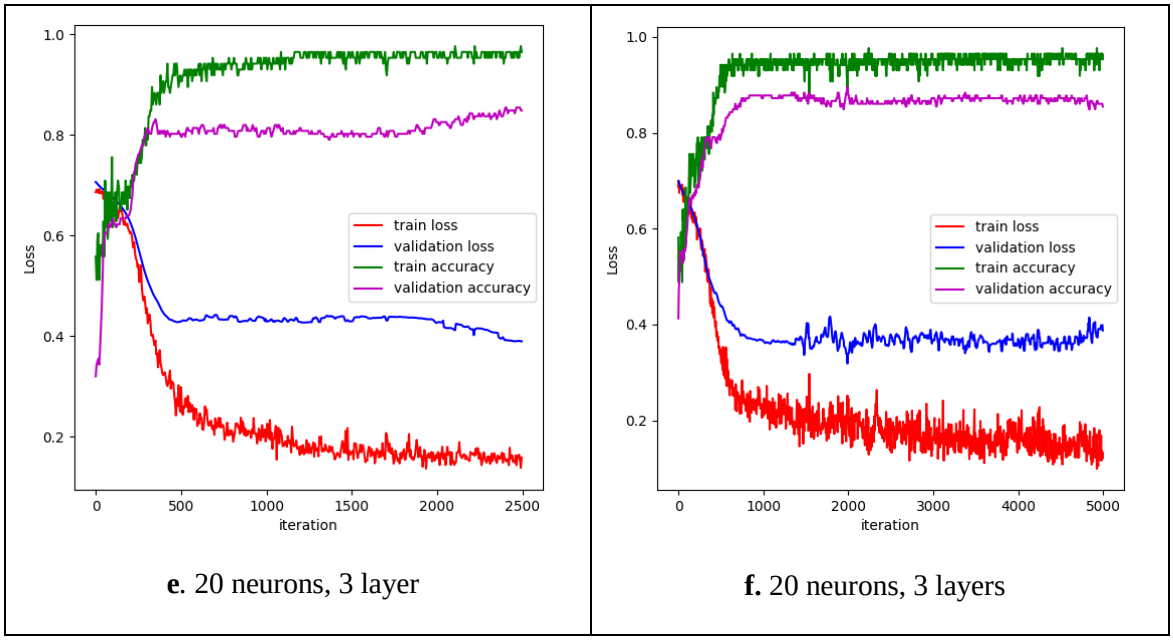
b. 30 neurons, 1 layer



c. 27 neurons, 2 layer



d. 27 neurons, 2 layer



e. 20 neurons, 3 layer

f. 20 neurons, 3 layers

Figure 3. 4 Loss and accuracy graph of network fed with aggressive & mild

Network provided the best results with 85.2% accuracy for validation dataset and 69.5 % for test dataset. At the same time, aggressive and gentle driving data was supplied to the network to observe its response to untrained aggressive and gentle driving data. The network generated low accuracy result with rates of 73.9%. Figure 3.5 exhibits the loss and accuracy graph of network trained with 3 layer, 20 neurons and 1000 epoch.

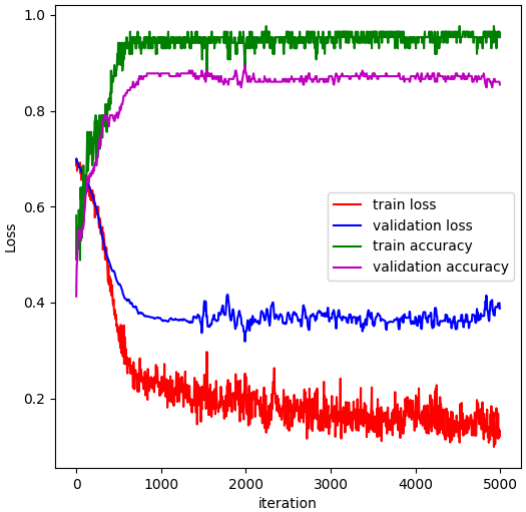


Figure 3. 5 Loss and accuracy of network with improved accuracy

3.2 Results for LSTM

In this study, data collected from the vehicle using LSTM deep learning method is classified as aggressive driving and gentle driving data. The effects of changes in hyperparameters on LSTM network were observed. When the network is fed with data suitable for classification, it was observed that the success rate increased significantly. The most important criteria for correct classification is to feed the network with the appropriate data. It was determined that changes in the number of layers or neurons were inadequate in increasing the success rate when specifically using data not suitable for classification.

Looking at the training results, it was observed that the network had difficulty in distinguish between 8-10 seconds driving behavior and 4-7 seconds driving data. The highest success rate obtained was about 92.3% for the validation dataset and 88.5% for the test dataset, when 0-3 seconds and 8-10 seconds data were combined.

3.3 CNN Training and Results

The convolutional layers are formed using one-dimensional filters moving within the sequence of data while learning during training. In many CNN architectures, as the number of layers increase, the number of filters increase. Each convolution is followed by pooling layers to reduce the sequence length. CNN structure used for driver profiling is shown in Figure 3.6.

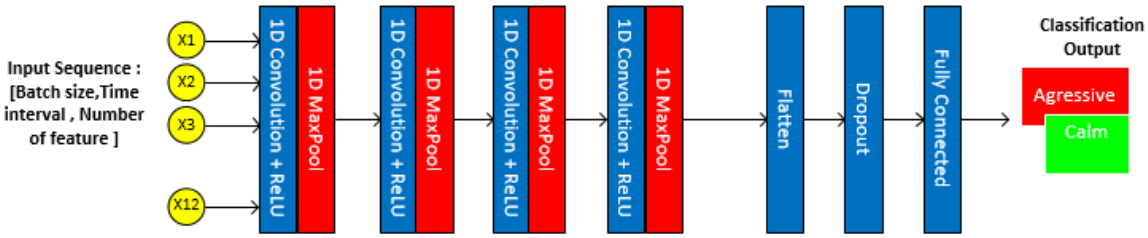


Figure 3. 6 CNN structure used in driver profiling

As mentioned previously, 0-3 sec data was aggressive, 4-7 sec mild and 8-10 sec data was assigned a label for gentle driving. In the previous LSTM study, it was observed that the network could not exceed the success rate of 70% when classifying aggressive driving, mild-driving and gentle driving cases together. However, it was also determined that the reason for the failure was, the difficulty of CNN with distinguishing between mild and gentle driving data. In Figure 3.7 and Figure 3.8, it is obvious that during learning the network cannot exceed the success rate of 73.2%.

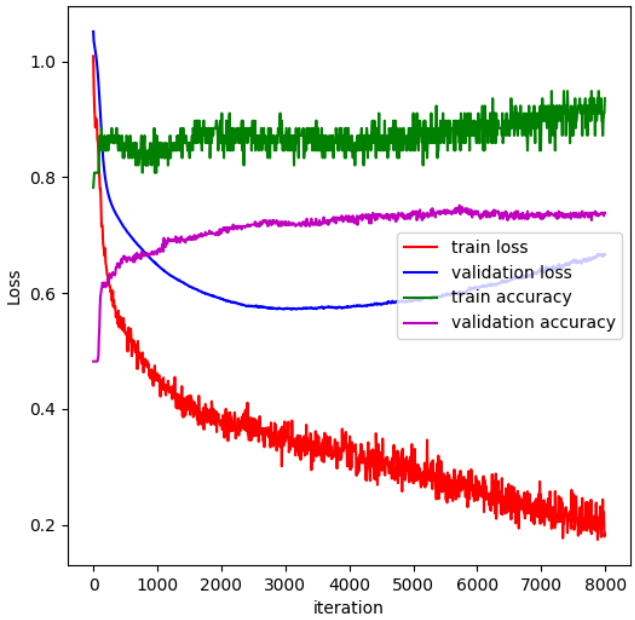


Figure 3. 7 Network performance for aggressive, mild and gentle driving are together

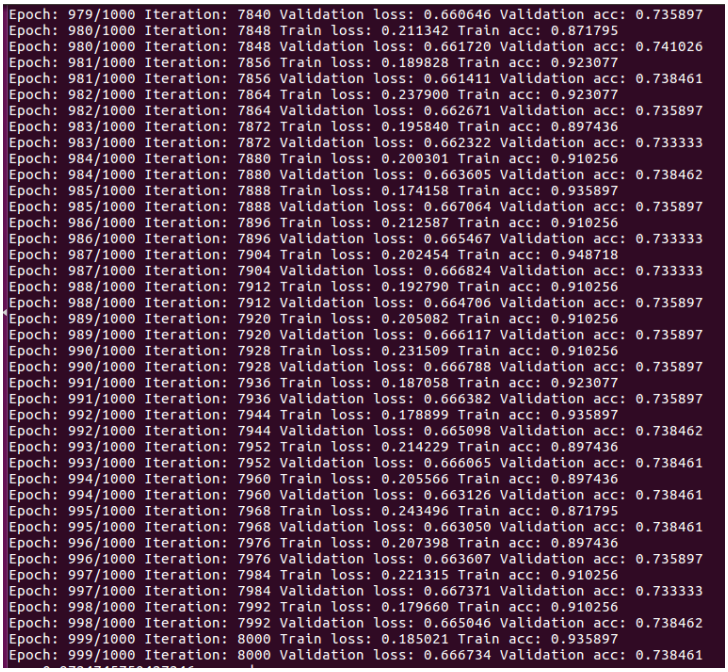


Figure 3. 8 Network success rates for train and test dataset when aggressive, mild and gentle driving are together

For this reason, instead of training three driving data together, the network was modified for binary classification. Only aggressive-mild and aggressive-gentle combinations were studied.

There are 430 samples in the training set for aggressive and mild driving data and 185 samples in the test set. There are 546 samples in the training set for aggressive and gentle driving data and 234 samples in the test set. Learning rate was selected as 0.0001 and the batch size was varied according to the data set. Batch size of 86 for aggressive and mild driving data set and the batch size of 78 for aggressive and gentle driving data were used. The number of layers and epoch were used as optimization parameters and the success rate of the network was increased by the changing these parameters.

As shown in Table 3.4, each convolution layer was followed by pooling layer to reduce sequence length, then proceeding with a dropout regularization of 0.5 to prevent overfitting, and then finalized with a fully connected layer. Adam optimizer for cost minimization, and ReLU for activation function was used in each convolution. 4 layer CNN architecture was trained and tested as shown in Table 3.4. These different CNN

architectures were supplied with different data combinations. Results of training and testing was illustrated in Table 3.5 and Figure 3.9.

Table 3. 4 CNN architecture

1 Layer	2 Layer	3 Layer	4 Layer
Conv1 (filter 24)	Conv1 (filter 24)	Conv1 (filter 24)	Conv1 (filter 24)
Max pooling (2)	Max pooling (2)	Max pooling (2)	Max pooling (2)
Dropout 0.5	Conv2 (filter 48)	Conv2 (filter 48)	Conv2 (filter 48)
Fully Connected	Max Pooling (2)	Max Pooling (2)	Max Pooling (2)
	Dropout 0.5	Conv3 (filter 96)	Conv3 (filter 96)
	Full Connected	Max pooling (2)	Max pooling (2)
		Dropout 0.5	Conv4 (filter 192)
		Fully Connected	Max Pooling (2)
			Dropout 0.5
			Fully Connected

The success rate of classification of aggressive and mild driving data can be seen in Table 3.5. The response of the network to the gentle driving data that it has not fed before can be seen in Figure 3.9. When Table 3.5 is examined, the success of the network in classifying aggressive and mild driving data is less than its success in classifying aggressive and gentle driving data. The highest success rate was achieved with 2 layers and 500 epochs in aggressive and mild driving cases. Network provided 87.7% accuracy for validation dataset and 91% accuracy for the test dataset. Although it is not trained with gentle driving data, it is obvious that it can distinguish between aggressive and gentle

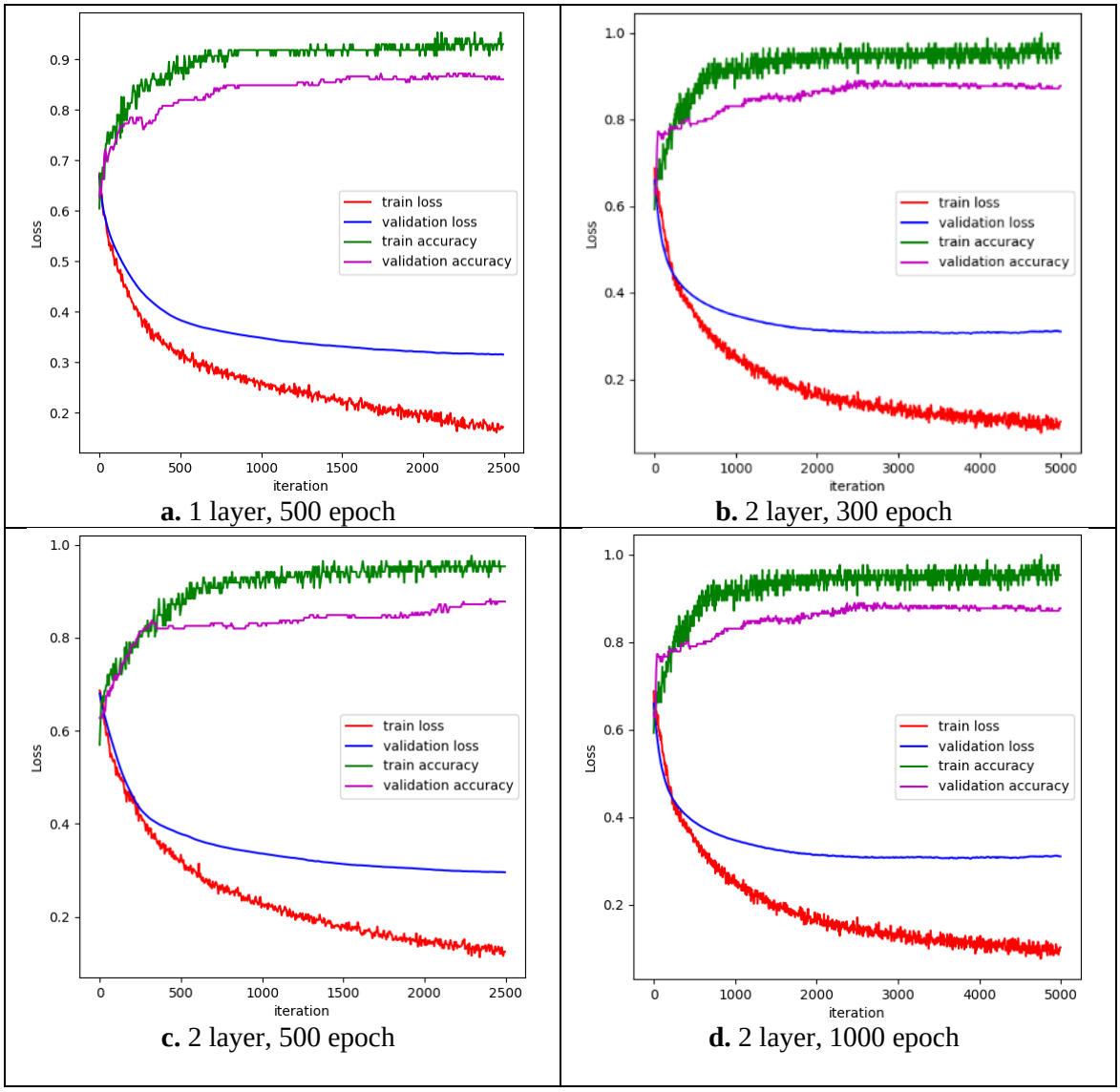
driving much better. It provides 97% accuracy for aggressive-gentle driving dataset. Another crucial issue is that overfitting occurs as the number of layers increases or when the network is trained for a long period of time.

Table 3. 5 Output of the network trained with aggressive & mild dataset (* overfitting)

Number of layers	Epoch	Loss at validation dataset	Accuracy at validation dataset	Aggressive & mild test dataset accuracy	Aggressive & gentle test dataset accuracy
1	500	0.355	83.7%	81.9%	91.6%
2	300	0.308	85.4%	88.8%	97.9%
2	500	0.29	87.7%	91%	97%
2	1000	0.31	87.7%	86.1%	94%
3	300	0.314	85.8%	81.9%	95.8%
3	500	0.294	88.8%	77%	94%
3*	1000	0.513	87.1%	83.3%	93.75%
4*	500	0.395	87.1%	83.3%	95.9%
4*	1000	0.553	86.5%	77.7%	91.6%

As shown in Figures 3.9.g, Figures 3.9.h and Figures 3.9.j, the loss value for the validation data increase after a while and this is an indication of the beginning of the memorization of the network. It is also obvious that when the success of the training attains a value of 1.0, the network begins to memorize the data. While the network exhibits a high success rate in the training data, it becomes difficult to classifying the test data. To avoid

memorization, the layers of the network can be reduced or training can be stopped when the memorization becomes evident.

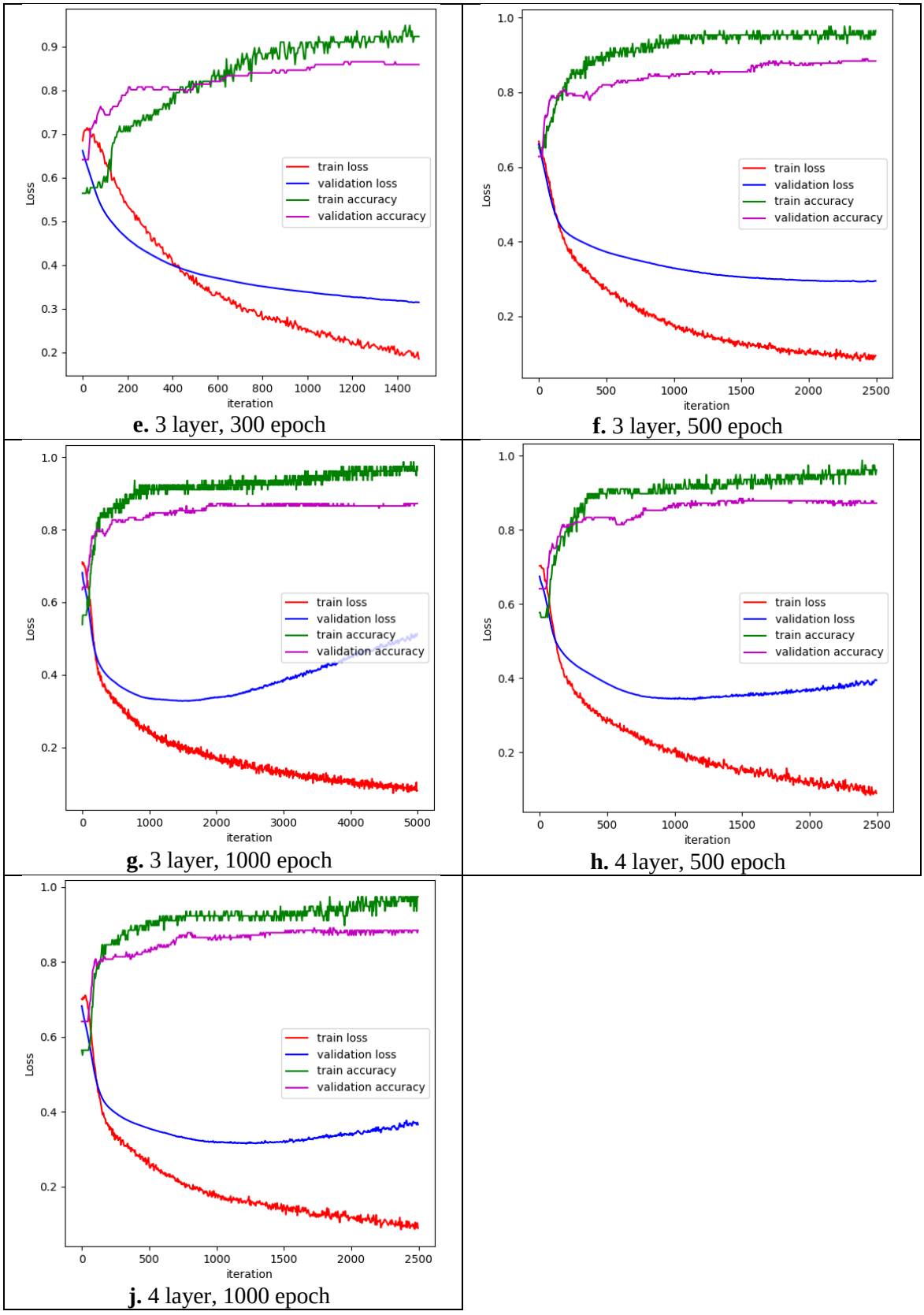


a. 1 layer, 500 epoch

b. 2 layer, 300 epoch

c. 2 layer, 500 epoch

d. 2 layer, 1000 epoch



e. 3 layer, 300 epoch

f. 3 layer, 500 epoch

g. 3 layer, 1000 epoch

h. 4 layer, 500 epoch

j. 4 layer, 1000 epoch

Figure 3. 9 Loss and accuracy of the network trained with aggressive and mild data

Figure 3.10 indicates the highest accuracy rate for the network which was trained with 2 layers and 500 epoch.

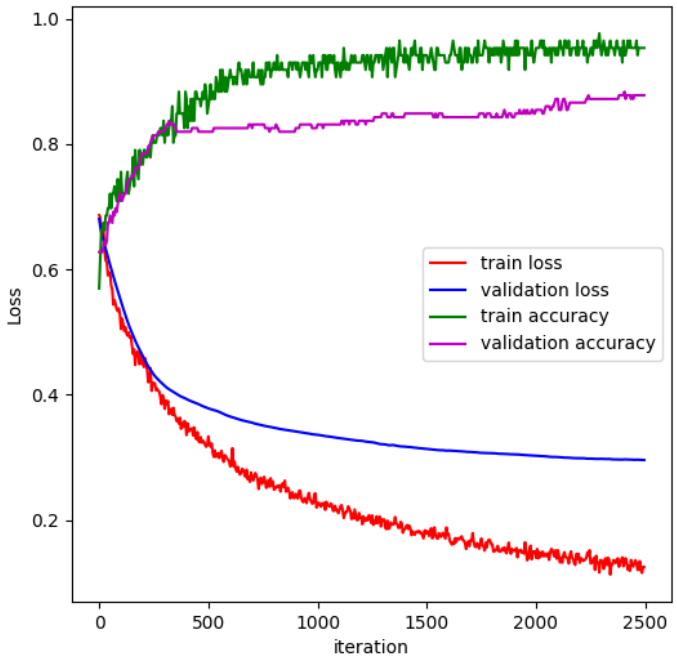


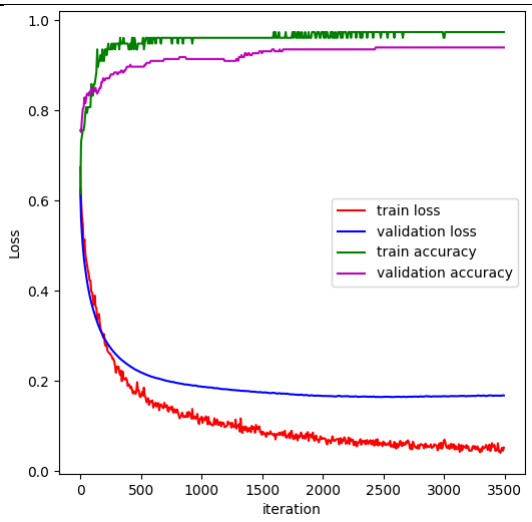
Figure 3. 10 Loss and accuracy graph of network with higher accuracy

Training results for aggressive and gentle driving data are shown in Table 3.6. Network successfully classified aggressive and gentle driving test data. The highest success rate was achieved in 2 layers and 500 epochs in aggressive and gentle driving cases. Network provided 93.5% accuracy for validation dataset and 94.7% accuracy for test dataset. Although it was trained with mild driving data, it is obvious that it could distinguish between aggressive and mild with 88.8% accuracy.

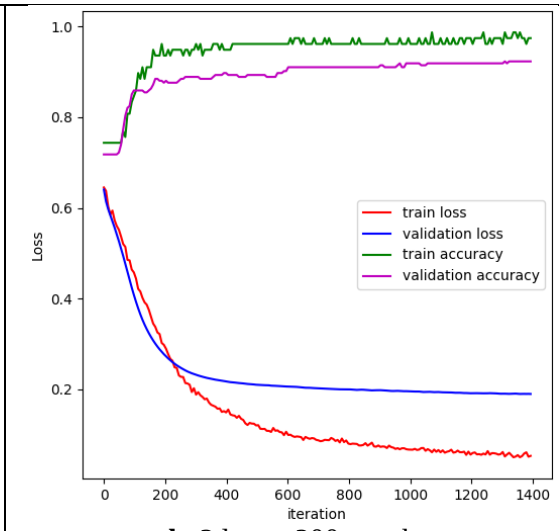
Table 3. 6 Outputs of the network trained with aggressive and gentle data (* overfitting)

Number of layers	Epoch	Loss at validation dataset	Accuracy at validation dataset	Aggressive & gentle test dataset accuracy	Aggressive & mild test dataset accuracy
1	500	0.167	94%	92.7%	88%
2	200	0.189	92.3%	94.7%	81.9%
2	500	0.212	93.5%	94.7%	88.8%
2	1000	0.238	93.1%	91.6%	75%
3	300	0.2	94%	94.7%	80.5%
3	500	0.216	93.5%	95.8%	73.6%
3*	1000	0.327	94.1%	93.7%	69.4%
4*	500	0.290	93%	91.66%	67%
4*	1000	0.616	93.6%	88.5%	70.8%

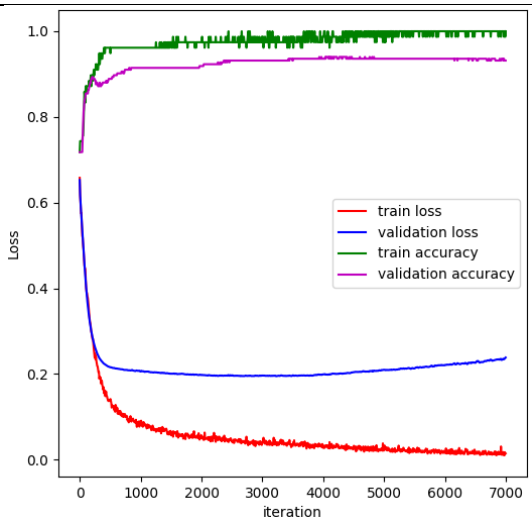
As seen from Figures 3.11.g, Figure 3.11.h and Figure 3.11.j, since the network memorized the data, the loss in the verification data began to increase after a while.



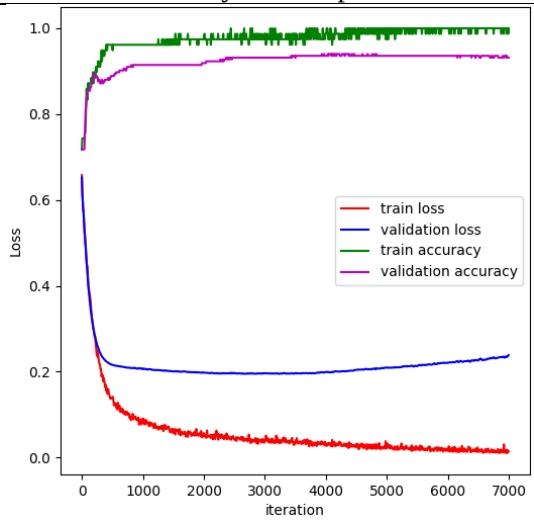
a. 1 layer, 500 epoch



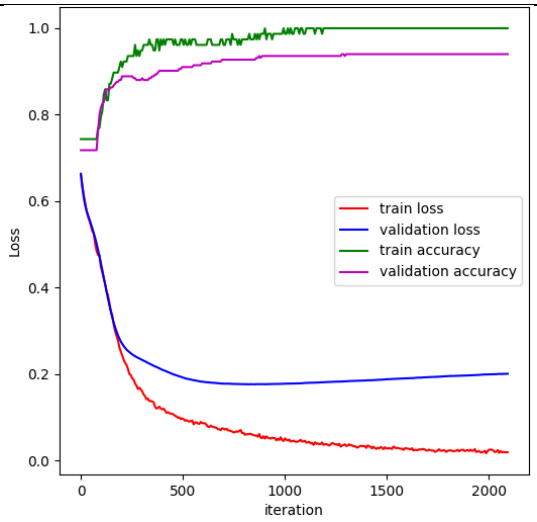
b. 2 layer, 200 epoch



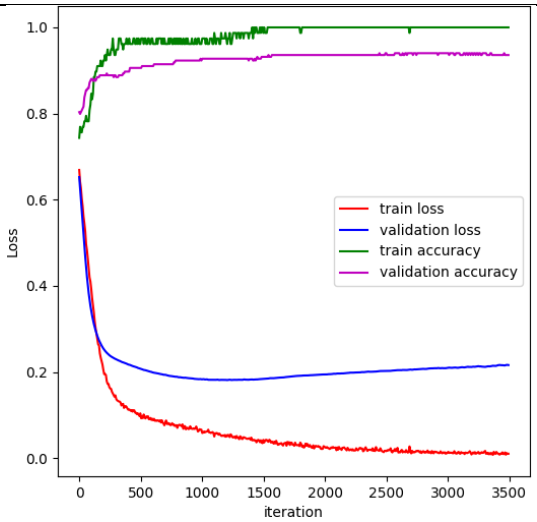
c. 2 layer, 500 epoch



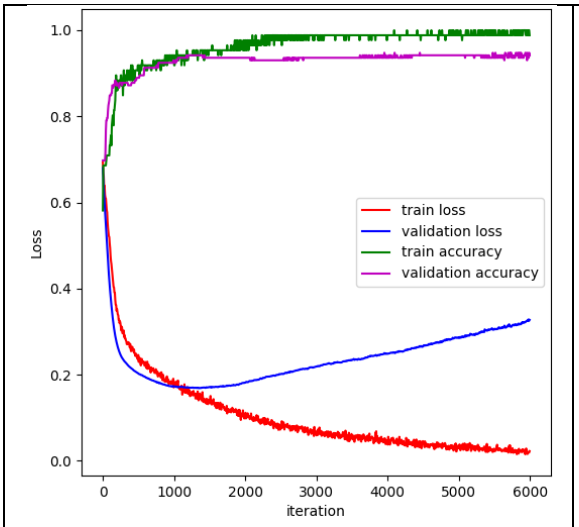
d. 2 layer, 1000 epoch



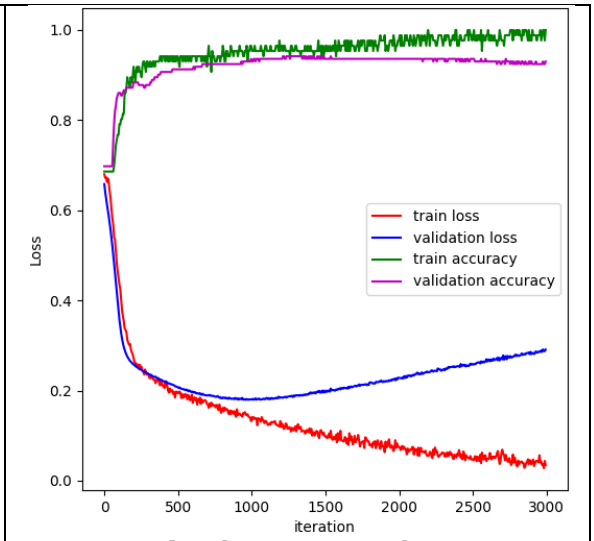
e. 3 layer, 300 epoch



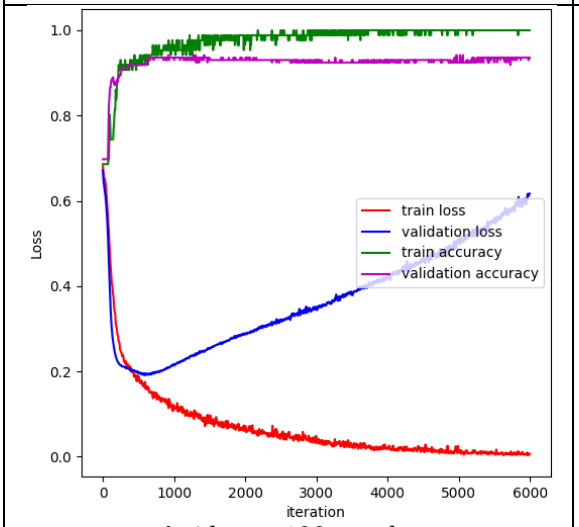
f. 3 layer, 500 epoch



g. 3 layer, 1000 epoch



h. 4 layer, 500 epoch



j. 4 layer, 100 epoch

Figure 3. 11 Loss and accuracy of the network trained with aggressive and gentle data

Figure 3.12 indicates the highest accuracy rate obtained for the network which was trained with 2 layer and 500 epoch.

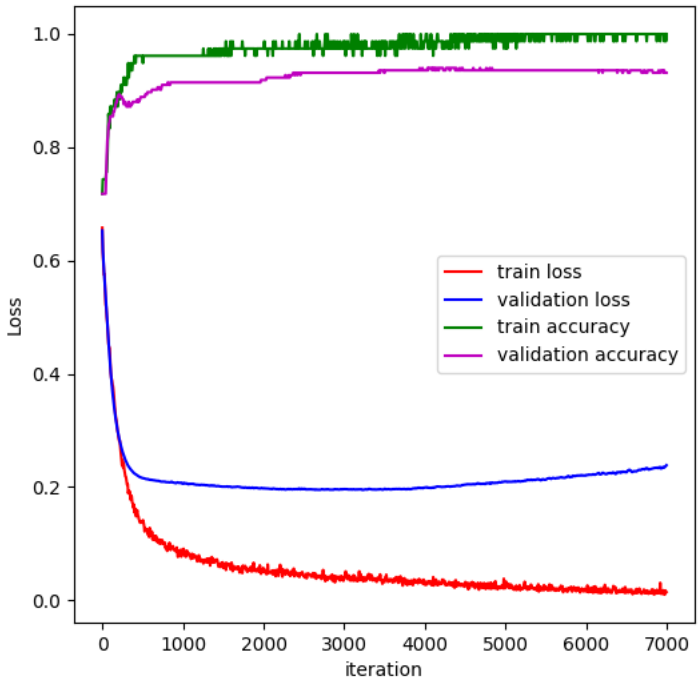


Figure 3. 12 Loss and accuracy of the network with has higher accuracy

3.4 Results for CNN

It was concluded that CNN provides high success in classifying aggressive - mild and aggressive - gentle driving data collected from the vehicle. As a result, 1D CNN provided more success than LSTM. During the tests, it took longer time to train the LSTM and to obtain the classification results from the LSTM-trained network. On the other hand however CNN did output faster results.

CHAPTER 4

4. CONCLUSION

It is very important to determine the driver's behavior for insurance and passenger transportation related companies. Such institutions aim to implement a payment system or sanction according to the behavior of the driver. In the following years, a driver profile will be determined for each driver and a punishment - reward system will be established to prevent traffic accidents. The aim in this study was to designate the behavior profile of the driver with 12 different parameters collected from the vehicle Can Bus communication system. When the studies on driver profiling are examined, it has been observed that there were significant number of studies on machine learning and deep learning. It has been observed that LSTM deep learning method was used in most of the studies which were based on additional time-dependent sensor data. The motivation of this research is based on the fact that there is no study on driver profiling with LSTM deep learning method relying on can-bus vehicle data. At the same time, it has drawn our attention that 1D CNN was used on time-dependent data with prominent achievements. In the literature no study has been encountered on CNN training with only vehicle data which created a motivation to use, 1D CNN for driver profiling.

Aggressive driving, mild driving and gentle driving data were collected from an Otokar vehicle. Collected data was pre-processed, labeled and fed into the neural network. LSTM and CNN models were separately trained and assessed in terms of accuracy in classifying the test data. Both methods demonstrated 70% success rates in classifying three different driving modes. After careful examination, the difficulty was with classification of the mild driving and gentle driving. This is possibly, due to the similarities between mild and gentle driving, where mild and gentle driving data last for 7 sec and 9 sec respectively. Therefore, the network sometimes, could not make the correct classification. For this reason, the network was fed with aggressive - mild driving and aggressive - gentle driving data in binary combinations.

LSTM network was fed with aggressive and gentle driving data, providing 92.3% accuracy for validation dataset and 88.5% accuracy for the test dataset. When this network was fed together with 0-3 sec and 4-7 sec, best results was 85.2% for validation dataset and 69.5 % for the test dataset.

1D-CNN was feed with aggressive and mild driving data, providing 87.7% accuracy for validation dataset and 91% accuracy for the test dataset. When this network was fed with aggressive and gentle driving data, it provided 93.5% accuracy for validation dataset and 94.7% accuracy for the test dataset.

As a result, 1D CNN was more successful than LSTM in classifying the driver profile. CNN achieved high success in extracting important features from the raw data with the filters it contains. Therefore, CNN learned the driver's distinguishing parameters better than LSTM. During the tests, LSTM takes longer time, whereas CNN works with a much faster response. For example, one prediction was lasted 0.5 ms for CNN and one prediction was lasted 25 ms for LSTM method implementation.

References

[1] Ö. HEMDİL, "Karayolu Trafik Kaza İstatistikleri, 2017," 27 June 2018. [Online]. Available: http://www.tuik.gov.tr/PreHaberBultenleri.do?id=27668.

[2] A. A. Malikopoulos and J. P. Aguilar, "Optimization of driving styles for fuel economy improvement," Anchorage, AK, USA, 2012.

[3] J. E. Meseguer, C. K. Toh, C. T. Calafate, J. C. Cano and P. Manzoni, "Drivingstyles: a mobile platform for driving styles and fuel consumption characterization," *Journal of Communications and Networks,* vol. 19, no. 2, pp. 162-168, 2017.

[4] D. I. Tselentis, G. Yannis and E. I. Vlahogianni, "Innovative Insurance Schemes: Pay as/how You Drive," *Science Direct,* vol. 14, pp. 362-371, 27 June 2016.

[5] W. Nai, Y. Chen, Y. Yu, F. Zhang, D. Dong and W. Zheng, "Effective presenting method for different driving styles based on hexagonal eye diagram applied in pay-how-you-drive vehicle insurance," in *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, Hangzhou, China, 2016.

[6] H. Eren, S. Makinist and A. Yilmaz, "Estimating driving behavior by a smartphone," in *2012 IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, 2012.

[7] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, 2011.

[8] A. D. Alvarez, F. S. Garcia, J. E. Naranjo, J. J. Anaya and F. Jimenez, "Modeling the Driving Behavior of Electric Vehicles Using Smartphones and Neural

Networks," *IEEE Intelligent Transportation Systems Magazine* , vol. 6, no. 3, pp. 44-53, 21 July 2014.

[9] E. Carvalho, B. V. Ferreira, J. Ferreira, C. d. Souza, V. H. Carvalho, Y. Suhara, A. S. Pentland and P. Gustavo, "Exploiting the use of recurrent neural networks for driver behavior profiling," in *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA, 2017.

[10] B. Nirmali, S. Wickramasinghe, T. Munasinghe, C. R. Amalraj and D. H. Bandara, "Vehicular data acquisition and analytics system for real-time driver behavior monitoring and anomaly detection," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, Peradeniya, Sri Lanka, 2017.

[11] Ö. Kumtepe, E. Yüncü and G. B. Akar, "A multimodal approach for aggressive driving detection," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, Zonguldak, Turkey, 2016.

[12] A. E. B. Masri, H. Artail and H. Akkary, "Toward self-policing: Detecting drunk driving behaviors through sampling CAN bus data," in *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, Ras Al Khaimah, United Arab Emirates, 2017.

[13] J. Carmona, M. A. Miguel, D. Martin, F. Garcia and A. Escalera, "Embedded system for driver behavior analysis based on GMM," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 2016.

[14] M. V. Ly, S. Martin and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, QLD, Australia, 2013.

[15] G. C. M. Quintero, J. A. Lopez and J. M. Rua, "Intelligent erratic driving diagnosis based on artificial neural networks," in *2010 IEEE ANDESCON*, Bogota, Colombia, 2010.

[16] J. Morton, T. A. Wheeler and M. J. Kochenderfer, "Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior," *IEEE Transactions on Intelligent Transportation Systems,* vol. 18, no. 5, pp. 1289-1298, 13 September 2016.

[17] A. Klusek, M. Kurdziel, M. Paciorek, P. Wawryka and W. Turek, "Driver Profiling by Using LSTM Networks with Kalman Filtering," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 2018.

[18] S. M. Lee, S. M. Yoon and H. Cho, "Human activity recognition from accelerometer data using Convolutional Neural Network," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, South Korea, 2017.

[19] S. Kiranyaz, T. Ince and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks," *IEEE Transactions on Biomedical Engineering,* vol. 63, no. 3, 2015.

[20] D. Li, J. Zhang, Q. Zhang and X. Wei, "Classification of ECG signals based on 1D convolution neural network," in *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Dalian, China, 2017.

[21] X. Zhai and C. Tin, "Automated ECG Classification Using Dual Heartbeat Coupling Based on Convolutional Neural Network," *IEEE Access,* vol. 6, 2018.

[22] M. Deshmane and S. Madhe, "ECG Based Biometric Human Identification Using Convolutional Neural Network in Smart Health Applications," in *2018 Fourth*

International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, India, 2018.

[23] Y. Chen and Y. Xue, "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Kowloon, China, 2015.

[24] T. Zebin, P. J. Scully and K. B. Ozanyan, "Human activity recognition with inertial sensors using a deep learning approach," in *2016 IEEE SENSORS*, Orlando, FL, USA, 2016.

[25] M. Z. Uddin and M. M. Hassan, "Activity Recognition for Cognitive Assistance Using Body Sensors Data and Deep Convolutional Neural Network," *IEEE Sensors Journal*, 2018.

[26] A. H. Fakhrulddin, X. Fei and H. Li, "Convolutional neural networks (CNN) based human fall detection on Body Sensor Networks (BSN) sensor data," in *2017 4th International Conference on Systems and Informatics (ICSAI)*, Hangzhou, China, 2017.

[27] Y.-T. Pang , S.-W. Syu, Y.-C. Huang and B.-H. Chen , "An Advanced Deep Framework for Recognition of Distracted Driving Behaviors," in *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, Nara, Japan, 2018.

[28] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis and F.-Y. Wang, "Driver Activity Recognition for Intelligent Vehicles: A Deep Learning Approach," *IEEE Transactions on Vehicular Technology(in press)*, 2019.

[29] S. Yan, Y. Teng, J. S. Smith and B. Zhang, "Driver Behavior Recognition Based on Deep Convolutional Neural Networks," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Changsha, China, 2016.

[30] Z. Gao, Y. Liu, J. Y. Zheng, R. Yu, X. Wang and P. Sun, "Predicting Hazardous Driving Events Using Multi-Modal Deep Learning Based on Video Motion Profile and Kinematics Data," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Hawaii, USA, 2018.

[31] Q. Wang, Y. Liu, J. Liu, Y. Gu and S. Kamijo, "Critical Areas Detection and Vehicle Speed Estimation System Towards Intersection-Related Driving Behavior Analysis," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 2018.

[32] S. H. Sanchez, R. F. Pozo and L. A. H. Gomez, "Estimating Vehicle Movement Direction from Smartphone Accelerometers Using Deep Neural Networks," *sensors*, 2018.

[33] Otokar. [Online]. Available: https://www.otokar.com.tr/tr. [Accessed 19 May 2019].

[34] "Tensorflow," [Online]. Available: https://www.tensorflow.org/. [Accessed 19 May 2019].

[35] A. Karpathy, "CS231n: Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/. [Accessed 19 May 2019].

[36] W. D. Mulder, S. Bethard and M. F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Computer Speech & Language*, vol. 30, no. 1, 17 September 2014.

[37] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, 1997.

[38] S. Yan, "Understanding LSTM and its diagrams," [Online]. Available: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714. [Accessed 25 May 2019].

[39] C. Colah, "Understanding LSTM Networks," [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 19 May 2019].

[40] [Online]. Available: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html. [Accessed 21 May 2019].

[41] A. Deshpande, 21 May 2019. [Online]. Available: https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html.

[42] N. Ackermann. [Online]. Available: https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf. [Accessed 22 May 2019].

[43] J. Jordan, "Setting the learning rate of your neural network.," [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Accessed 19 May 2019].

[44] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, San Diego, 2015.

[45] A. Geron, Hands-On Machine Learning with Scikit-Learn and Tensorflow, O'Reilly, 2017.

[46] T. Tieleman, "Neural Networks for Machine Learning," [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/. [Accessed 19 May 2019].

[47] A. Moujahid, "A Practical Introduction to Deep Learning with Caffe and Python," [Online]. Available: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/. [Accessed 19 May 2019].

[48] [Online]. Available: https://deeplearning4j.org/docs/latest/deeplearning4j-nn-early-stopping. [Accessed 19 May 2019].

[49] "Another look into overfitting," [Online]. Available: https://medium.com/randomai/another-look-into-over-fitting-33e15b044a5e. [Accessed 19 May 2019].

[50] "Explore overfitting and underfitting," [Online]. Available: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit. [Accessed 19 May 2019].

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from overfitting," *Journal of Machine Learning Research 15 (2014)*, pp. 1929-1958.

[52] "Keras: The Python Deep Learning library," 19 May 2019. [Online]. Available: https://keras.io/.

[53] "scientific computing library for Python," [Online]. Available: https://www.numpy.org/. [Accessed 25 May 2019].