Identifying Onboard Diagnosis-Relevant Errors of an SCR System, using Neural Networks

Master Thesis

Submitted in Fulfilment of the

Requirements for the Academic Degree

M.Sc. Automotive Software Engineering

Dept. of Computer Science

Chair of Computer Engineering

Submitted by: Anoop Vasudevan

Student ID: 423904

Date: 25.05.2018

Supervising tutor: Dr. Ariane Heller

Dr. André Bojahr (IAV, Berlin)

Acknowledgement

I would like to take the opportunity to thank everyone who have guided and supported me in the course of this master thesis. I am highly grateful to Prof. Dr. Wolfram Hardt who has always supported me in the course of my master's programme. I would like to thank Prof. Dr. Wolfram Hardt and Dr. Ariane Heller for all the guidance given for the successful completion of this master thesis.

I acknowledge the fact that this entire master thesis would not have been possible without the opportunity given to me by IAV GmbH, Berlin. While at IAV, I was given all the necessary support and guidance at every phase of this master thesis by my team. I would like to especially thank Dr. André Bojahr, who was my advisor at IAV. Without his guidance and encouragement, this master thesis would not have been possible.

Finally, I would like to thank everyone who have supported me directly or indirectly throughout the course of this master thesis.

Abstract

The selective catalytic reduction system (SCR) is responsible for the reduction of harmful NO_x gases in vehicles. A faulty SCR system causes higher emissions of NO_x gases, which results in multiple environmental and health hazards. This important role played by the SCR system is the reason for selecting this topic for my thesis.

The goal of this thesis is to evaluate whether a neural network can be implemented for the easy analysis and classification of the various possible errors that could tamper with the proper functioning of the SCR system. The data required for the learning algorithm is already available with IAV. In this thesis report, I shall discuss an approach for the identification of SCR errors using a neural network on the already available data.

The results of this thesis could have a positive impact on the environment as well as on the health of the people. The results of this thesis would be of key importance for the automotive field and for industries that use diesel.

Keywords: Selective Catalytic Reduction, Supervised Classification, Neural Network, Error Diagnosis, NO_x reduction

Content

List of Figures

List of Tables

List of Abbreviations

SCR	Selective Catalytic Reduction
SNCR	Selective Non-Catalytic Reduction
OBD	On-Board Diagnostic
NO_x	Oxides of Nitrogen
ECU	Engine Control Unit
DEF	Diesel Exhaust Fluid
PM	Particulate Matter
IDE	Integrated Development Environment
GPU	Graphics Processing Unit

1 Introduction

This chapter gives a brief overview of the topic ‘Identifying Onboard Diagnosis-Relevant Errors of an SCR System, using Neural Networks’.

Section 1.1 explains my motivation for this thesis.

Section 1.2 explains the problem statement.

Section 1.3 explains the organization of the thesis report.

1.1 Motivation

Air pollution is a major cause of concern all over the world. As per the World Health Organization, around 36% of deaths from lung cancer, 34% deaths from stroke and 27% deaths from heart diseases are due to air pollution [1].

85% of air-pollution is man-made. Of this, 60% is caused by the exhaust from motor vehicles [4].

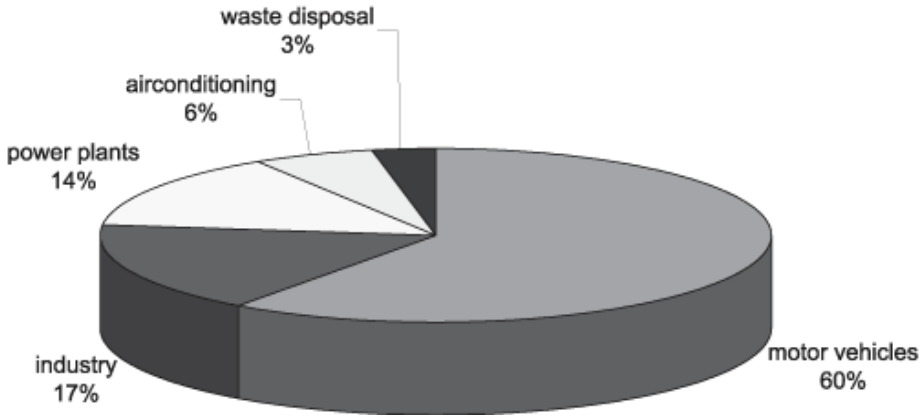


Figure 1.1 Man-made causes of Air Pollution

With the growing demand for vehicles and the growing demand for cheaper fuel, the sale of diesel cars has increased to a point where half the vehicles on the roads now run on diesel. Though cheaper than petrol, the burning of diesel causes higher pollution than the burning of petrol [3]. To reduce the pollution caused by the exhaust from diesel vehicles, the Selective Catalytic Reduction (SCR) system has been in use since the 1980s [2].

1.2 Problem Statement

Like any system, the SCR system also fails from time to time. Any fault to any part of the system leads to the faulty functioning of the SCR system. A faulty SCR system lowers the NO_x reduction. These NO_x gases get mixed with the atmosphere and cause multiple problems such as [2]-

- Visibility Impairment
- Respiratory Diseases
- Acid Rains
- Water Quality Deterioration

Due to these problems, the proper diagnosis and fixing of the SCR system is of great importance.

Due to multiple sensors and inlets to the SCR system, sometimes, the diagnosis process takes a lot of time. In certain cases, the diagnosis team fails to detect the exact error in spite of spending hours on the diagnosis process. In these situations, the entire SCR system is replaced [9].

Multiple attempts have been made to pinpoint the exact error of the SCR using traditional programming methods. However, due to spikes in values, upwards or downwards, and various other reasons, these attempts were never successful [11].

The reason for this thesis is three-fold-

- To attempt a machine learning algorithm to pinpoint errors, where traditional programming has failed
- To lower the time taken for SCR fault diagnosis.
- To ensure that only the faulty part is changed, and that the other parts are kept intact.

1.3 Organization of Thesis

- Chapter 2 gives a brief overview of the current scenario of the problem. This chapter explains how the diagnosis of errors is currently done.
- Chapter 3 gives an overview of the SCR System (and it's possible errors) and the On-Board Diagnostics System. This chapter also gives an overview of the programming language and programming technique used.
- Chapter 4 explains the concept that is implemented as part of this thesis. This chapter explains the algorithm in detail.
- Chapter 5 explains the various steps of implementation in the suggested approach.
- Chapter 6 gives an overview of the results of the implementation. This chapter compares the current approach to the implemented approach.
- Chapter 7 gives a summary of the work done. This chapter also explains about the limitations of the work done, and how the work can be improved.

2 State of the Art

This chapter gives a brief overview of the current scenario of the problem.

Section 2.1 explains how the diagnosis of the SCR system is currently done.

Section 2.2 explains how the issue is notified to the driver, or the user of the system.

2.1 Diagnosis Process

The OBD system of the vehicle checks multiple components of the vehicle, of which the following are part of [19]-

Component	Parameter
SCR NO_x Catalyst	• Conversion efficiency • Catalyst aging • Reductant Tank level • Injection feedback control
NO_x adsorber	• NO_x adsorbing capability • Feedback control
DPF	• Filtering performance • Active regeneration fuel delivery • Feedback control
Exhaust gas sensors	• Air fuel ratio faults • Air fuel ratio performance • NO_x sensor performance • NO_x sensor faults

Table 2.1 Relevant OBD Checks

The OBD checks these parts regularly, and notes down when there is any malfunction. Every component has a different criteria for categorizing it as a malfunction. For majority of these components, a threshold value is set, based on the country's emission regulations and all values above (or below, depending upon component) are categorized as error [19].

A diagnosis engineer downloads the details of the component from the OBD in case of error. However, every aspect of the error is not available to the diagnosis team. They have to run multiple tests on the components to know where exactly the fault is. This is a highly time-consuming process [19].

2.2 Information to Driver

Currently there is a system in place, where the driver or the user of the vehicle is notified when the SCR is faulty [17]. This notification is mostly done on the instrument panel display of the car.

Figure 2.1 Instrument Panel Error Display

3 Theoretical Background

This chapter gives a brief overview of the various tools and systems used in the context of this thesis.

Section 3.1 explains about the SCR system and the possible errors of the SCR.

Section 3.2 explains about On-Board Diagnostics.

Section 3.3 gives a brief idea about python.

Section 3.4 gives an introduction to machine learning.

3.1 SCR and Possible Errors

Diesel engines, which run with an excess of air, do not create as much hydrocarbons and carbon monoxide as petrol engines. However, they produce harmful NO_x and PM [7].

The selective catalytic reduction system or SCR in short, is an emissions controlling technology used in the exhaust stream of diesel vehicles for the reduction of harmful NO_x gases. Currently, the SCR system is one of the most fuel-efficient and cost-effective technologies for the reduction of the harmful emissions [7].

Using SCR, a 90% reduction of NO_x, 30-50% reduction of particulate matter, and a 50-90% reduction of carbon monoxide & hydrocarbons is now possible. With the growing demand for diesel vehicles, the SCR system has gained vital importance. The SCR system is currently the best solution for the growing emission regulation standards [7].

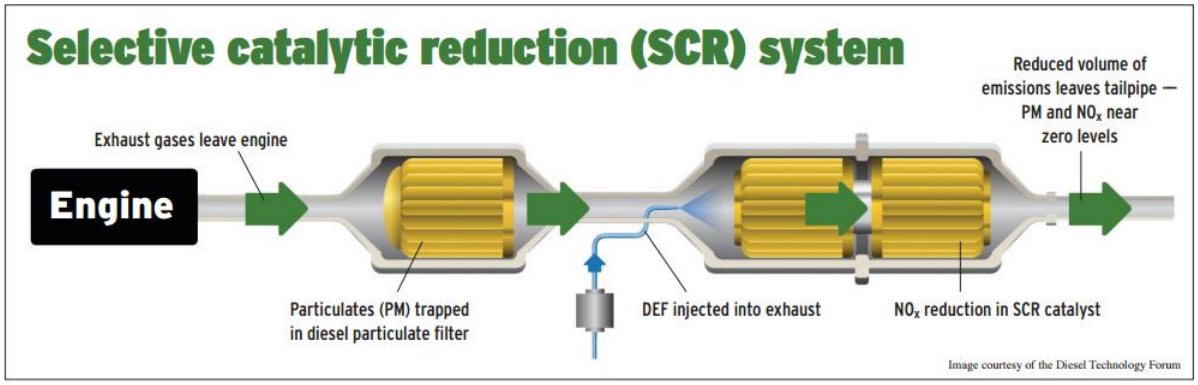


Figure 3.1 SCR System

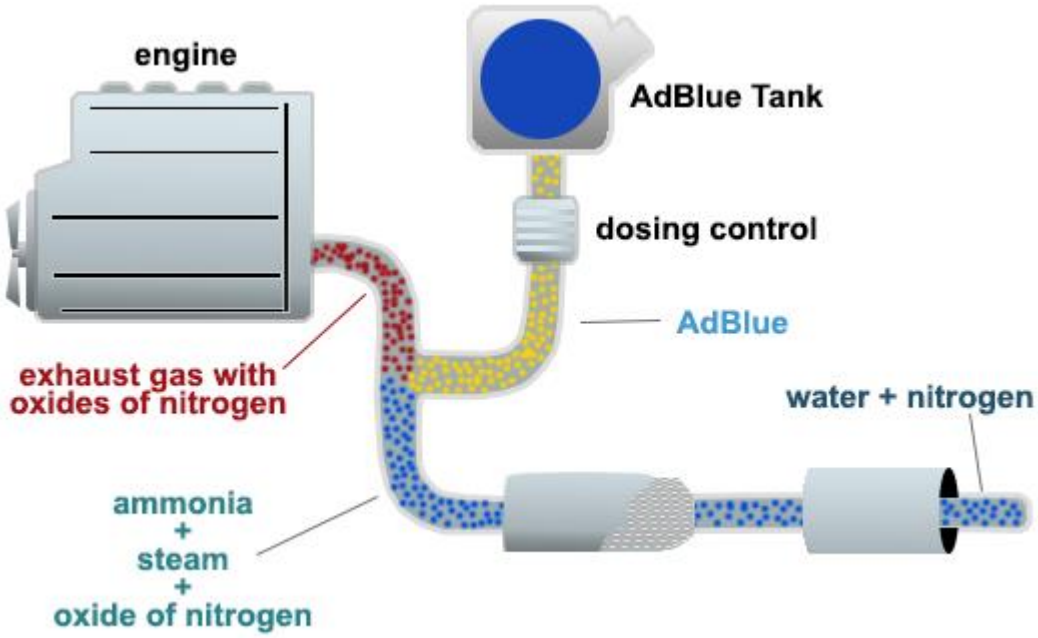


Figure 3.2 SCR System Process

The SCR operates using DEF, which is commonly known as Adblue. DEF is an aqueous solution of urea (NH3), which is stored separately in the vehicle, for use by the SCR system [8]. The chemical reactions occurring within the SCR are explained in the next section of this chapter.

Within the SCR, the injected DEF sets off a chemical reaction, which converts the harmful NO_x gases into nitrogen and water. After conversion, the produced components are expelled through the vehicle tail-pipe [7].

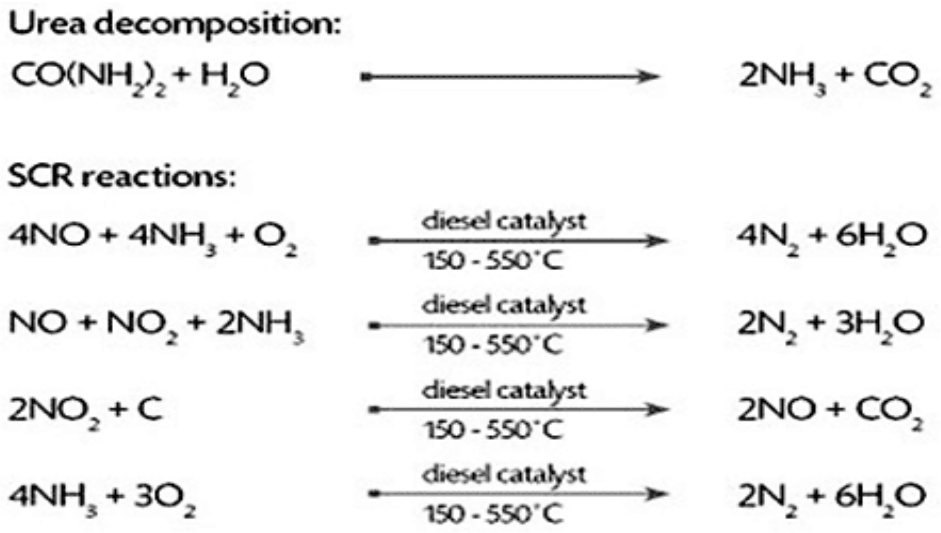


Figure 3.3 Reactions within SCR

Advantages of SCR [17]

- It reduces more NO_x than SNCR
- Easy installation

Disadvantages of SCR [17]

- To avoid SO_3 formation, the process is required to be performed at a high temperature
- Catalysts are expensive
- Catalysts can get negatively affected by the chemical reactions
- Expensive compared to SNCR

Chemical Reactions [20]

(1) $6NO + 4NH_3 \rightarrow 5N_2 + 6H_2O$

(2) $4NO + 4NH_3 + O_2 \rightarrow 4N_2 + 6H_2O$

(3) $6NO_2 + 8NH_3 \rightarrow 7N_2 + 12H_2O$

(4) $2NO_2 + 4NH_3 + O_2 \rightarrow 3N_2 + 6H_2O$

(5) $NO + NO_2 + 2NH_3 \rightarrow 2N_2 + 3H_2O$

(6) $8NO_2 + 6NH_3 \rightarrow 7N_2O + 9H_2O$

(7) $4NO_2 + 4NH_3 + O_2 \rightarrow 4N_2O + 6H_2O$

(8) $2NH_3 + 2O_2 \rightarrow N_2O + 3H_2O$

(9) $4NH_3 + 3O_2 \rightarrow 2N_2 + 6H_2O$

(10) $4NH_3 + 5O_2 \rightarrow 4NO + 6H_2O$

(11) $2NH_3 + 2NO_2 + H_2O \rightarrow NH_4NO_3 + NH_4NO_2$

(12) $NH_3 + SO_3 + H_2O \rightarrow NH_4HSO_4$

(13) $2NH_3 + SO_3 + H_2O \rightarrow (NH_4)_2SO_4$

- Equations (1), (2), (3), (4) and (5) show the reactions where NO_x is converted to elemental nitrogen.
- Equation (2) shows the dominant reaction.
- Equations (1) and (2) show the reaction of nitrogen oxide.
- Equations (3) and (4) show the reaction of nitrogen dioxide.
- Equation (5) is a very fast reaction and it requires low temperature of the SCR. This low temperature is ensured by increasing NO_2 levels within.
- When NO_2 concentration is higher than NO concentration, N_2O formation occurs. This is shown in Equation (6) and (7).
- An excess of oxygen in the SCR system may cause other reactions such as those shown in equations (8), (9) and (10).
- Equation (8) shows the production of nitrous oxide due to partial oxidation of ammonia.

- Equation (9) shows the production of elemental nitrogen due to partial oxidation of ammonia.
- Equation (10) shows the production of nitric oxide due to the complete oxidation of ammonia.
- Equation (11) shows the reaction of ammonia with nitrogen dioxide, which produces ammonium nitrate.
- Equation (11) occurs at low temperatures. The ammonium nitrate produced in equation (11) may lead to the incorrect functioning of the catalyst. To ensure that ammonium nitrate is not formed, the temperature can be set to a high value.
- Since ammonium nitrate is formed by the reaction of NO_2 with ammonia, as shown in equation (11), this formation can also be brought down by using precise amounts of ammonia.
- Due to the presence of sulphur in the system, sulphur dioxide is formed. Sulphur dioxide gets oxidized to form sulphur trioxide.
- Equations (12) and (13) show the reactions of sulphur trioxide with ammonia.
- The components formed as a result of equations (12) and (13) cause the catalyst to behave incorrectly. This can again be fixed by ensuring that the temperature is maintained at a high level.

The equations above show the importance of maintaining the temperature in the SCR. If the temperature lowers than a specific threshold, the SCR processes are affected [20].

The equations above also show how important it is to maintain the level of ammonia passed to the system. If the level increases beyond the necessary limit, secondary emissions are produced as shown in the equations above. If the level decreases beyond a certain limit, the required NO_x conversion does not take place [20].

Limitations [6]

- SCR systems are delicate and prone to contamination due to the various chemical reactions that occur within it [20].
- The catalysts in the SCR are made porous to increase the surface area for the reduction of NO_x. However, these pores are easily covered by fine particulates or other components, which are produced as part of the chemical reactions that occur within the SCR, such as ammonium sulphate and ammonium nitrate.
- The exact amount of ammonia that needs to be provided to the SCR system needs to be known in advance, in order to ensure that the reactions take place as expected.
- As seen in the chemical reactions section, temperature is a major limitation that needs to be considered.

Applications [15]

- Diesel cars
- Diesel trucks
- Industries which use diesel
- Marine vessels such as cargo vessels
- Waste incineration
- Energy plants
- Metal industry
- Greenhouse horticulture

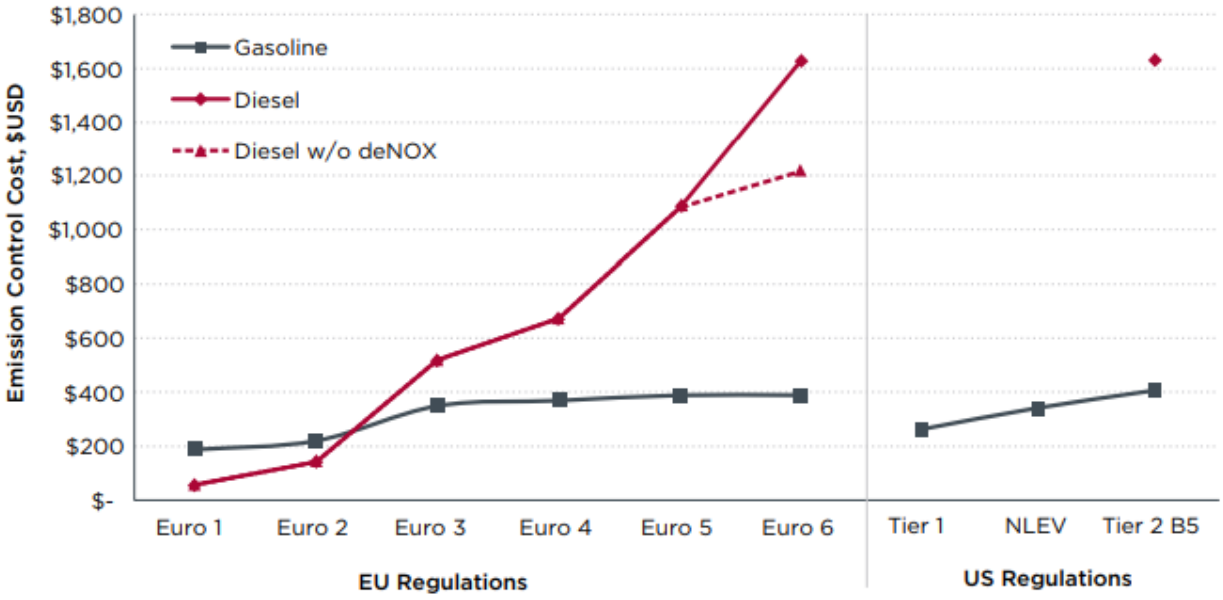


Figure 3.4 Comparison of Emissions Control Technology Costs

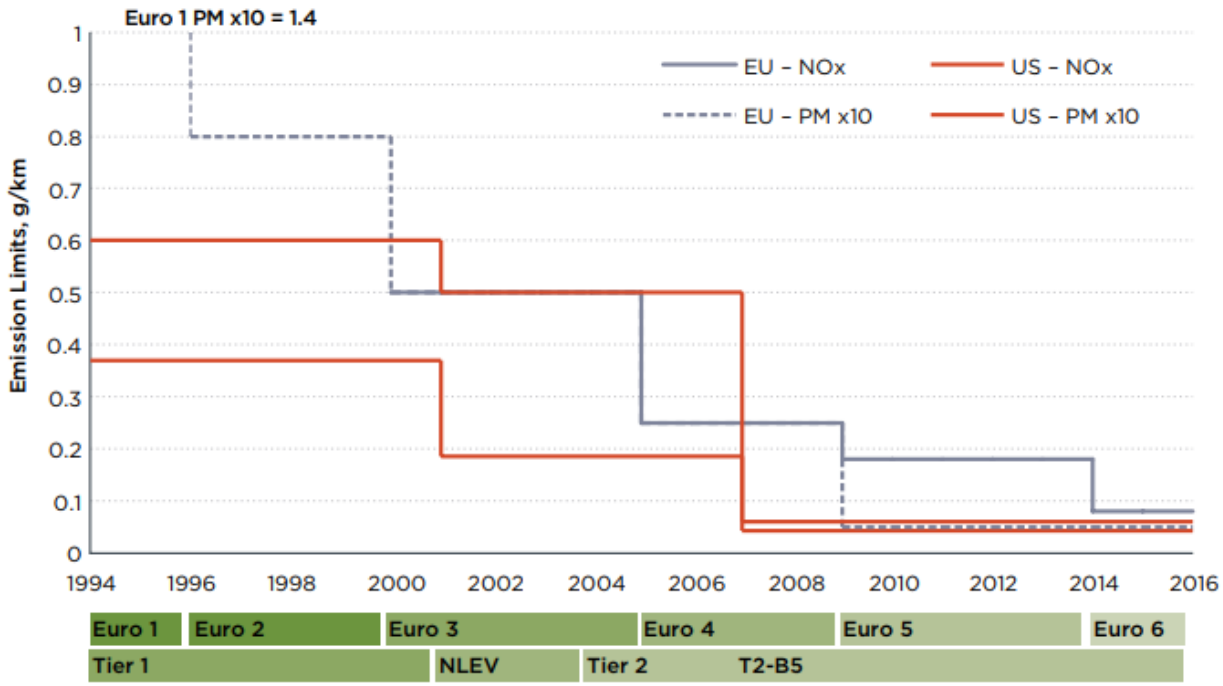


Figure 3.5 Emission Limits

DEF [15]

- Primary requirement for SCR
- Non-toxic aqueous urea
- Increasing world-wide distribution [19]

DIESEL	4-CYL OR Vd<2.5 LITERS	6-CYL OR Vd>2.5 LITERS
REGULATION	TIER 2 BIN 5	TIER 2 BIN 5
INTRODUCTION YEAR	2009	2009
REGULATED POLLUTANTS	(NO_x/PM/CO)	(NO_x/PM/CO)
EMISSION LEVELS, G/KM	0.04/0.006/2.5	0.04/0.006/2.5
1. A/F control & engine-out emission	**Assuming 4-cyl, Vd=2.0L**	**Assuming 6-cyl, Vd=3.0L**
Fuel system - 50% of cost (a)	$420	$459
Turbocharger - 50% of cost (b)	$138	$155
Intercooler - 50% of costs (b)	$32	$39
VGT (extra cost) - 50% of costs (b)	$50	$60
EGR valves (c)	$38	$38
EGR cooling system (c)	$58	$66
Engine mapping and tuning (d)	R&D	R&D
Improvements on combustion chamber & nozzle geometry (e)	R&D	R&D
Cost of A/F control & engine-out emission	**$736**	**$817**
2. Aftertreatment systems	**Vd=2.0 L**	**Vd=3.0 L**
Diesel oxidation catalyst (DOC) (f)	$78	$116
Diesel particulate filter (DPF) (f)	$332	$468
Lean NO_x trap (LNT) (f)	$413	($602)*
Selective catalytic reduction (SCR) (f, g)	-	$633**
Cost of aftertreatment systems (h)	**$823**	**$1,217**
3. Total cost of hardware [1+2]	$1,559	$2,035
4. Fixed costs (R&D, tooling, certification)	$51	$51
5. Total cost of emission control tech. [3+4]	$1,610	$2,086

Figure 3.6 Breakdown of Emissions Control Costs

TECHNOLOGY	NAS	ICCT	NAS	ICCT	EPA	ICCT	EPA	ICCT	COMMENTS
	4-CYL, 2.0 L		6-CYL, 3.5 L		4-CYL, 2.0 L		6-CYL, 3.0 L		
DOC	$226	$80	$262	$135	$216	$80	$277	$116	EPA SVR= 0.5 ICCT SVR= 0.75
DPF	$284	$333	$299	$536	$401	$333	$534	$468	EPA SVR= 1.0 ICCT SVR= 2.0
LNT	$647	414$	-		$392	414$	-	-	
SCR	-	-	$637	$559	-	-	$854	$524	EPA SVR= 1.0 ICCT SVR= 1.0
Total	**$1157**	**$827**	**$1198**	**$1230**	**$1009**	**$827**	**$1665**	**$1108**	

Figure 3.7 Costs of Various Available Technologies

Possible Errors

The SCR comprises of various inlets and sensors. Any error in any of these parts could lead to the faulty functioning of the entire system.

The possible errors of the SCR system are-

- Faulty Adblue level sensor
- Faulty Temperature sensors
- Faulty NO_x sensors
- Faulty enabling signals

The major consequences of errors on any part of the SCR are-

- Faulty conversion- When the optimum conditions of the SCR system are not maintained, the conversion may not occur as expected, leading to higher NO_x emissions, and sometimes, secondary emissions.

For example, as seen in the chemical reactions of this section, maintaining the temperature at a high level is required to ensure that the catalyst does not get affected by ammonium sulphate and ammonium nitrate. The importance of maintaining the precise amount of ammonia in the SCR was also explained in the chemical reactions section.

- Ammonia slip- When the optimum conditions of the SCR system is not maintained, unreacted ammonia escapes through the system [16].

  The ammonia slip increases at higher NH_3/NO_x ratios. When the NH_3/NO_x ratio goes higher than one, the ammonia slip increases. If the ratio goes lower than a certain threshold, the NO_x reduction capability of the SCR is lowered. Hence, an ideal ratio of 0.9 to 1 is used, thereby ensuring decent levels of NO_x reduction and ensuring that the ammonium slip is low [20].

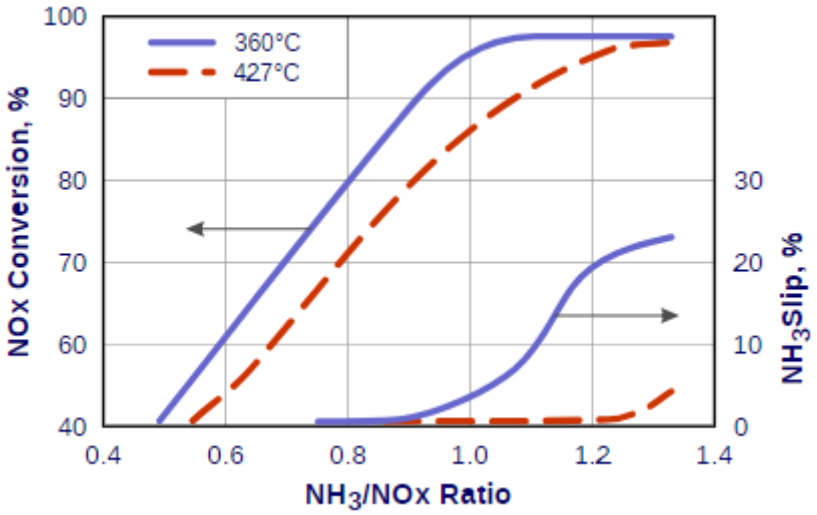


Figure 3.8 Proportionality of NO_x Conversion and NH_3/NO_x

3.2 On-Board Diagnostics

Emissions from automobiles have always been non-friendly to the environment, and to humans. Due to this, various emission checks have been implemented on automobiles nowadays. These checks are able to ensure that malfunctions of the various components of the vehicle do not go unnoticed [17].

These checks are done by the On-Board Diagnostic system on the vehicle. They are also responsible for [17]-

- Control of fuel injection
- Control of transmission

Features

- Monitors all emissions-related components of the vehicle.
- Tracks the various sensors to judge the performance of the emissions-related components of the vehicle.
- When any malfunction is detected, the driver is notified on the vehicle's instrument panel.

Figure 3.9 Vehicle Instrument Panel

Diagnosis using OBD

- Upon malfunction, the OBD records the component details [17].
- The diagnosis engineer downloads the details of the faulty component from the OBD [17].

3.3 Python

For the purpose of this thesis, the python programming language has been used. The reasons for high usability of python are [33]-

- Python is a high level programming language.
- A wide range of applications such as basic arithmetic computations to complex machine learning algorithms can be implemented using Python.
- Functional programming is supported.
- Object-oriented programming is supported.
- Python is open-source.
- Huge community is available worldwide for support.
- Multiple libraries already exist, and are still being created for the purpose of making computation with python easier and easier.
- Installing and using the libraries is made easy by Conda and Pip.

Python was designed by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands, in 1990 [33].

Python can be used on multiple IDEs. For the purpose of this thesis, we have used the Spyder IDE. Some of the features of the Spyder IDE are [34]-

- Supports advanced editing.
- Supports interactive testing and debugging.
- Has the MIT License
- Is part of the Anaconda python distribution.

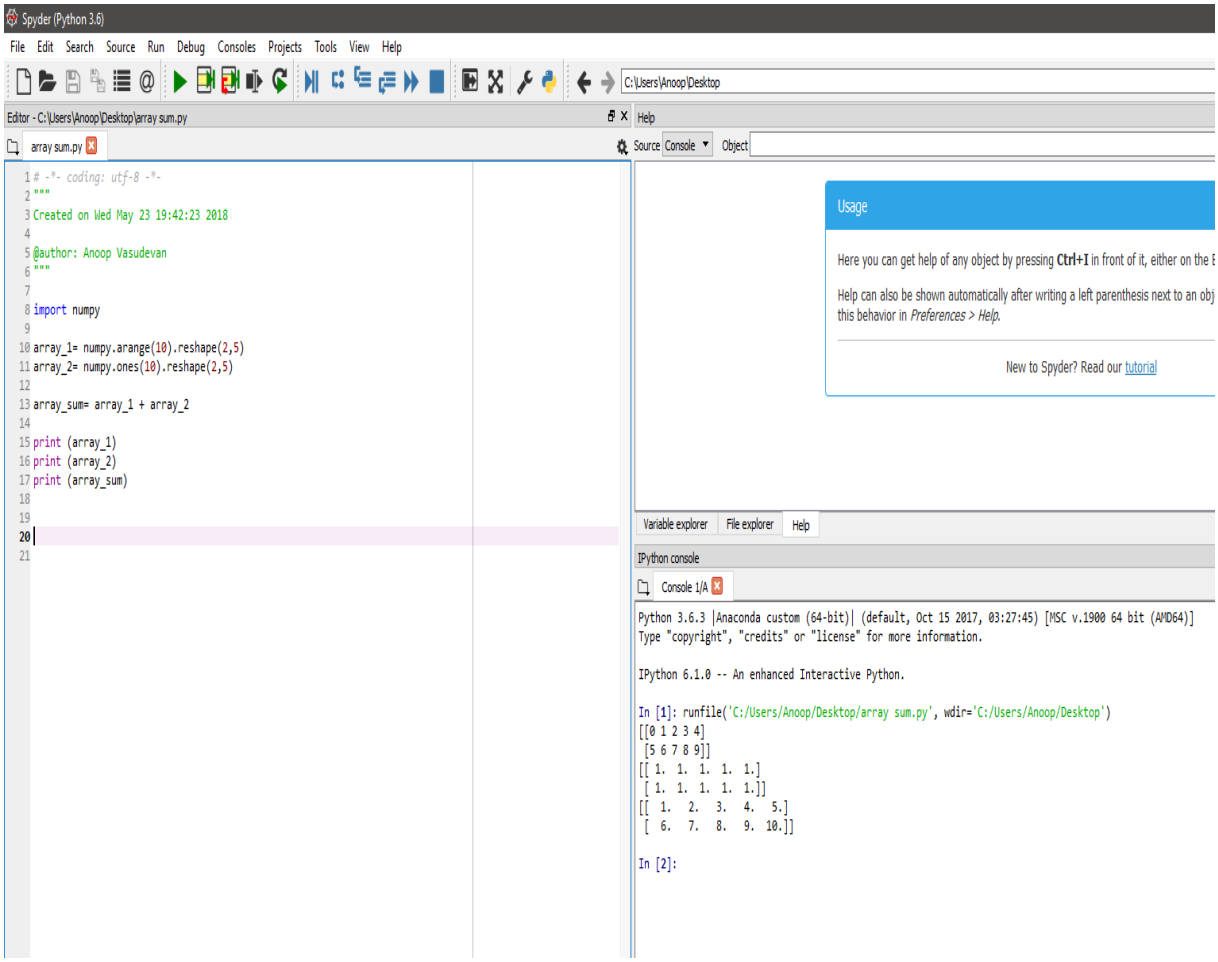


Figure 3.10 Spyder IDE

3.4 Machine Learning

The concepts of machine learning started prior to the 1950s. However, the realization of these concepts took many years in the making since machines powerful enough to process through the huge amounts of data did not come into existence until recently. The modern approach to machine learning could be said to have started 25 years ago, when a computer was taught to recognize shapes [24].

Machine learning provides a system the ability to learn and improve itself without the need for explicit programming. The focus of a machine learning program is its ability to learn from a huge amount of data [25].
The basic idea of machine learning is the development of an algorithm that reads from the input, processes the input and performs a statistical analysis to predict the output, which is pre-learnt to be within an acceptable range. This is similar to the process of data mining, where information is searched from a vast amount of data, and predictive modeling, where the program adjusts itself based on the data [26].

Difference between traditional programming and machine learning

- In traditional programming, a user creates a program, and provides data to this program. The program then produces the output.
- In machine learning, the input and output is provided to a system, and the system produces the program based on the input and output.

This difference can be seen below-

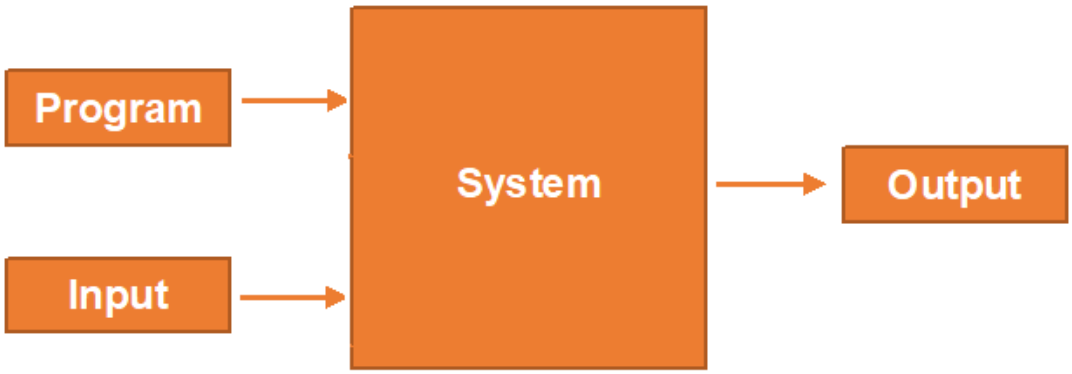


Figure 3.11 Traditional Programming

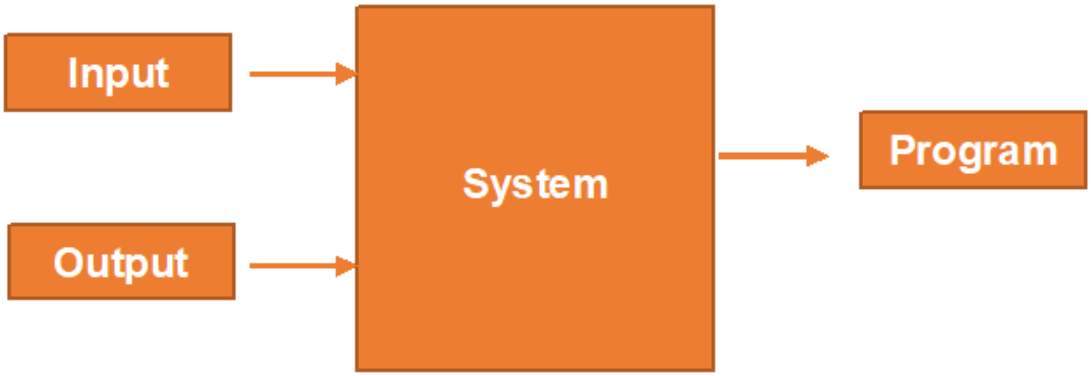


Figure 3.12 Machine Learning

Types [22]-

1. Supervised Learning

   In this type of learning, all the inputs and the corresponding outputs of these inputs are provided to the algorithm. The algorithm learns from these sets of inputs and outputs, and forms relevant connections. Once the algorithm learns the connections, it can be used for further data processing [25].

2. Unsupervised Learning

   In this type of learning, only the inputs are provided to the algorithm. The algorithm learns from these inputs, and finds patterns within these inputs. The algorithm differentiates between the inputs based on these patterns. Once the algorithm has learnt the patterns, it will be able to categorize any input that is given to it [25].

3. Reinforcement Learning

   In this type of learning, the algorithm is taught the best possible output for any input set. In reinforcement learning, all action that can be given as output is marked with a certain reward. At every point, the algorithm tries to find the action for which the reward is the maximum. Once this action is decided, the algorithm performs this action as the output for the given input [25].

Some of the applications of machine learning are given in Table 4.1 below-

Domain	Example
Robotics	Creating robots which are able to interact with humans and the environment
Finance	Prediction of stock variations
Web Search	Setting ranks for pages based on various factors
Space exploration	Performing space probe with minimal human intervention

Social networking	Finding relationship between users and communities
Debugging	Suggesting where the exact cause of the error lay

Table 3.1 Machine Learning Applications

As seen in Table 3.1, Debugging is one of the primary applications of machine learning [27]. In this report, we will focus on this domain of machine learning.

For this thesis, supervised machine learning is used for the debugging process. A layout of the process is given below-

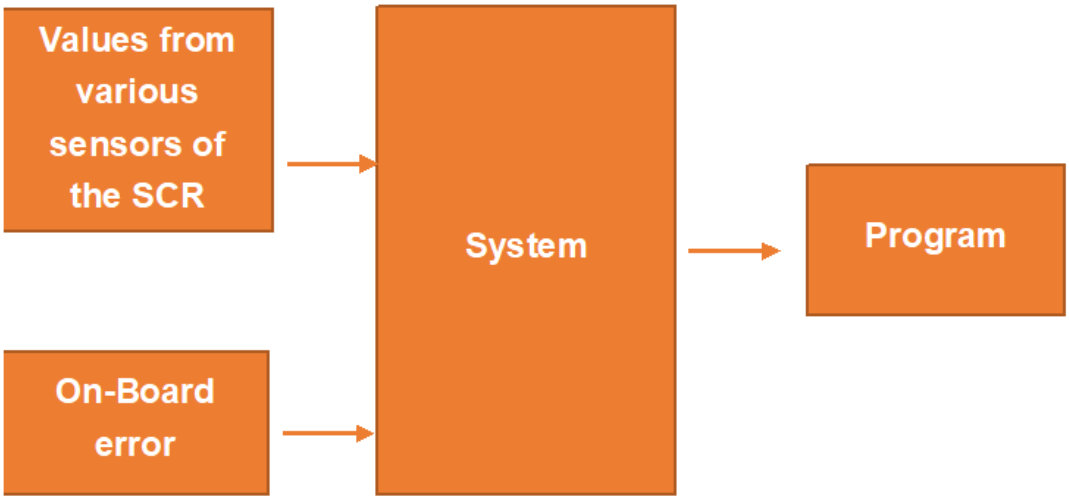


Figure 3.13 Layout of Algorithm

The inputs given to the algorithm here are the various sensor values, and the outputs are the various on-board errors that were obtained for these sensor values.

Since driving with a faulty SCR system is illegal, the errors for the purpose of this thesis have been simulated.

In supervised machine learning, the inputs and outputs are fed to the machine, and the machine finds a mapping function for these values, such as-

$$Y=f(x)$$

Where x is the input and Y is the output, and f(x) is the mapping function [27].

For the purpose of this thesis, x is the sensor values and Y is the OBD errors.

So our equation becomes,

$$\text{OBD errors} = f(\text{sensor values})$$

This mapping can be done using various methods of supervised machine learning such as-

- Nearest Neighbor

  The nearest neighbor algorithm classifies the input based on the values most similar to the given input and the category of these values [28].

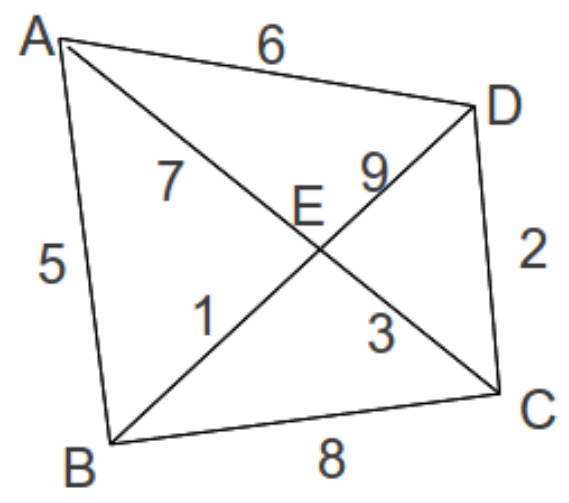


Figure 3.14 Nearest Neighbour Example

- Naive Bayes

  This algorithm classifies the data based on probabilities of the Bayes' theorem [28].

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Figure 3.15 Bayes' Theorem

- Decision Trees

  This algorithm classifies the inputs by continuously splitting the inputs based on certain parameters [28].

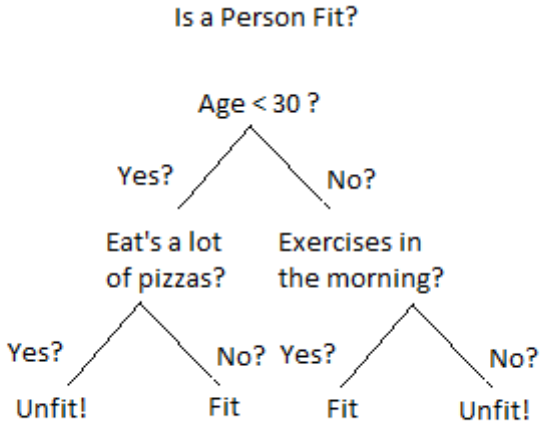


Figure 3.16 Decision Tree Example

- Linear Regression

  In linear regression, the system assumes linear relations between the inputs and the outputs. This linear relation helps the system in classifying the data accordingly [28].

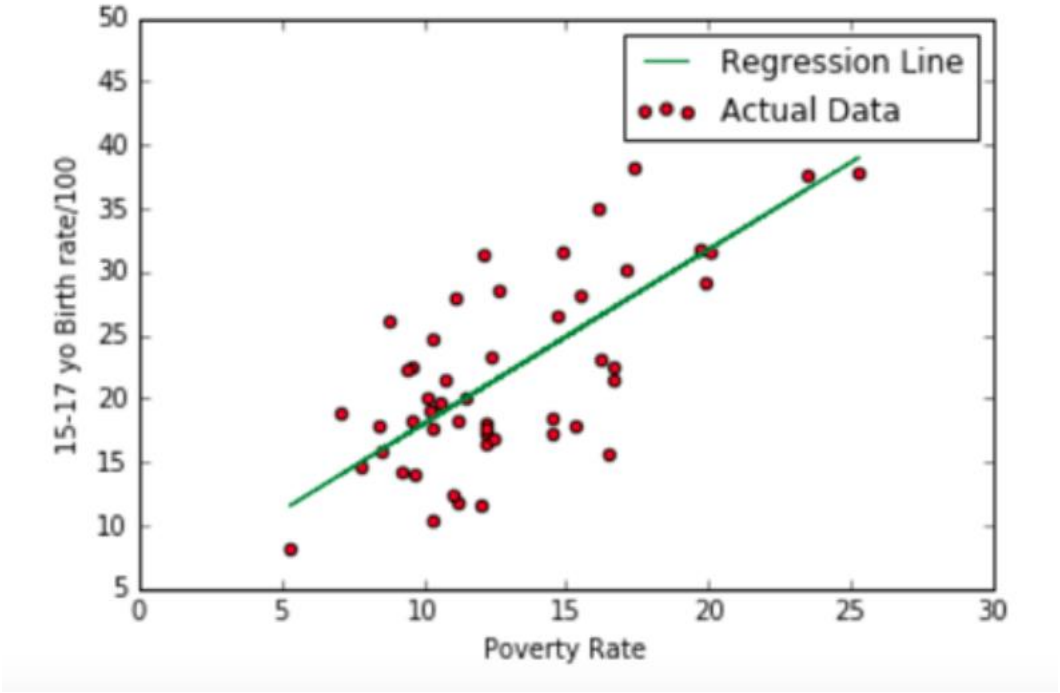


Figure 3.17 Linear Regression Example

- Support Vector Machines

  This algorithm transforms the data into higher dimensions using a technique called kernel trick. This transformation makes it easier to classify the data [28].

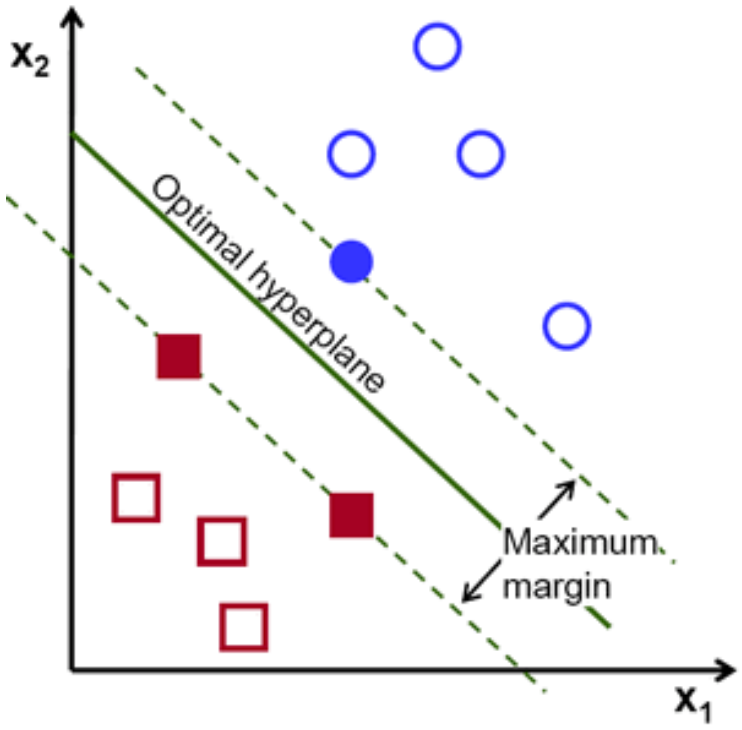


Figure 3.18 Support Vector Machine Example

- Neural Networks

  This algorithm breaks down the data into features, and allots the features to different processors (called nodes), which are placed in parallel for processing. These nodes learn the features and classify the data based on them [28].

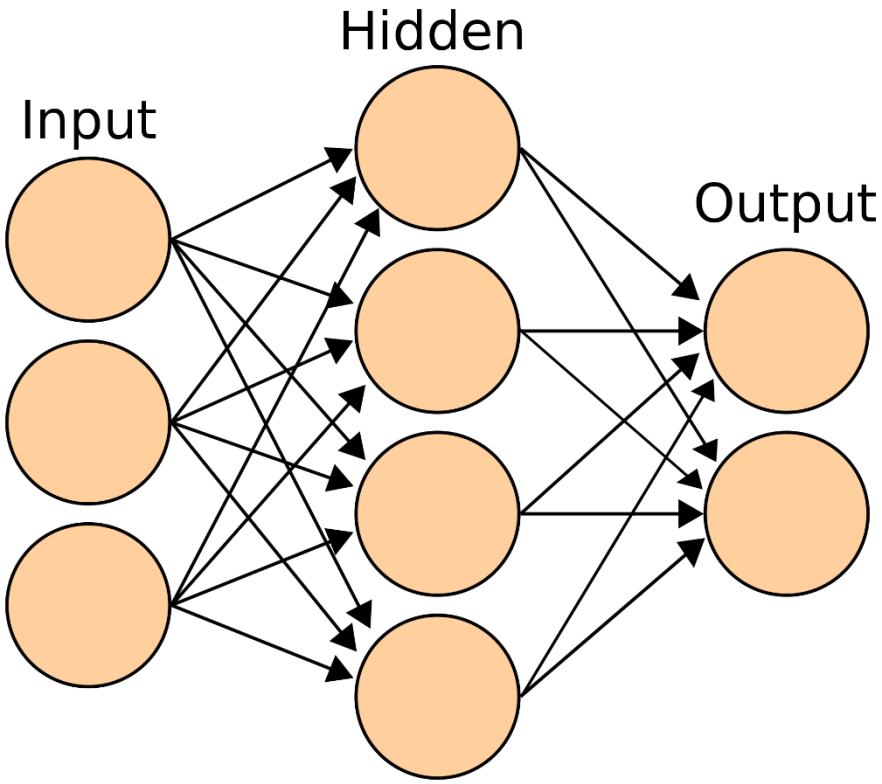


Figure 3.19 Neural Network Example

As seen from above, multiple algorithms can be used for the required purpose. However, we have used neural networks due to its similarity to the functioning of the human brain.

4 Concept

This chapter gives a brief overview of the approach that was implemented for the identification of the SCR errors.

Section 4.1 explains about the algorithm that will be used for the implementation.

Section 4.2 explains about gradient descent.

Section 4.3 explains about activation functions.

4.1 Algorithm

As explained in section 3.4, we have used the neural network algorithm for this thesis.

Neural network is a computational model used in machine learning. It was inspired on the neurological connections within the human brain [29]. The computations of the neural network are done by units called nodes. These nodes can also be considered as individual neurons [29]. There are three types of nodes. They are-

- Input Nodes

  The network gathers information from the user (or environment) through these nodes. The information is passed from the input nodes to the hidden nodes without any computation [29]. The input neurons collectively form the input layer of the network. Every neural network has one input layer.
- Output Nodes

  These nodes are responsible for passing on the computed information to the user (or environment) [29]. The output neurons collectively form the output layer of the network. Every neural network has one output layer.
- Hidden Nodes

  These nodes are responsible for the computations. They are called 'hidden' since they do not interact with the user or environment. They receive information from the input neurons and pass them to the output neurons after computation [29]. A hidden layer comprises of a set of hidden neurons. A neural network may have one or more hidden layers.

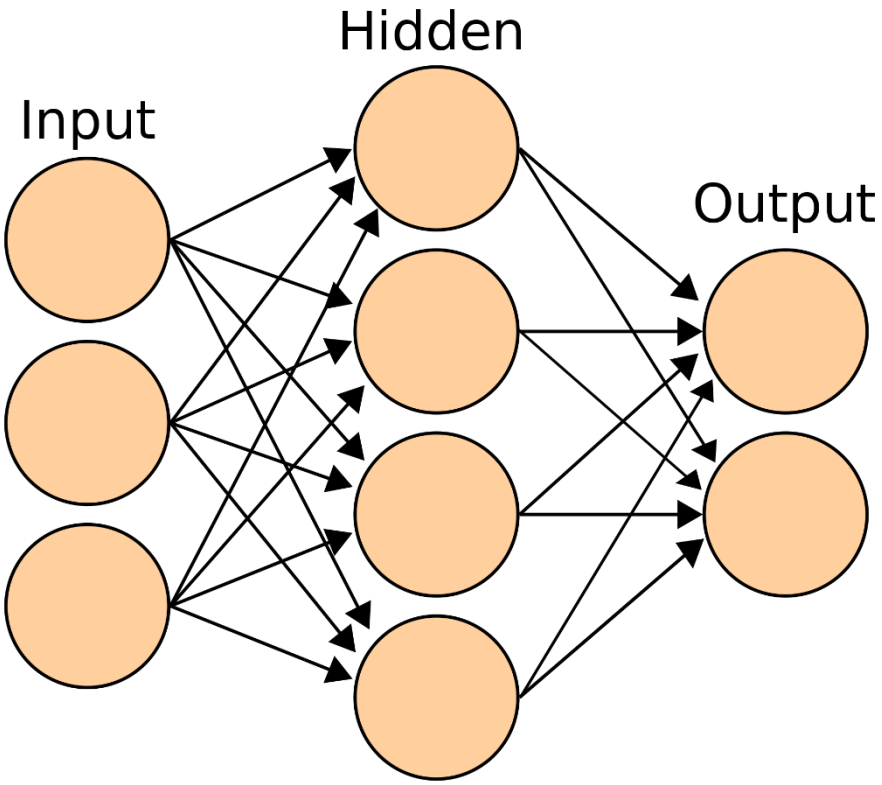


Figure 4.1 Neural Network with 1 Hidden Layer

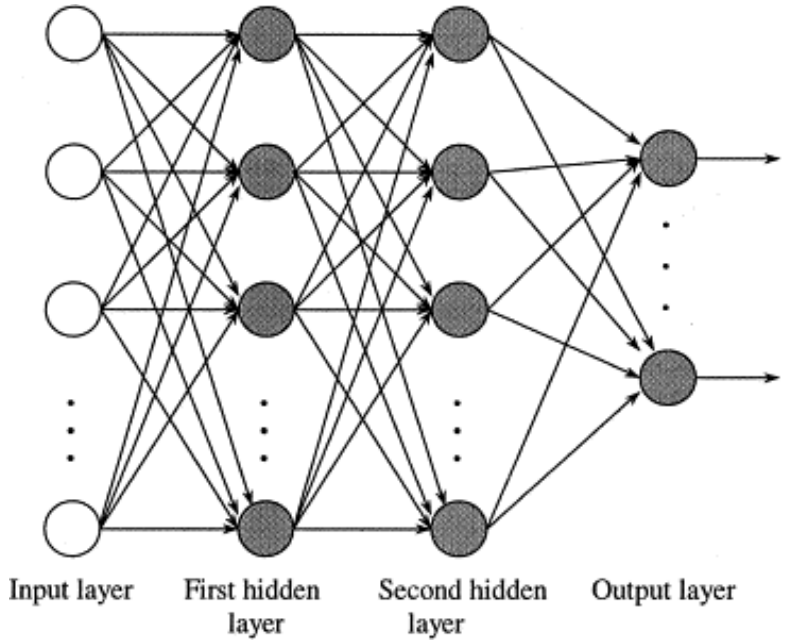


Figure 4.2 Neural Network with more than 1 Hidden Layer

Every neuron in each of the hidden layers creates an output based on the following equation [22]-

$$Y = \Sigma (\text{weight} * \text{input}) + \text{bias}$$

- *input* is the feature that is computed over, by the respective neuron
- *weight* and *bias* are the parameters learnt by the system
- *weight* is multiplied to the input
- *bias* is added to the result of multiplication of the input and weight

Each neuron performs computations on the features given to it, and passes the output to a neuron of the next layer. Eventually, all outputs converge at the output layer [22].

The steps involved in the working of a neural network are explained below [22]-

1. Weights and biases are randomly initialized.
2. Inputs are provided to the input layer of the neural network.
3. The input features are forwarded to the first hidden layer, which performs the computations on the inputs, and passes the outputs to the next hidden layer.
4. The features are forwarded from every hidden layer to the next hidden layer.
5. The last hidden layer forwards the outputs to the output layer.
6. A check is done on the output layer, where the difference between the generated output and the expected output is computed. This computed value is called error.
7. This error is passed back to the hidden layers, starting from the last hidden layer till the first hidden layer. The neurons in each of the hidden layers adjusts the weight and bias based on the back propagated error.
8. The inputs are again forwarded from the input layer to the hidden layers. The hidden layers now compute the inputs based on the new weight and bias values.
9. Steps 3-8 are repeated multiple times until the optimum weight and bias values are computed.

4.2 Gradient Descent

An important concept of neural networks is the optimization. Optimization is done to minimize the loss (or cost) function [25]. Multiple optimizers are available for this purpose, some of them being [29]-

- RMSprop
- Stochastic Gradient Descent
- Adam
- Adagrad
- Nesterov Accelerated Gradient

For the purpose of this thesis, we have used the stochastic gradient descent.

In order to understand Stochastic Gradient Descent, it is important to understand gradient descent.

Gradient Descent is a way to minimize the loss function. The gradient descent determines the learning rate at which the local minimum of the loss function is computed. The learning rate is the size of steps to be taken after each computation, in order to reach the minima [29]. There are 3 variants of Gradient Descent. They are-

- Batch Gradient Descent

  The gradient of the cost function is computed initially for the entire training dataset. This is a slow method, since the gradient for the entire training dataset needs to be performed for making just one update to the learning rate [29].
- Stochastic Gradient Descent

  The learning rate is updated after calculating the gradient at each step.

  Due to continuous updates, there are high fluctuations in the loss function [29].
- Mini-Batch Gradient Descent

  This is similar to batch gradient descent, however, this is much faster since the learning rate is adjusted after computing the gradient for small batches of the training dataset [29].

4.3 Activation Functions

In section 4.1, the algorithm used is explained in detail. The working of a neural network is also explained in nine steps. Among these steps, step 7 states that-

This error is passed back to the hidden layers, starting from the last hidden layer till the first hidden layer. The neurons in each of the hidden layers adjusts the weight and bias based on the back propagated error.

Though this step explains about the adjustment of weights and bias according to the back-propagated error, the degree to which each neuron must adjust itself is not clear. This degree is defined by the activation function [26].

Activation functions define whether a neuron is relevant or not. There are multiple functions which can be used for this purpose. They are [22]-

- Step Function

  This activation function is threshold based. If the output of a neuron is above a set value, the output is relevant. If not, it is not.

  A= 1, if Y > threshold

  A= 0, otherwise

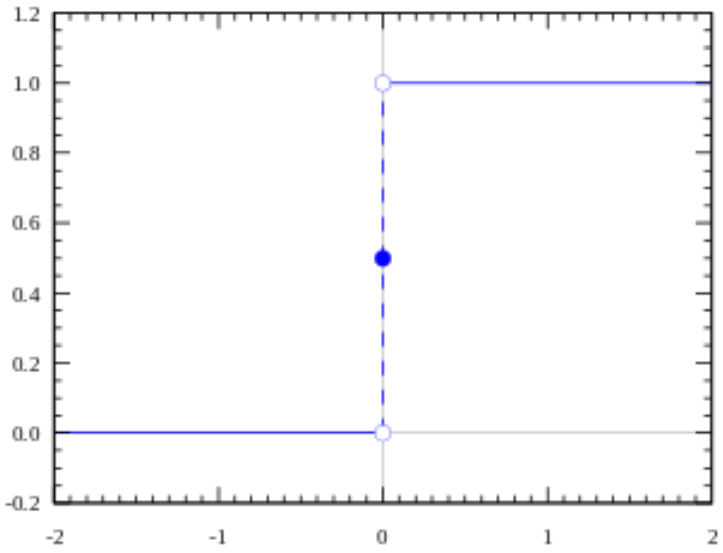


Figure 4.3 Step Function

- Linear Function

  In this activation function, the relevance is proportional to the input.

  $A = cx$

- Sigmoid Function

  This is one of the most popular activation functions due to its non-linear nature. The output of the sigmoid function always lies in the range (0, 1) as opposed to (-infinity, infinity).

  $A = 1/ (1+e^{-x})$

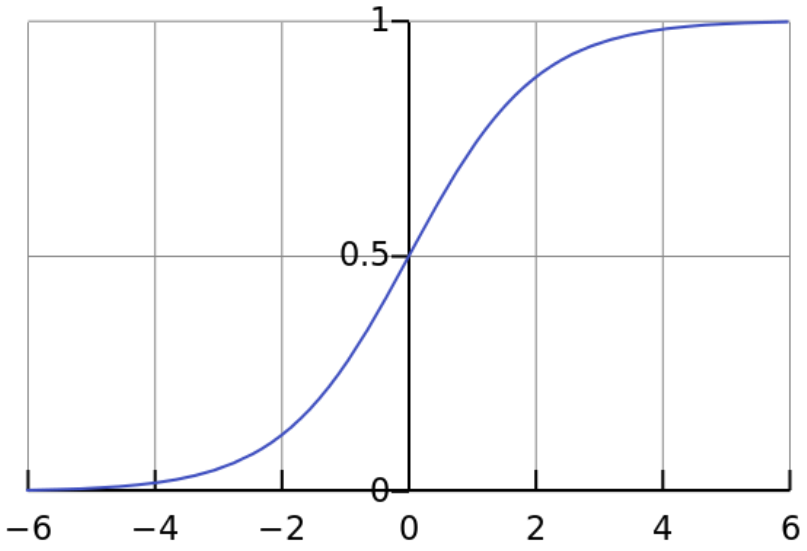


Figure 4.4 Sigmoid Function

- Tanh function

  The tanh function is similar to sigmoid, and the output is bound to (-1, 1).

  $Tanh(x) = 2/ (1+e^{-2x}) - 1$

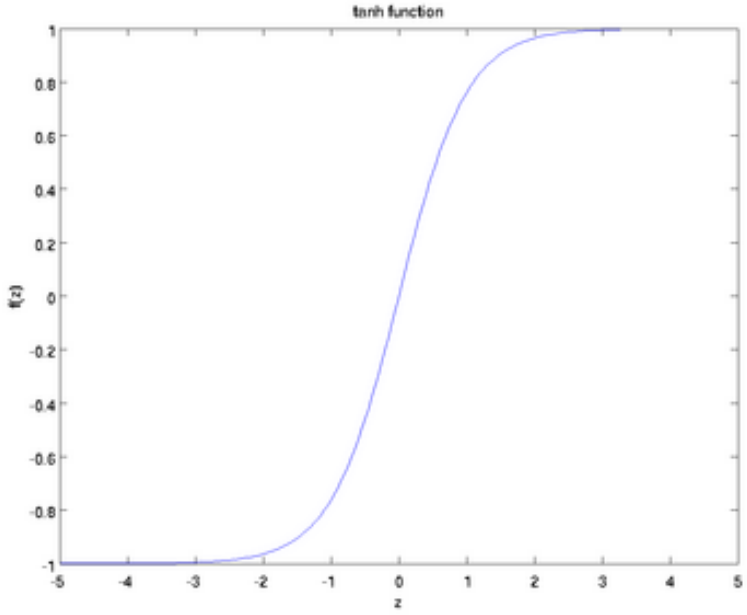


Figure 4.5 Tanh Function

- ReLu Function

  ReLu gives an output only if output is positive, however, ReLu is non-linear and the output lies within [0, INF).

  A(x)= max (0, x)

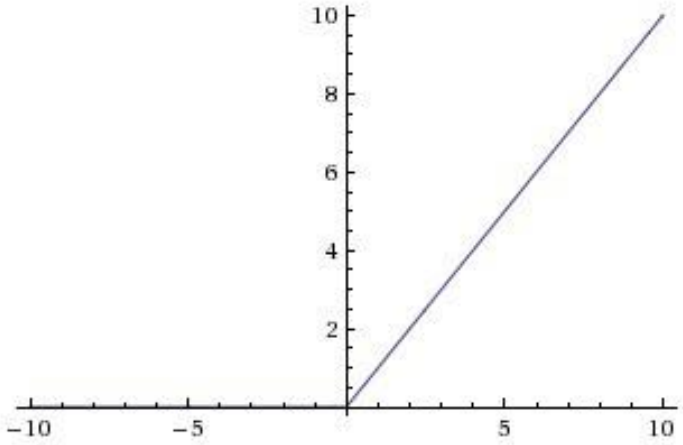


Figure 4.6 ReLu Function

- Softmax Function

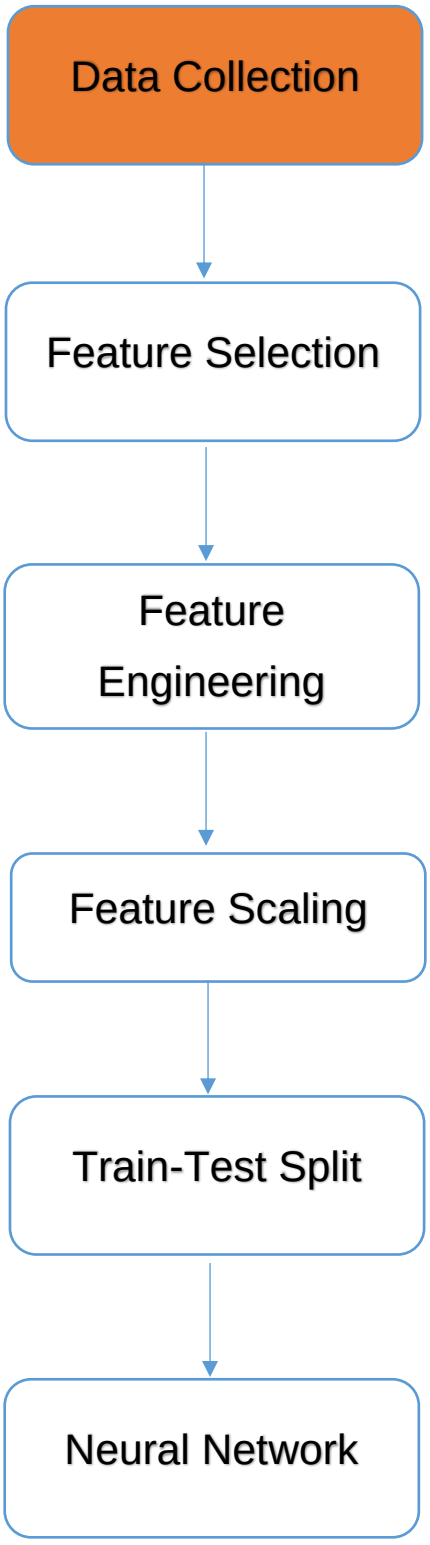
  This function calculates the probabilities of each output over all possible outputs.

  $A = e^x / (\Sigma e^x)$

  The softmax function is generally only applied to the final hidden layer.

5 Implementation

This chapter gives a brief overview of how the approach is implemented.



Section 5.1 explains about how the data was collected. This step is marked with a different color since this step was done prior to the implementation of this thesis.

Section 5.2 explains about how the features were selected.

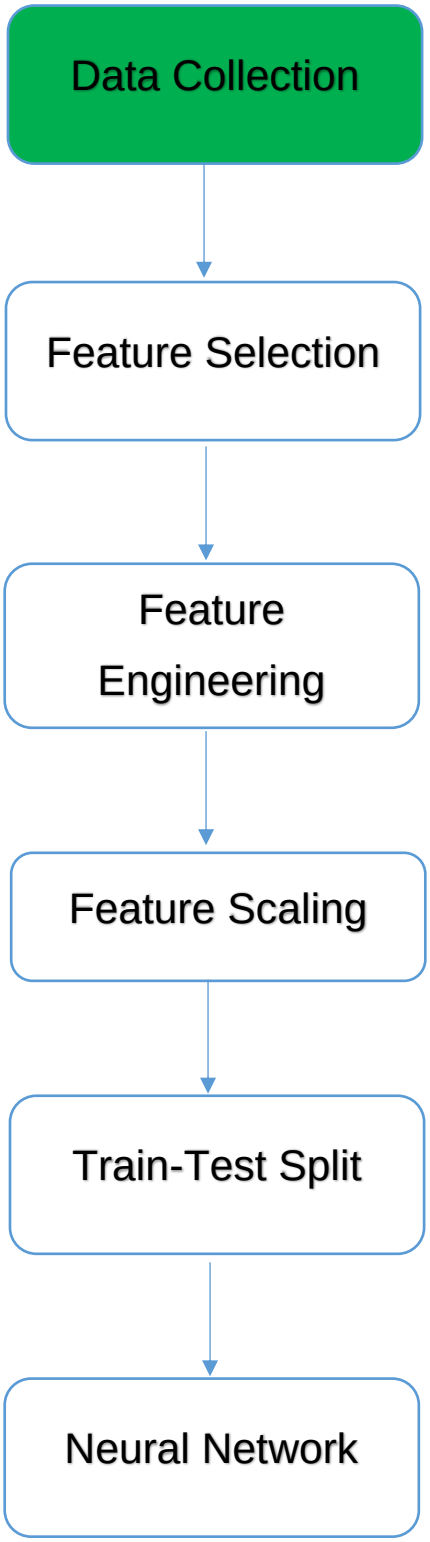
Section 5.3 explains about how certain features were manipulated for the learning process.

Section 5.4 explains about the process of feature scaling.

Section 5.5 shows how the dataset is split into training set and testing set.

Section 5.6 explains how the neural network is implemented.

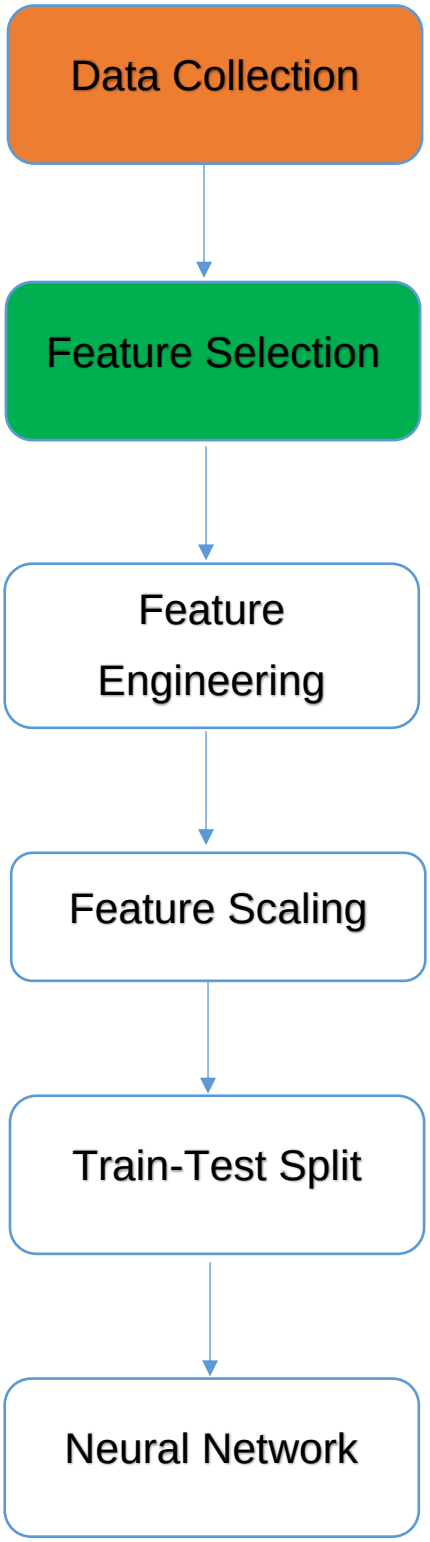
5.1 Data Collection



Data collection is the process of accumulating data for the purpose of teaching the algorithm. The data collection could be done in-house or the data can be purchased from data brokers [24]. For the purpose of this thesis, the data collection was done in-house at IAV GmbH.

- In this step, the values of the different sensors and the OBD errors were recorded.
- Since driving a car with a faulty SCR is illegal, majority of the data was simulated.
- After the recording and simulation, a dataset of 90GB was attained.

5.2 Feature Selection



Feature Selection is the process of deciding which features of the dataset are required for the purpose of teaching the algorithm as intended [24].

- A thorough check was done over the various inputs and outputs of the 90GB dataset that was collected as part of the previous step. This check determined which inputs and outputs were relevant for the SCR. The inputs are-
  - SCR_NOM_rNOxNSCDs
  - SCR_NOS_rNOxNoCat2Ds
  - ScrObdNoeCo_EffFil
  - ExhTp_TpScrUs
  - ExhTp_TpScrDs
  - ExhTp_TpScrAvg
  - SCRLdg_mNH3LdNom
  - SCRMod_mEstNH3Ld
  - EXH_MAF_Unfiltd
  - ExhTp_TpDocUs
  - ExhTp_TpDpfUs
  - MaCo_Dos_dmNH3MetAdapDes
  - SPL_AirFuelRatio2
- The outputs are-
  - PAR_NH3_Drift
  - PAR_NH3_Offset
  - PAR_NO2_NOx_Drift
  - PAR_NO2_NOx_Offset
  - PAR_NOxDs_Drift
  - PAR_NOxDs_Offset
  - PAR_NOxDs_Tlow
  - PAR_NOxUs_Drift
  - PAR_NOxUs_Offset

- PAR_NOxUs_Tlow
- PAR_T6_Drift
- PAR_T6_Offset
- PAR_T6_Tlow
- SCRGreg_E0_des

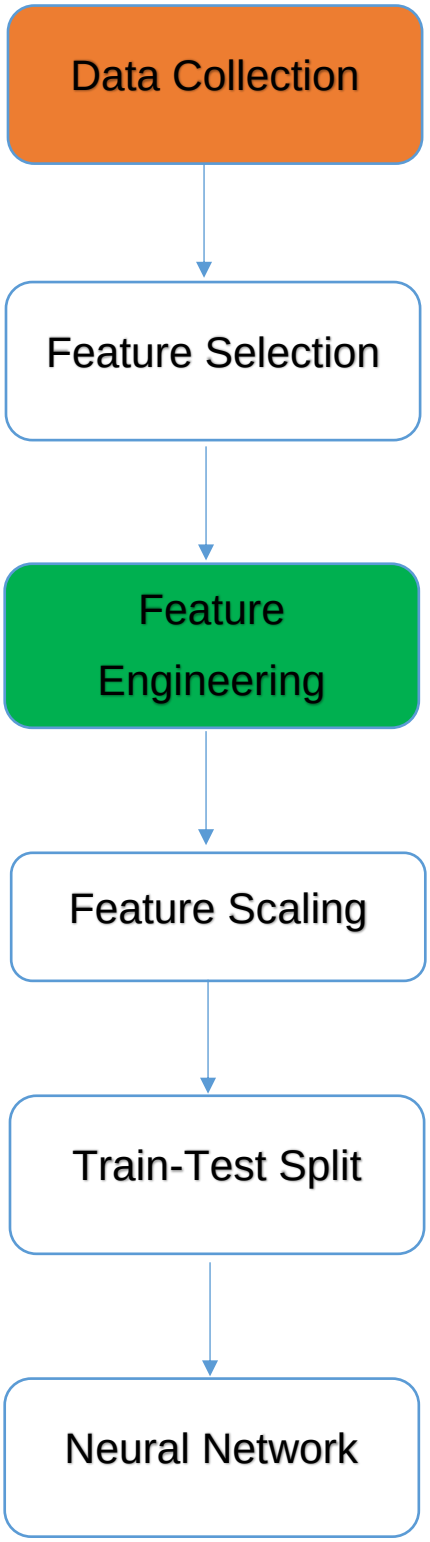
- The outputs were normalized based on their ideal values. The ideal values for each of the output signals are given below-
  - PAR_NH3_Drift = 1
  - PAR_NH3_Offset = 0
  - PAR_NO2_NOx_Drift = 1
  - PAR_NO2_NOx_Offset = 0
  - PAR_NOxDs_Drift = 1
  - PAR_NOxDs_Offset = 0
  - PAR_NOxDs_Tlow = 0
  - PAR_NOxUs_Drift = 1
  - PAR_NOxUs_Offset = 0
  - PAR_NOxUs_Tlow = 0
  - PAR_T6_Drift = 1
  - PAR_T6_Offset = 0
  - PAR_T6_Tlow = 0
  - SCRGreg_E0_des = 78000

```
import numpy

output_channels=['PAR_NH3_Drift','PAR_NH3_Offset','PAR_NO2_NOx_Drift',
                 'PAR_NO2_NOx_Offset','PAR_NOxDs_Drift',
                 'PAR_NOxDs_Offset','PAR_NOxDs_Tlow','PAR_NOxUs_Drift',
                 'PAR_NOxUs_Offset','PAR_NOxUs_Tlow',
                 'PAR_T6_Drift','PAR_T6_Offset','PAR_T6_Tlow','SCRGreg_E0_des']
#output channels required for classification

normalised_outputs=numpy.array([1,0,1,0,1,0,0,1,0,0,1,0,0,78000])
#values of output channels under optimum conditions

out_array=(out_array!=normalised_outputs).astype(int)
#out_array contains the values of the output channels, from the dataset

```

Figure 5.1 Python Snippet of Output Values Normalisation

- It was also observed that the above mentioned signals were behaving randomly at certain times. It was determined that the values were correct only when certain status signals were set. Hence, only those input signal values were selected, where the following status signals had a minimum value of 0.5 each-
  - SCR_NOS_stNOxNoCat2Ds
  - SCR_NOM_stNOxNSCDs
  - DStgy_stMetStgy

5.3 Feature Engineering



Feature Engineering is the process of transforming the feature space of a predictive model to improve its performance. In this step, new features are created from existing features using multiple methods such as arithmetic operations [23].

- It was observed that the input values that are required for the purpose of this thesis had a tendency for sudden spikes in values. These spikes could be positive or negative.
- Due to these spikes, it was evident that computing the algorithm for individual values of the inputs would result in an incorrectly-learnt algorithm. Hence, instead of individual values, a range of values was set as input.
- This range of values was attained by computing the average of the values over the past 'x' seconds. After multiple computations, the ideal 'x' value was found to be 30 seconds.
- It was also observed that the OBD took a certain amount of time (y) for logging the corresponding errors from the input signals. So, it was important to provide the inputs that were computed as the average of the past 'x' seconds, in intervals of 'y' seconds. After multiple computations, the ideal 'y' was found to be 180 seconds (3 minutes).
- The output to be given to the system was stored in an array. The element corresponding to the error would set to 1 when it occurred. However, there was no element corresponding to a situation where there was no error. Hence, a new element was added to the output array, which would set to 1 when all others were 0.

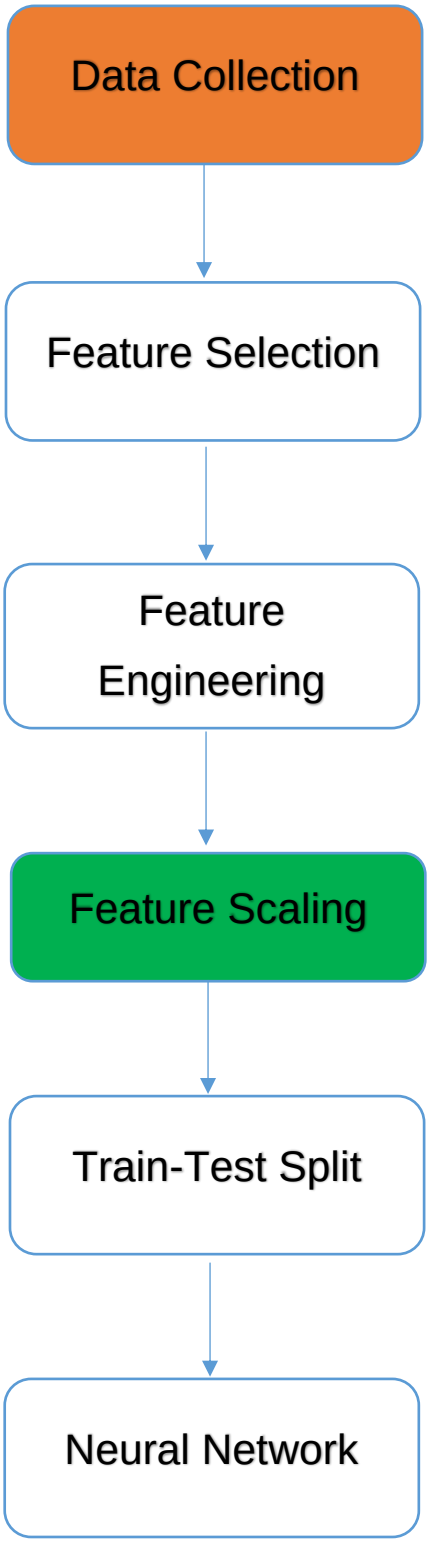
```
import numpy
import scipy

one_second=100 #one second in time steps
filt=numpy.ones(one_second) #array of ones over one second
time_chunks_length=1800 #history based on defined y

#convolution of inputs done over 1 second, over the columns
in_convoluted_array=numpy.apply_along_axis(lambda m: scipy.signal.fftconvolve(m, filt, 'valid'),
                                           axis=1, arr=in_array)
#in_array is the array of inputs from the dataset

#saving the convoluted results based on the required history and interval
in_array= numpy.zeros([in_convoluted_array.shape[1]-time_chunks_length,
                       in_convoluted_array.shape[0],6])

#creating new array with values based on required history
for i in range(in_array.shape[0]):
    index=numpy.asarray([j*3*one_second+i for j in range(1,7)])
    in_array[i,:,:]=in_convoluted_array[:,index]

```

Figure 5.2 Python Snippet of Historical Value Computation

```
#adding an extra column to the errors, which equates to 1 if no error exists
no_error_bit=0 if numpy.any(out_array) else 1
out_array=numpy.hstack([out_array,no_error_bit])
```

Figure 5.3 Python Snippet of 'no-error' Element Addition to Outputs

5.4 Feature Scaling

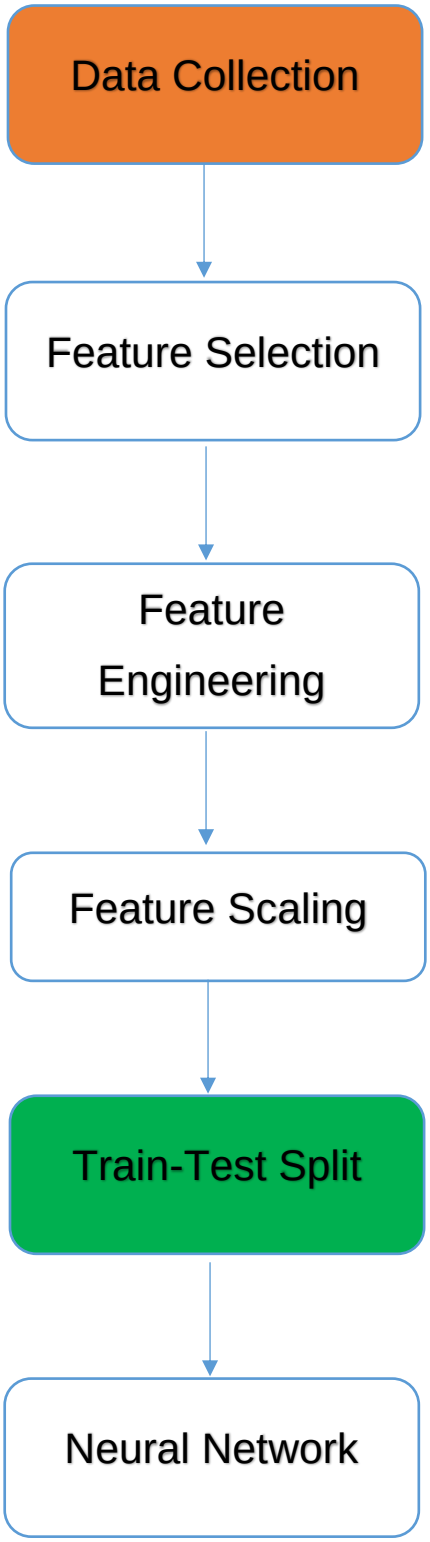


Feature Scaling is the process of ensuring that all the features of the data are within the same range [22].

```
12 from sklearn.preprocessing import StandardScaler
13
14 scaler=StandardScaler() # object for normalisation of inputs
15 input_normalised=scaler.fit_transform(input) #normalising the inputs
16
```

Figure 5.4 Python Snippet of Input Values Normalisation

5.5 Train-Test Split



This step is done to split the complete dataset into training and testing sets. This step is done for the following reason [22]-

- Once the training of the dataset is done, there should be a dataset to validate that the training process was successful.
- Using the entire or part of the same dataset that was used for training, will most often give a perfect result.
- However, this is not what is expected from a machine learning algorithm. The algorithm should be able to predict the output for future data.

The percentage of data that will be used for training and for testing needs to be determined beforehand. While determining this ratio, it should be ensured that overfitting and underfitting do not occur.

Overfitting

Overfitting is when too much data is provided to the algorithm. This results in the algorithm learning the provided data so well, that the algorithm's prediction accuracy would be almost 100% for the training data, and almost 0% for new data [22].

Underfitting

Underfitting is when too less data is provided to the algorithm. This results in the algorithm learning the model very poorly, thereby reducing the algorithm's prediction accuracy for the training data and the new data [22].

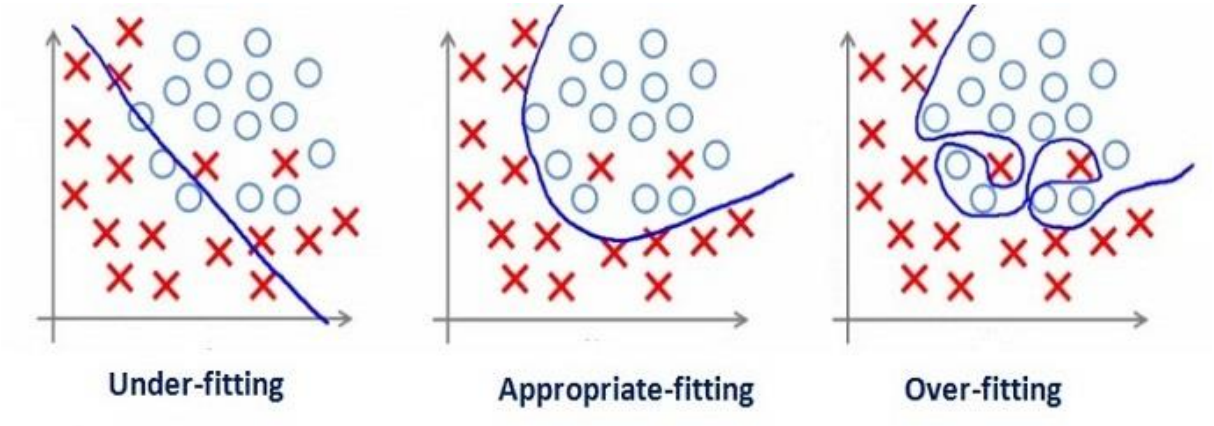


Figure 5.5 Overfitting and Underfitting

Generally, as a rule, the following is done-

- 75% of the data is set as training set
- 25% of the data is set as testing set

This is the general norm of splitting the data into training and testing set. However, as the need changes, most often, this limit is changed. Sometimes a ratio of 70-30 is followed. Sometimes, a ratio of 90-10 is also done [22].

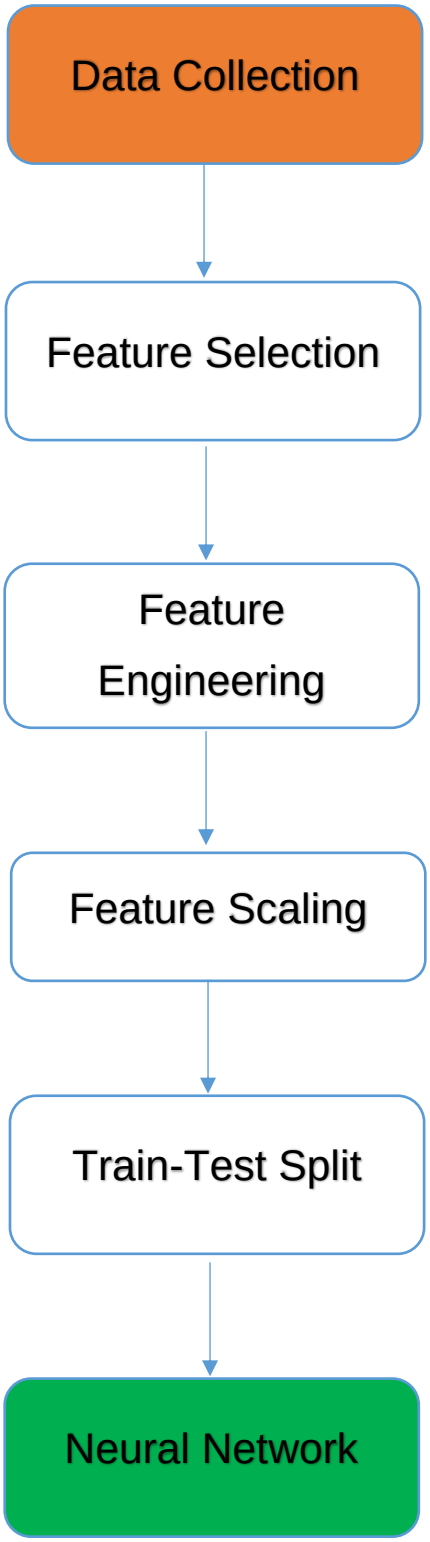
For the purpose of this thesis,

- We split the dataset into 70% training, and 30% testing.
- The 30% testing set is not used for training the algorithm.
- Upon completion of the learning process, the accuracy of the algorithm is checked for the training and the testing sets.

```
12 from sklearn.model_selection import train_test_split
13
14 input_train, input_test, output_train, output_test = train_test_split(input_normalised,
15                                                     output_normalised, test_size=0.3)
16
```

Figure 5.6 Python Snippet of Dataset Splitting

5.6 Neural Network



Prior to the learning process, the following steps were done to the dataset to ensure that the algorithm learns the model perfectly [22]-

- Selecting only the required features of the dataset.
  This step is explained in section 5.2.
- Building new features to ensure that the algorithm learns exactly what is intended.
  This step is explained in section 5.3.
- Data normalization to ensure that all the data is in the same range.
  This step is explained in section 5.4.
- Splitting the dataset into training and testing set to ensure that the validation of the algorithm can be done. This step is particularly important to ensure that overfitting and underfitting do not happen.
  This step is explained in section 5.5.

Once the above steps were done, the data was ready for the neural network.

Multiple methods are possible to build a neural network. However, for the purpose of this thesis, we have used the *keras* library.

Keras is an open source library, written by François Chollet [30]. The library can run over the following backend libraries-

- Theano
  Theano is an open source library that can be used for defining, optimizing and evaluating multi-dimensional array expressions very efficiently. Theano was developed by a group of machine learning enthusiasts at Université de Montréal [31].
- Tensorflow
  Tensorflow is also an open source library that allows users to perform numerical computations using data flow graphs. Tensorflow was developed by Google [32].

For the purposed of this thesis, keras is used over the theano backend.

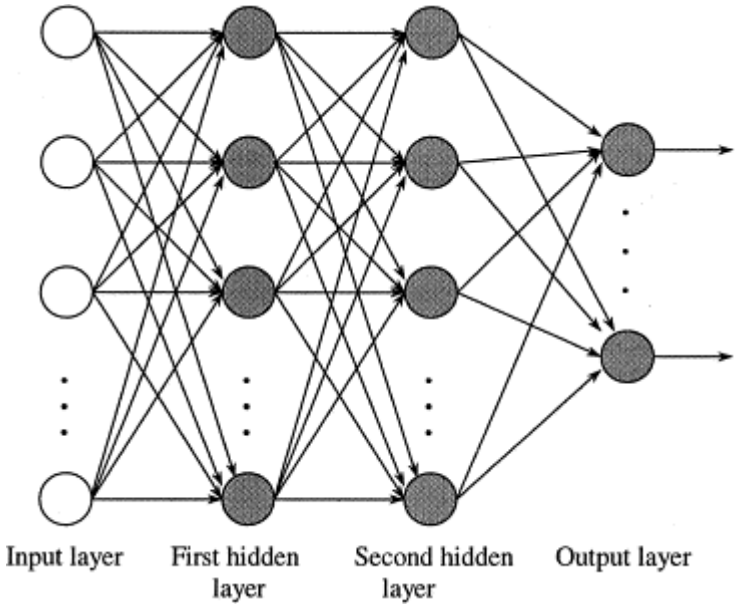


Figure 5.7 Neural Network Layout

As seen in the figure above, a neural network, like the human brain, has a complex structure-

- There is always one input layer. However, the number of neurons in the input layer needs to be decided.
- There is always one output layer. However, the number of neurons in the output layer needs to be decided.
- There could be one or more hidden layers. The number of hidden layers, and the number of neurons in each hidden layer needs to be decided.

As a general rule, the number of neurons in the input layer is set equal to the number of inputs. The number of neurons in the output layer is also set equal to the number of outputs.

Inputs

- The input channels were determined and extracted from the dataset (13 input channels).
- The inputs which had spikes in their readings were omitted.
- The specific input values were neglected to ensure that the algorithm did not learn from the unusual values.
- A 180 seconds range was decided for taking the inputs (determined after multiple trial and error).
- Within this selected range, for each point, the average over the past 30 seconds was computed (determined after multiple trial and error).
- This led to the number of inputs being computed as 78-

  $$N = 13 * (180/30) = 78$$

- Hence, the number of neurons in the input layer was set as 78.

Outputs

- The output channels were determined and extracted from the dataset (14 output channels).
- These output channels were normalized based on their ideal values.
- An extra channel was added to the outputs in the dataset, which would set to 1 when all other channels were set to 0 (not faulty).
- This led to the number of outputs being computed as 15.
- Hence, the number of neurons in the output layer was set as 15.

Hidden Layers

Determining the number of hidden layers and the number of neurons took a large number of trial and error iterations.

- Initially, the number of hidden layers was set as 1. The number of neurons in this layer was set as the median of the number of input neurons and the number of output neurons.

  Number of hidden layer neurons = (78 + 15)/2 = 46.5 ~ 46
- Slowly the number of hidden layers were increased, in such a way that the number of neurons in the hidden layers would be determined by splitting the difference of the number of the input and output neurons.

  For example, if the number of hidden layers were 3,
  - the first hidden layer would have (78 + 15)/4 * 3 = 69.75 ~ 69 neurons,
  - the second hidden layer would have (78 + 15)/4 * 2 = 46.5 ~ 46 neurons, and
  - the third hidden layer would have (78 + 15)/4 * 1 = 23.25 ~ 23 neurons.
- Until this point, the number of neurons was decreasing from the input layer to the output layer, through the hidden layers. This was changed, such that the first hidden layer would have more neurons than the input layer, and the number of neurons would decrease from this layer towards the output layer.

  For example, if the number of hidden layers were 3,
  - the first hidden layer would have 78 * 2 = 156 neurons,
  - the second hidden layer would have (156+15)/3 * 2 = 114 neurons, and
  - the third hidden layer would have (156 + 15)/3 * 1 = 57 neurons
- These steps were repeated for multiple combinations, to get the best accuracy.

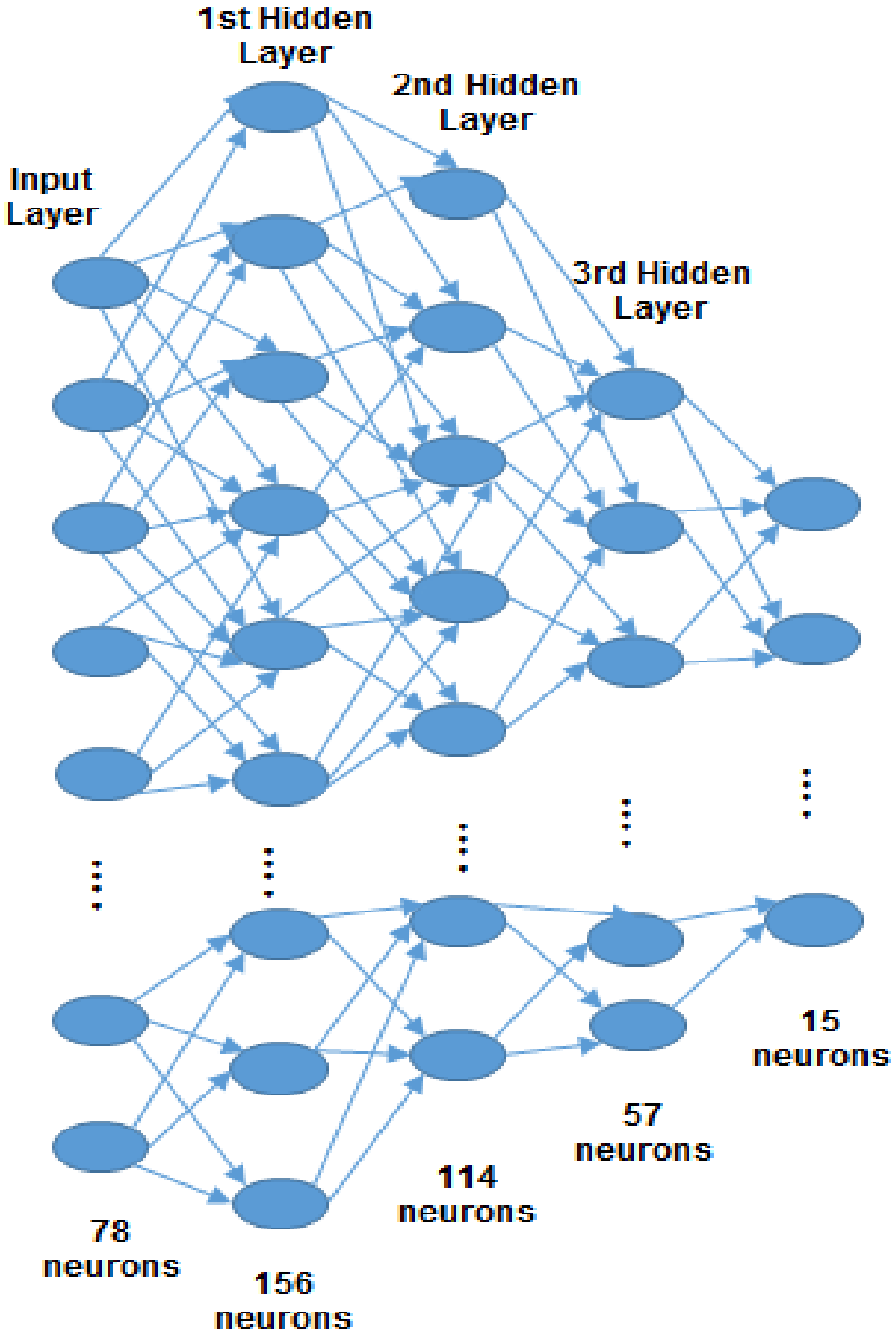


Figure 5.8 One of the Implemented Neural Network Models

Activation Function

As mentioned in section 4.3, choosing the activation functions was also of vital importance.

- Since only one of the outputs was supposed to be set at any point, the softmax activation function was used for the final hidden layer.

  This resulted in the outputs being set according to probabilities within the range (0, 1). For the purpose of verification, the output with the highest value was determined to be the set element.

  Softmax-

  $$A = e^x / (\Sigma e^x)$$

- The sigmoid and ReLu activation functions were tried for the input layer and the hidden layers.

  Sigmoid-

  $$A = 1 / (1+e^{-x})$$

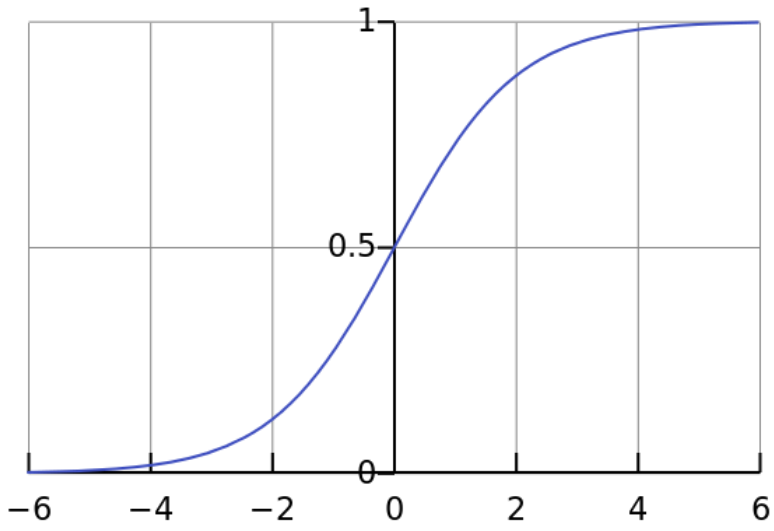


Figure 5.9 Sigmoid Function

ReLu-

$$A(x) = \max(0, x)$$

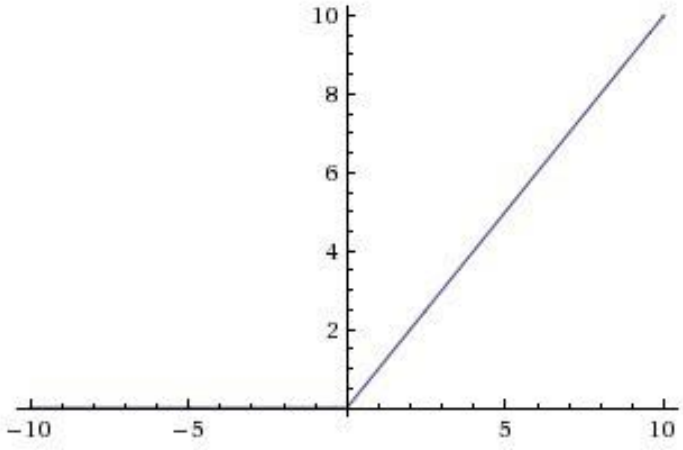


Figure 5.10 ReLu Function

- Upon multiple computations, it was determined that the ReLu activation provided the best accuracy.

Loss

Loss, or cross-entropy loss is a measure of the performance of a prediction model. Cross-entropy loss lies in the range (0, 1). Loss increases as the predicted result diverges from the expected result. A perfectly learnt algorithm would have a loss of 0 [22].

Multiple cross-entropy losses are possible in *keras* such as-

- Binary Cross Entropy
  Used for multi-label outputs [22].
- Categorical Cross Entropy
  Used for multi-class outputs [22].

Both these cross entropy losses were tried in this thesis, and it was determined that binary cross entropy loss was a better judge of performance for this model.

Batch Size

Batch size is the number of training examples in each iteration of the neural network [22].

- If the batch size is too high, the algorithm doesn't get to learn the all the features in detail, thereby causing underfitting.
- If the batch size is too low, the algorithm learns too much, thereby causing overfitting.

Multiple batch sizes were tried for this thesis, and a batch size of 1000 was seen to provide the highest accuracy.

Epoch

Epoch is the number of times that the neural network will perform the back-propagation of errors [22].

- If the number of epochs is too high, the algorithm takes too long to learn the features.
- If the number of epochs is too less, the algorithm doesn't learn the model perfectly due to lack of enough error adjustments.

Multiple epoch values were tried for this thesis, and 100 epochs was seen to provide the highest accuracy without taking unnecessarily long.

Note- A method to ensure that the algorithm doesn't take too much time, after it has learnt as much as possible, is by using an 'Early Stop'. This method checks the epochs, and if it sees that the change in loss in the past few epochs is lesser than a certain threshold, stops the learning process.

Learning Rate

Learning rate is defined as the rate of optimization of the gradient. It is in the range (0, 1) [22].

- A low learning rate results in a high computation time [22].
- A high learning rate results in a not-so-well-learnt model [22].

After multiple trial and error, a learning rate of 0.5 was determined to provide the best accuracy of prediction.

```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers,callbacks

early_stop=callbacks.EarlyStopping(monitor='loss',min_delta=0.01,patience=3,mode='auto')
# if the change in loss in the previous 3 epochs is below 0.01, the learning is terminated

classifier=Sequential() #initalise the classifier

classifier.add(Dense(units=156,kernel_initializer='uniform',activation='relu',input_dim=78))
#input layer of 78 neurons and 1st hidden layer of 156 neurons
classifier.add(Dense(units=114,kernel_initializer='uniform',activation='relu'))
#2nd hidden layer of 114 neurons
classifier.add(Dense(units=57,kernel_initializer='uniform',activation='relu'))
#3rd hidden layer of 57 neurons
classifier.add(Dense(units=15,kernel_initializer='uniform',activation='softmax'))
#output layer of 15 neurons, with softmax classifier

#setting the optimizer as stochastic gradient descent
sgd=optimizers.SGD(lr=0.5, momentum=0.0, decay=0.0, nesterov=False)

#compiling the classifier with the given loss and defined optimizer
classifier.compile(optimizer=sgd,loss='binary_crossentropy')

#learning the algorithm
classifier.fit(input_train,output_train,batch_size=1000,epochs=100,callbacks=[early_stop])
```

Figure 5.11 Python Snipper of Neural Network using Keras

Note-

- The computations for the purpose of this thesis take a huge amount of time, when processed on a machine without a gpu.
- The same computations can be speeded up by up to 400% when run on a machine with a gpu, by setting the below theano flag-

  THEANO_FLAGS= 'device=cuda, floatX=float32' [31]

6 Evaluation and Results

This chapter gives a brief overview of the results of the implementation.

Section 6.1 explains how the accuracy was calculated.

Section 6.2 explains the calculated accuracy of the algorithm.

Section 6.3 explains how the algorithm is better than the current scenario.

6.1 Accuracy of Algorithm

As mentioned in Chapter 5, the dataset was divided into training set and testing set. 70% of the dataset was set as the training set, upon which the algorithm learnt the classification.

When the learnt algorithm was implemented on the testing set, which comprises of 30% of the dataset, the accuracy was computed based on the following calculations-

- The classifier was run over the testing set of the input data.
- The results of the classifier's prediction was saved.
- The expected outputs from the testing set was compared to the classifier's results.
- This comparison helped in realizing the accuracy of the algorithm over the testing set.

```
48 import numpy
49
50 #computing the accuracy of the algorithm on the testing set
51 test_pred=classifier.predict(input_test)
52 test_pred_index=numpy.argmax(test_pred,axis=1)
53 test_output_index=numpy.argmax(output_test,axis=1)
54 test_check=test_pred_index==test_output_index
55 test_accuracy=(numpy.count_nonzero(test_check)/len(test_check))*100
56
```

Figure 6.1 Python Snippet of Accuracy Calculation over Testing Set

The algorithm's predicting capability over the training set was also calculated, to ensure that-

- The algorithm had learnt the classification well enough, or in other words, to ensure that there was no underfitting.
- The algorithm hadn't learnt the classification too well, or in other words, to ensure that there was no overfitting.

The accuracy of the algorithm on the training set was computed based on the following calculations-

- The classifier was run over the training set of the input data.
- The results of the classifier's prediction was saved.
- The expected outputs from the training set was compared to the classifier's results.
- This comparison helped in realizing the accuracy of the algorithm over the training set.

```
38 import numpy
39
40 #computing the accuracy of the algorithm on the training set
41 train_pred=classifier.predict(input_train)
42 train_pred_index=numpy.argmax(train_pred,axis=1)
43 train_output_index=numpy.argmax(output_train,axis=1)
44 train_check=train_pred_index==train_output_index
45 train_accuracy=(numpy.count_nonzero(train_check)/len(train_check))*100
46
```

Figure 6.2 Python Snippet of Accuracy Calculation over Training Set

6.2 Result

- The accuracy of the algorithm over training and testing sets were found to be around **85%**.
- This proved that the algorithm was able to predict the outputs of the training and the testing tests with similar accuracies.
- Since the accuracies are similar, it can be established that overfitting and underfitting have not occurred.
- The best accuracy was observed for the below model-
    - Input layer with 78 neurons.
    - Output layer with 15 neurons.
    - Model with 5 hidden layers.
    - First hidden layer with 312 neurons.
    - Second hidden layer with 260 neurons.
    - Third hidden layer with 195 neurons.
    - Fourth hidden layer with 130 neurons.
    - Fifth hidden layer with 65 neurons.
    - Average values of input was calculated over the past 30 seconds within a range of 180 seconds.
    - ReLu activation was used for all layers except the last.
    - Softmax activation was used for the last layer.
    - Learning rate is 0.5.
    - Loss function is Binary Cross Entropy.
    - Batch size is 1000.
    - Epoch is 100.

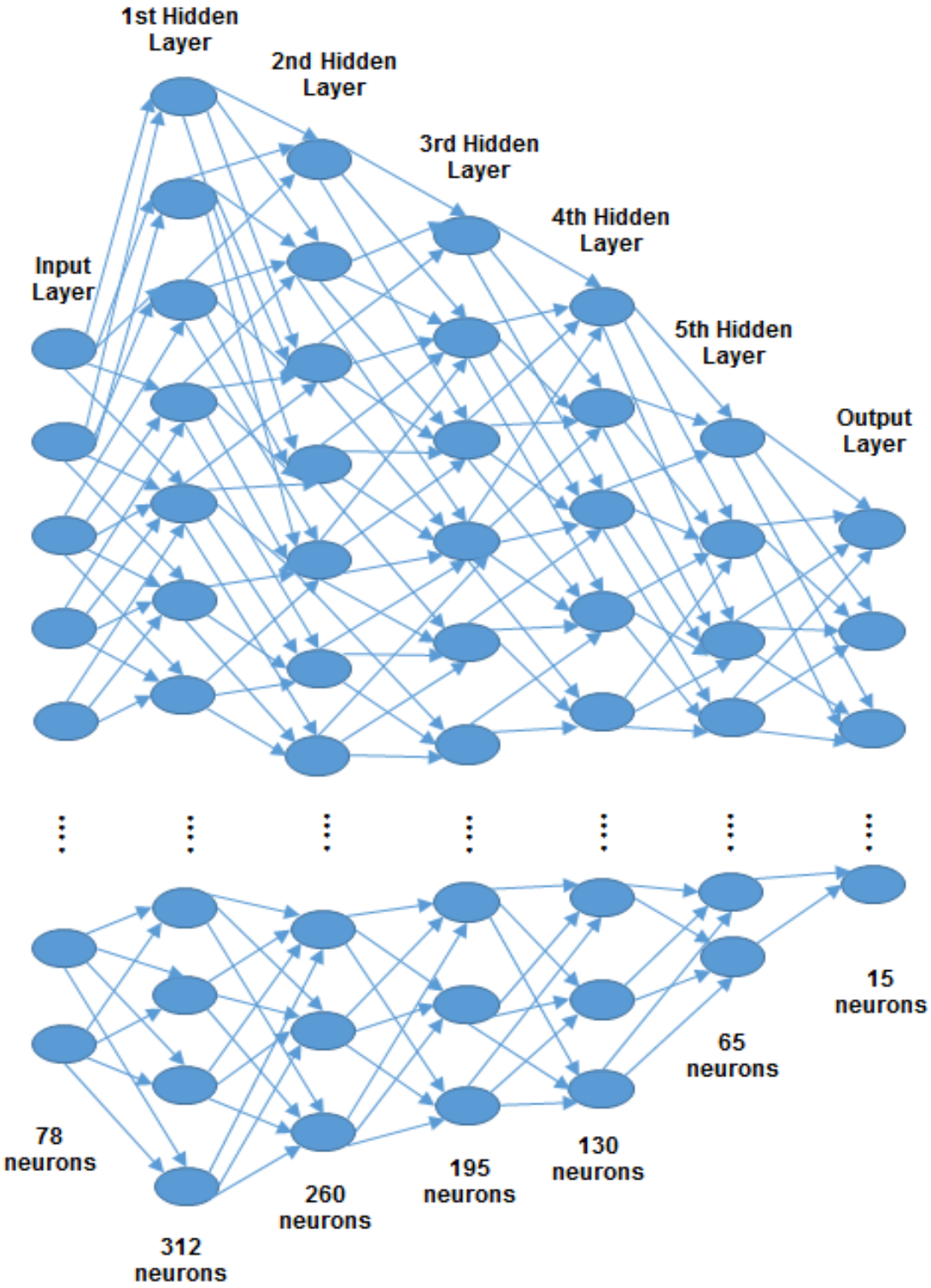


Figure 6.3 Best Accuracy Model

6.3 Comparison to State of the Art

- The algorithm was able to predict the error with a high accuracy based on the sensor values, within a short time.
- The algorithm when implemented on the car's ECU would have the results as shown in the table below-

Parameter	State-of-the-Art	With Implemented Algorithm
Fault details	High-level	Specific cause
Fault diagnosis	Many hours	Few minutes
Wastage of spare parts	High (when exact diagnosis is not done)	Low (since specific cause is informed to the diagnosis team)
Cost	High, due to high fault diagnosis time, and replacement of parts when diagnosis is not correct	Low, since the fault diagnosis is automated and quick

Table 6.1 Comparison of Implemented Algorithm and State-of-the-Art

7 Conclusion

This chapter gives a brief overview of the work done and future prospects.

Section 7.1 explains about the summary of the entire work done.

Section 7.3 explains about the limitations and the future work that could be done on the topic.

7.1 Summary

This thesis was done to investigate whether a machine learning algorithm could be implemented to classify the various possible errors of the SCR system. The entire process was done on Python 3.5 using multiple libraries such as Keras, numpy, sci-kit learn and pickle.

This report gives an overview of the SCR system, and highlights its importance in today's world. The algorithm for the error classification was explained in detail. A brief overview of machine learning and neural networks has also been included in the report for the better understanding of the algorithm.

The steps of implementation of the algorithm was explained in detail, starting from the data collection till the learning of the algorithm by the neural network. Feature selection, feature engineering, feature scaling and the splitting of the dataset into training set and testing set are also explained as part of the implementation process.

The implemented algorithm was then validated on the test dataset, where an accuracy of 85% was observed. This shows that the machine learning algorithm is effective in pin-pointing the exact error of the SCR 85% of the time.

7.2 Outlook

Since the collection of real-time data for this thesis was not possible due to strict laws regarding driving a car with a faulty SCR system, only simulated data was used in the hopes that the simulated data would be similar to the real data.

Though IAV has over 90GB of data for the implementation and evaluation of the algorithm, it needs to be noted that only 70% of the data could be used for training the algorithm to avoid over-fitting.

Even with these limitations, the algorithm is still ready to be implemented into the ECU of the car. However, as of now, it would only be accurate 85% of the time.

With more data, this algorithm could be made even more accurate, and pinpointing the SCR errors could one day become 100% precise.

Bibliography

[1] World Health Organization: http://www.who.int/airpollution/en/ [Online; accessed May-2018]

[2] Green, J.: Effects of Car Pollutants on the Environment (2018)

[3] Icopal-Noxite: http://www.icopal-noxite.co.uk/nox-problem/nox-pollution.aspx [Online; accessed May-2018]

[4] OBRIEN-envproject: https://obrien-envproject.wikispaces.com/HaileeZiehrman+made+vs.+natural+air+pollution [Online; accessed May-2018]

[5] Koebel, M., Elsener, M., Kleemann, M.: Urea-SCR: a promising technique to reduce NOx emissions from automotive diesel engines (2000)

[6] Marine Insight: https://www.marineinsight.com/maritime-law/imo-mepc-66-noxregulations-arguments-scr-technology/ [Online; accessed May-2018]

[7] Diesel Technology Forum: https://www.dieselforum.org/about-clean-diesel/what-is-scr [Online; accessed May-2018]

[8] European Automobile Manufacturers Association: http://www.acea.be/industry-topics/tag/category/diesel-exhaust-fluid-adblue [Online; accessed May-2018]

[9] Chen, R.: Model-Based Fault Detection and Diagnosis of Selective Catalytic Reduction Systems for Diesel Engines (2014)

[10] Herman, A.D., Wu, M., Cabush, D.D.: Diagnostic Methods for Selective Catalytic Reduction (SCR) Exhaust Treatment System (2010)

[11] Ma, S., Meng, F.: Modelling, Analysis, and Experimental Testing of a Selective Catalytic Reduction Dosing System (2017)

[12] Benjamin, S.F., Roberts, C.A.: Significance of droplet size when injecting aqueous urea into a Selective Catalytic Reduction after-treatment system in a light-duty Diesel exhaust (2012)

[13] Buecker, B.: Selective Catalytic Reduction: Operational Issues and Guidelines (2011)

[14] Truck Parts and Service: https://www.truckpartsandservice.com/the-abcs-of-def/ [Online; accessed May-2018]

[15] Blue Plus: http://www.blueplus.com.hk/web/en/scr_system.html [Online; accessed May-2018]

[16] Jääskeläomem, H., Majewski, W.A.: Urea Dosing Control (2018)

[17] Lyons, A.: On Board Diagnositcs (OBD) Program Overview (2015)

[18] Eijnden, E.v.d., Cloudt, R., Willems, F., Hiejden, P.v.d.: Automated model fit tool for SCR control and OBD development (2009)

[19] DieselNet: https://www.dieselnet.com/standards/us/obd.php [Online; accessed May-2018]

[20] Sanchez, F.P., Bandivadekar, A., German, J.: Estimated Cost of Emission Reduction Technologies for Light-Duty Vehicles (2012)

[21] Analytics Vidhya: https://www.analyticsvidhya.com/blog/2014/10/ann-worksimplified/ [Online; accessed May-2018]

[22] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning (2016)

[23] Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E.B., Turaga, D.: Learning Feature Engineering for Classification (2017)

[24] Kodratoff, Y.: Introduction to Machine Learning (2014)

[25] Expert System: http://www.expertsystem.com/machine-learning-definition/ [Online; accessed May-2018]

[26] SearchEnterpriseAI: https://searchenterpriseai.techtarget.com/definition/machine-learning-ML [Online; accessed May-2018]

[27] Brownlee, J.: Start Machine Learning (2015)

[28] Brownlee, J.: Machine Learning Algorithms (2016)

[29] Schalkoff, R.J.: Artificial Neural Networks (1997)

[30] Semantic Scholar: https://www.semanticscholar.org/topic/Keras/494839 [Online; accessed May-2018]

[31] Github: https://github.com/Theano/Theano [Online; accessed May-2018]

[32] GitHub: https://github.com/tensorflow/tensorflow [Online; accessed May-2018]

[33] Python: https://www.python.org/ [Online; accessed May-2018]

[34] Anaconda: https://anaconda.org/anaconda/spyder [Online; accessed May-2018]