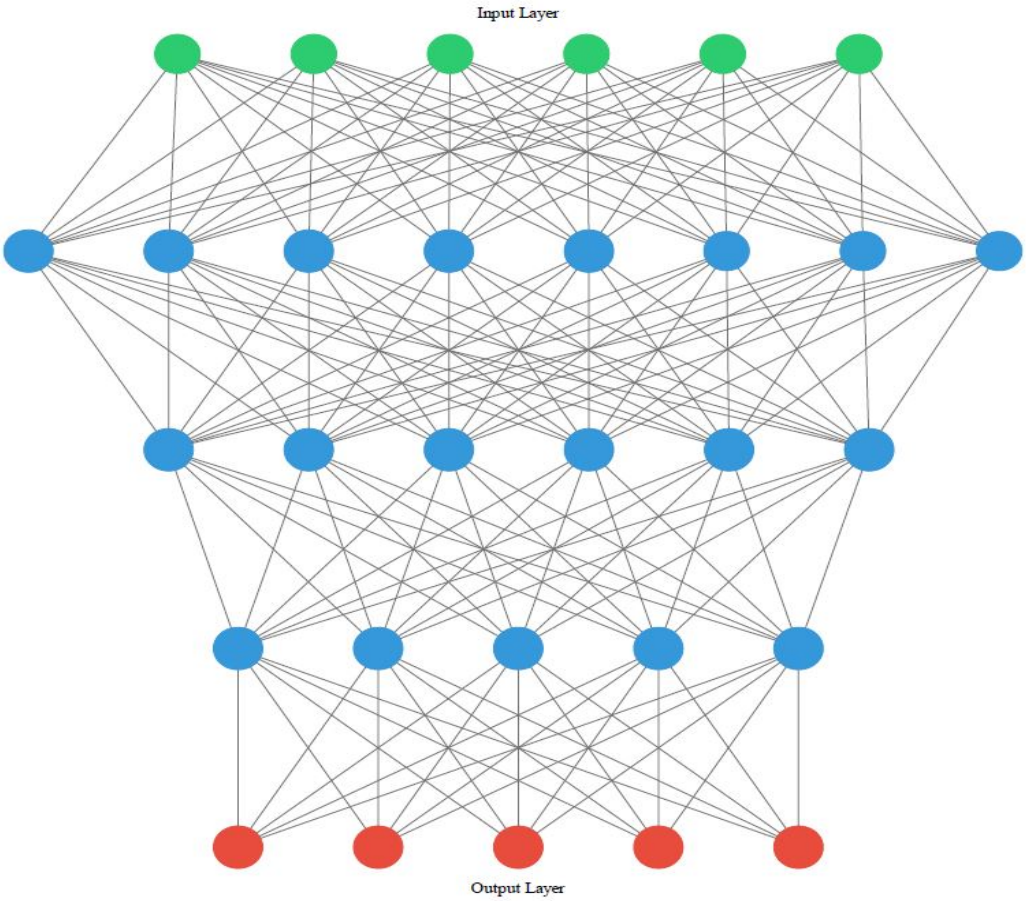
CHALMERS
UNIVERSITY OF TECHNOLOGY



Increased Energy Efficiency with Deep Learning Methods

Fuel Economy Optimisation of a Plug-In Hybrid vehicle for a Drive Cycle using AI methods.

Master's Thesis in *Electrical Engineering*

Anup Vasu Padaki
Nishwa Ishfaq

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Master's thesis 2019

Optimising the Fuel Economy of a Plug-In Hybrid Vehicle for a Drive Cycle with Deep Learning methods.

The paper explains the role of Deep Learning in optimisation of the Fuel consumed by the Plug-in Hybrid Vehicle considering all the constraints and the complexities involved.

Anup Vasu Padaki
Nishwa Ishfaq

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Optimising the Fuel Economy of a Plug-In Hybrid Vehicle for a Drive Cycle with Deep Learning methods.

The paper explains the role of Deep Learning in optimisation of the Fuel consumed by the Plug-in Hybrid Vehicle considering all the constraints and the complexities involved.

Anup Vasu Padaki
Nishwa Ishfaq

Supervisor: Anna Rimark, CAE Energy, CEVT AB
Examiner: Prof. Tomas McKelvey, Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg.
Telephone +46 31 772 1000

Cover: The Cover Figure shows a fully connected feed forward neural network implemented in the dissertation.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2019

Increase Energy Efficiency with Deep Learning Methods
Optimising the Fuel Economy of a Plug-In Hybrid Vehicle for a Drive Cycle with Deep Learning methods.
Anup Vasu Padaki
Nishwa Ishfaq
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Plug-in Hybrid Vehicles are complicated with many interacting sub-systems and multiple energy sources. It is a challenging task to develop model based controllers that fulfils the requirements and are optimal in energy consumption.

This thesis research focuses on deriving an AI controller by using tools from optimisation and Deep Machine Learning. An Artificial Neural Network is proposed to control the gear shifting, power-splits and the PHEV modes. The parameters of the network are learned by optimising the fuel consumption while satisfying the constraints using simulations of the complete New European Drive Cycle. Validation tests for a simulated vehicle with a conservative plant model show that the AI controller consumes less fuel than the baseline controller while satisfying the SoC neutrality constraint.

Keywords: Neural Network (NN), Deep Learning, Optimisation, Loss function, SoC, Fuel Economy, Genetic Algorithm, Plug-in Hybrid Vehicle (PHEV).

Acknowledgements

Throughout the writing of this dissertation we have received a great deal of support and assistance. We would like to offer our greatest token of appreciation to our thesis examiner and supervisor Prof. Tomas Mckelvey, whose expertise and guidance was valuable in formulating the research questions and methodology. We would also like to thank our supervisors Anna Rimark and Borja Rojo at the industry, who has provided valuable feedback and resources for the successful completion of the project. We also would like to thank our colleagues who have supported us in one way or another throughout the dissertation. Finally, we would also like to thank our friends and family for their unconditional love and support.

Anup Vasu Padaki, Nishwa Ishfaq
Gothenburg, June 2019

Contents

List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DCTH	Duel Clutch Transmission Hybrid
BSFC	Brake Specific Fuel Consumption
DML	Deep Machine Learning
EM	Electric Machine
FMU	Functional Mock-up Interface
GA	Genetic Algorithm
HEV	Hybrid Electric Vehicle
IID	Independent and Identically Distributed
ICE	Internal Combustion Engine
ML	Machine Learning
NEDC	New European Driving Cycle
NN	Neural Network
PHEV	Plug-In Hybrid Vehicle
RNN	Recurrent Neural Network
RL	Reinforcement Learning
RLC	Road Load Curve
ReLU	Rectified Linear Unit
SoC	State of Charge
SysID	System Identification
TCU	Transmission Control Unit
WLTC	Worldwide Harmonised Light Vehicles Test Procedure

b	Bias Matrix
f	Objective Function
g_i	Gear Number
$g_{i_{ice}}$	Internal Combustion Engine Gear Number
$g_{i_{em}}$	Electric Machine Gear Number
g_r	Gear ratio array
$\dot{m}_f$	Fuel Flow Rate
n	length of a vector

N	Engine Speed
w	Weight Matrix
X	Input Vector or Feature Vector
$\hat{y}$	Predicted Value
y	True Value
$\mathcal{L}$	Loss
Ω	Regularisation Function
τ_{dem}	Torque Demand
ω_{wheel}	Wheel Speed
τ_{ice}	Engine Torque
τ_{em}	Electric Machine Torque
τ_{eg}	Electric Generator Torque
ω_{em}	Electric Machine Speed
ω_{eg}	Electric Generator Speed
P_{dem}	Power demand
P_{ice}	Engine Power
P_{em}	Electric Machine Power
P_{eg}	Electric Generator Power

1

Introduction

1.1 Background

Protection of the environment and improvement of air quality is an important objective of the different automotive markets. In the automotive industry, legislation and standards aim to reduce the emission of CO_2. In order to achieve this goal, the automotive industry has developed throughout the years different solutions such as electric vehicles and hybrid vehicles. During the last decade, Hybrid Electrical Vehicles (HEVs) were introduced to the mass market as a means to improve the fuel economy of conventional vehicles by adding an electric machine (EM) and energy buffer to the powertrain. Compared to conventional vehicles, HEVs reduce the fuel consumption and emissions due to several reasons: downsized engine, regeneration of the braking energy, and the higher efficiency gained from the extra freedom in choosing the engine operating point. Plug-In Hybrid Vehicles (PHEVs) are the next generation of HEVs. Unlike HEVs which only operate in charge sustaining mode, where the State of Charge (SoC) in the battery is fluctuating around a fixed level. PHEVs can also, due to their high capacity storage, operate in charge depletion mode where the vehicle drives a limited range on electricity from the grid. However, this requires a controller to decide when to switch between different modes while discharging the battery to a lower limit at the end of the driving cycle [7].

The PHEV technology requires new sets of power-train parallelisation and control strategies which need to cover a wider design space due to the increased number components and possible combinations. In such scenarios where there are a large number of variables and a complex interaction between them, Machine Learning (ML) and Deep Machine Learning (DML) methods have been used in industry with high levels of success [8].

The *Fuel Economy and Performance* team at *CAE Energy* department at **CEVT AB** is working on evaluating control strategies for a PHEV by running simulations of controllers and plant models. The plant models and controllers are imported to IPG CarMaker as Functional Mock-up Interfaces (FMUs) for running a simulation. This loop involves a lot of tools and consumes a considerable amount of time.

1.2 Problem Description

PHEVs are complicated in nature due to many interacting systems and sub-systems. The Engine Control Unit (ECU) and the Transmission Control Unit (TCU) are a good example of complex systems with numerous signals to control that may interact or overlap with one or many other signals. The TCU or the ECUs use model based controllers or sometime may use shift maps from the tests carried out and are not generally optimal in nature. The optimal control of these control systems can significantly increase the energy efficiency by minimising the fuel consumption of the automobile. The developed look-up table or model-based controllers are solved or say simulated using IPG CarMaker. IPG CarMaker allows to use the models from different tools like MATLAB/Simulink, Dymola, GT Suite etc. These models are imported as FMUs. These process is usually tedious and consumes a lot of time. The key objective is to investigate if it is feasible to increase the energy efficiency of the PHEV using the DML techniques. An additional objective also is to develop a much faster vehicle simulation platform using programming, where everything is designed or developed on a common platform. DML or ML techniques have proved to be quite efficient in non-linear and complicated problems[8]. The feasibility, design, analysis and evaluation of *Deep Machine Learning based controls* is carried out and is explained in the further sections of the dissertation. Therefore the need is to build an AI machine that can learn the key parameters effecting the energy efficiency per say Fuel consumption and optimise the fuel consumed over a specific drive cycle. An additional objective of the thesis is to have a faster vehicle simulation platform that runs a vehicle on a fixed drive cycle. The results could play an important role in engineering later on.

The problem can simply be stated as:
Minimise the fuel consumption of a plug-in hybrid vehicle by having a smart (AI) TCU with a better power management strategy for a complete New European Drive Cycle (NEDC)

1.2.1 Aim

The aim of the thesis work is to optimise the Fuel Consumption of a PHEV considering the different operating modes and TCU actions over a complete drive cycle.

1.2.2 Research Questions

- What are the challenges involved in hierarchical control system of a PHEV?
- What are the input, output and feedback signals of control system that influence the Fuel Economy, can they be used as inputs and outputs?
- What type of Deep Learning or Machine Learning technique is suitable?
- What kind of loss function can be used? What parameters is the loss function of?

1.2.3 Scope

The scope of the thesis is develop an AI controller that replaces the TCU for simulation purposes. The AI does not have practical system implications and replications like drive-ability and other conditions as yet.

1.3 Disposition

First, theory relevant for understanding the problem and the proposed solutions is presented in Chapter 2. Then the methods for implementing the theory and how it leads to the solution is explained in Chapter 3. The results are visualised and inferred in Chapter 4 and finally the main conclusions and discussions are given in Chapter 5.

1.4 Contribution

In recent years, deep learning has garnered tremendous success in a variety of application domains. This new field of machine learning has been growing rapidly, and has been applied to most traditional application domains, as well as some new areas that present more opportunities. Different methods have been proposed based on different categories of learning, including supervised, semi-supervised, and unsupervised learning. Experimental results show state-of-the-art performance using deep learning when compared to traditional machine learning approaches in the fields of image processing, computer vision, speech recognition, machine translation, art, medical imaging, medical information processing, robotics and control, bio-informatics, natural language processing (NLP), cybersecurity, and many others [9]. The dissertation investigates the feasibility on implementing the Function Approximation techniques and Deep Learning models in automotive industry.

2

Theory

This chapter briefly describes the theory behind PHEVs, Function Approximation, Deep Learning Techniques and Optimisation implemented for a PHEV with different modes. Energy optimisation in a PHEV is a fairly complicated task with a lot of constraints and difficulties as two energy storage buffers and two power sources are utilised to propel the vehicle.

2.1 Introduction to Plug in Hybrid Vehicles

2.1.1 Hybrid Electric Vehicles and their Configurations

Hybrid Electric vehicles (HEVs) are those where two or more energy sources are combined to propel the vehicle. The most common energy sources are the Internal Combustion Engine (ICE) and the Electric machine (EM). HEVs can be categorised into different types based on their architecture of power sources. Parallel hybrid, Series hybrid and Parallel-Series hybrid vehicles are well known types of HEVs. Plug-in hybrid vehicles are another topology in which the external sources can be used to charge the battery. The external source can be an ICE or an electric grid. These vehicles are developed to get better fuel economy and less green house emissions. These vehicles are taking over automotive market due to the electric range and their impact on the environment.

Series HEV

In series hybrid vehicles only an electric machine propels the vehicle and ICE extends the electrical range by converting mechanical energy into electrical energy. The battery energy can be restored by charging from an ICE and from electric grid [10]. The combustion engine is connected to an electric generator (EG) which converts the mechanical power to electrical power, to charge the battery or to propel the vehicle. The Series hybrid energy flow diagram is shown in the Figure 2.1. The strategy is to keep the engine running at its most efficient point and the EM drive the vehicle.

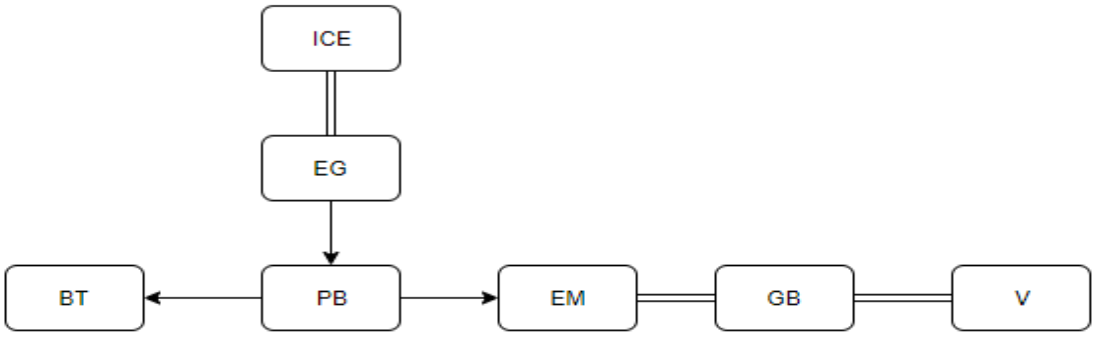


Figure 2.1: Basic series hybrid configuration,(EG: electric generator, PB: power link, BT: battery, EM: electric machine,GB: Gearbox , V: Vehicle). Double lines: Mechanical link, Solid arrows: Electrical link [1].

Parallel HEV

The two power sources EM and ICE are connected to the wheels. Thus, both the ICE and the EM can be used to propel the vehicle. The total traction power is a combination of energy sources, the size of the machines can be set for a part of the maximum required power. When the vehicle is propelled only with the EM, the engine can be disconnected, whereas on the other hand, when the ICE is driving the vehicle, the EM can be used as a electric generator to charge the battery conditioned on SoC levels, electrical energy can also be restored by regenerative braking or ICE can also be used to recharge the battery [10]. The energy flow diagram of Parallel hybrid vehicle is shown in the Figure 2.2.

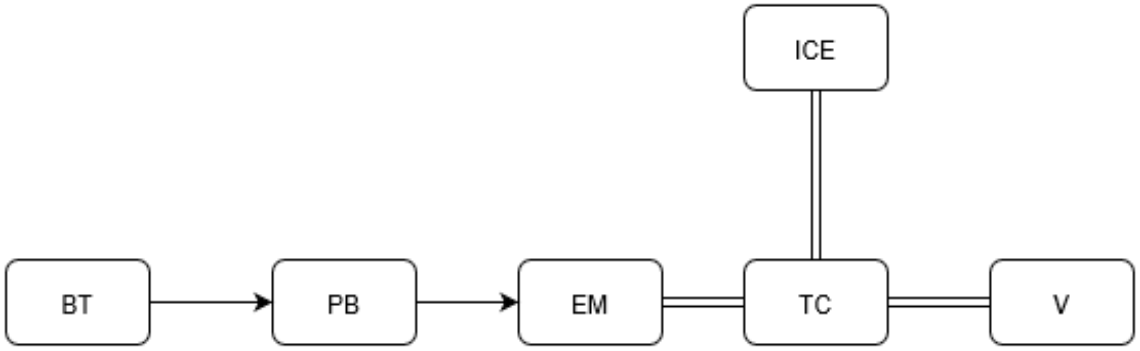


Figure 2.2: Basic Parallel hybrid configuration, (PB: power link, BT: battery, EM: electric machine, TC: torque coupler, V: vehicle). Double lines: Mechanical link, Solid arrows: Electrical link [1].

Parallel-Series/Combined HEV

The series and parallel hybrids are the first developed hybrid architectures. These architectures are combined to achieve the best of both HEVs in one design. Combined HEVs consist of both setups of parallel and series hybrid vehicles. A parallel

hybrid has been acquainted with some series HEV features to achieve better efficiency. The vehicle can operate in pure electric mode, in ICE mode or in a combined mode. There are two distinct electric machines present in Combined HEVs, as in a parallel hybrid configuration, one is used as a prime mover or for regenerative braking. The other machine acts like a electric generator in a series hybrid system. It is used to charge the battery via the engine or for the stop-and-start operation. It can also operate in series mode using the combustion engine only to power the electric generator operation[1, 10]. The combined HEV energy flow diagram is shown in the Figure 2.3.

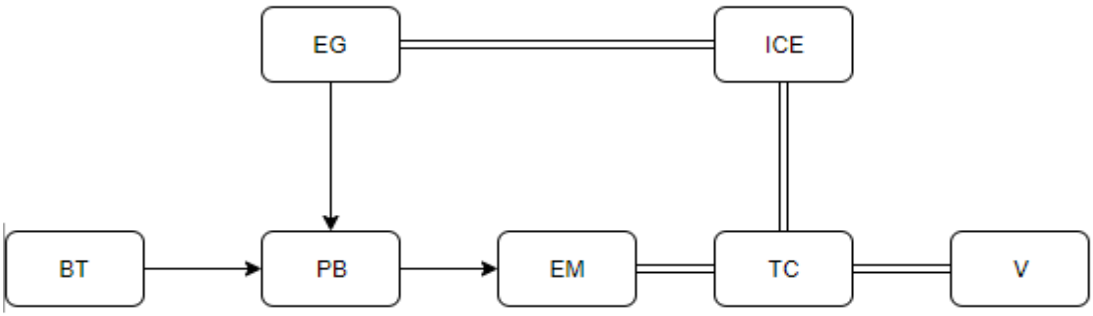


Figure 2.3: Configuration of a combined hybrid, (EG: electric generator, PB: power link, BT: battery, EM: electric machine, TC: torque coupler, V: vehicle). Double lines: Mechanical link, Solid arrows: Electrical link [1].

Plug-in Hybrid (PHEV)

Plugin Hybrid vehicles give the advantages of both electric and hybrid vehicles. Compared to HEVs, PHEVs have a larger battery capacity, longer driving range, and a better fuel economy with reduced emissions.

In PHEVs the battery can be recharged from the grid, like in battery electric vehicles (BEVs) and the ICE can be used when the vehicle is running. Therefore, the PHEVs are not limited to charge-sustaining operation (CS) as are non rechargeable hybrids. During driving, the battery can be continuously discharged until a lower limit of battery SoC is attained [1]. The range covered in the charge-depleting (CD) mode is the all-electric range (AER).

Degree of hybridisation(DoH) is the ratio between the maximum electric power($P_{em,max}$) and the total maximum power of engine $P_{ice,max}$ and electric machine$P_{em,max}$. One of the advantage of higher DoH is reduction in energy consumption and on the other hand it comes with additional costs. The DoH is expressed as:

$$DoH = \frac{P_{em,max}}{P_{em,max} + P_{ice,max}}. \tag{2.1}$$

2.1.2 Components of a PHEV

Internal Combustion Engine(ICE)

ICE is one of the energy source for HEVs. It converts fuel energy to mechanical energy. The thermodynamic efficiency of ICE is defined by the Equation as:

$$\eta_{ice} = \frac{\omega_{ice}.\tau_{ice}}{P_c} \tag{2.2}$$

Where ω_{ice} is the engine regular speed, τ_{ice} the engine torque and P_c is the enthalpy flow associated with mass fuel flow rate $\dot{m}_f$. And mass fuel flow rate is expressed as:

$$\dot{m}_f = \frac{P_c}{H_l}, \tag{2.3}$$

Where H_l is the fuels lower heating value. The thermodynamic efficiency (η_{ice}) of ICE mainly depends on the engine speed and the engine torque.

Electric Machine

Electric machines are used as starters and alternators in conventional vehicles. Starters helps to boost the engine to reach its idle speed and to start delivering torque. The alternators generate electricity in order to charge the small (i.e 12 V) battery and to power the electric auxiliary loads. In electric vehicles and HEVs, the electric machine is the main energy converter. The EM acts as a reversible machine [1], which can be used as:

- *Electric Mode*: to convert the electrical power from the battery into mechanical power to drive the vehicle.
- *Generator mode*: to convert the mechanical power from the engine into electrical power to recharge the battery and also to recuperate mechanical power available at the drive train to recharge the battery during deceleration (regenerative braking).

In parallel hybrid vehicles, PHEV's and in electric vehicles traction and generation can be done by the EM. In series hybrid vehicles and combined hybrid vehicles two different machines are needed for generation and traction [1].

Battery

Electro-chemical batteries are a main part of both electric vehicles (EVs) and hybrid-electric vehicles (HEVs). Batteries transform chemical energy into electrical energy when propelling a vehicle and vice versa while charging. They act as a reversible electrical energy storage system. Traction batteries are designed according to desired power of the electric path, and nominal capacity in order to match the desired driving range specifications. Nominal capacity is the integral of the current and it could be delivered by a full battery when completely discharged under certain circumstances. The state of charge (SoC), is the capacity remaining in the battery to provide energy. SoC is a dimension less unit and is represented as a percentage of its nominal capacity. The most important key parameters of batteries for EV and

HEV applications are high specific energy, high specific power, long calendar and cycle life, low initial and replacement costs, high reliability, wide range of operating temperatures and high robustness. The specific energy is the energy that can be stored in the battery per unit mass and its unit is Wh/kg [1].

Gear Box

The gear box is used to transform the mechanical power at speed ω_1 and τ_1 to another speed and torque ω_2 and τ_2 and the relationship is shown below in the Equations 2.4 and 2.5.

$$\omega_1 = \omega_2 . r_i \tag{2.4}$$

$$\tau_1 = \frac{\tau_2}{r_i} \tag{2.5}$$

Seven Dual Clutch Transmission for Hybrid - 7DCTH

In 7DCTs the two clutches are arranged concentrically, the larger outer clutch drives the even numbered gears and the smaller inner clutch drives the odd numbered gears. Gear shifting can be done without interfering torque distribution to the driven road wheels. Alternate gear ratios can pre-select an odd gear on odd gear shaft while the vehicle is being driven in an even gear on even shaft and vice versa. 7DCTs are the fastest-shifting road car transmission and are able to shift faster than a professional racing driver using a manual transmission. 7DCTs can shift more quickly than cars equipped with single clutch automated manual transmissions (AMTs) also called single clutch semi-automatics. The 7DCT offers faster gear shift times and a better fuel economy improving the efficiency compared to the other transmission types [11]. 7DCT is shown along with the PHEV architecture in the Figure 2.4.

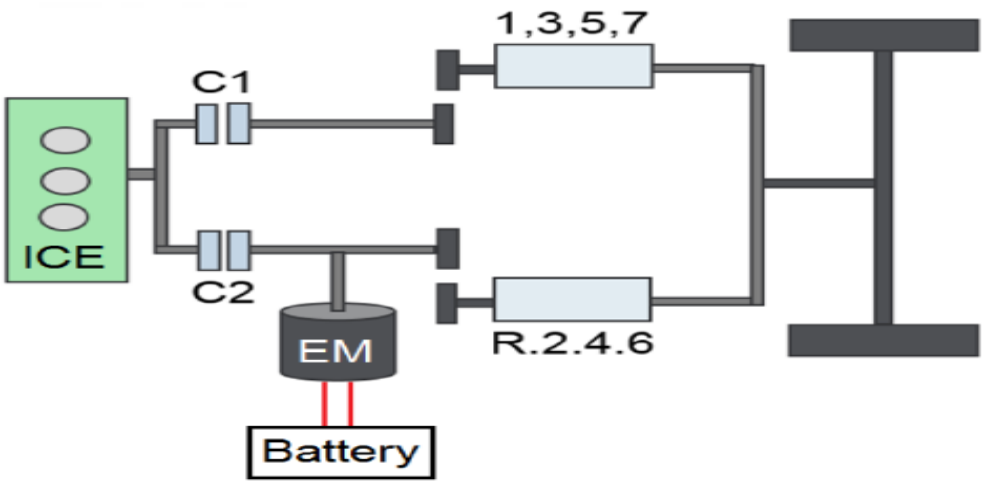


Figure 2.4: The figure shows the PHEV architecture with 7 Dual Clutch Transmission C1 and C2.

C1 and C2 are the two clutches that are connected to odd and the even gear shafts respectively. ICE is connected to both the gear shafts through both the clutches while the electric machine is only connected through even gear shafts.

Drive Cycle

The drive cycles are the vehicle trajectories along a defined path. A drive cycle is defined by a velocity profile across time for a specific road. A NEDC speed profile is shown below in the Figure 2.5 that has been used extensively in the dissertation.

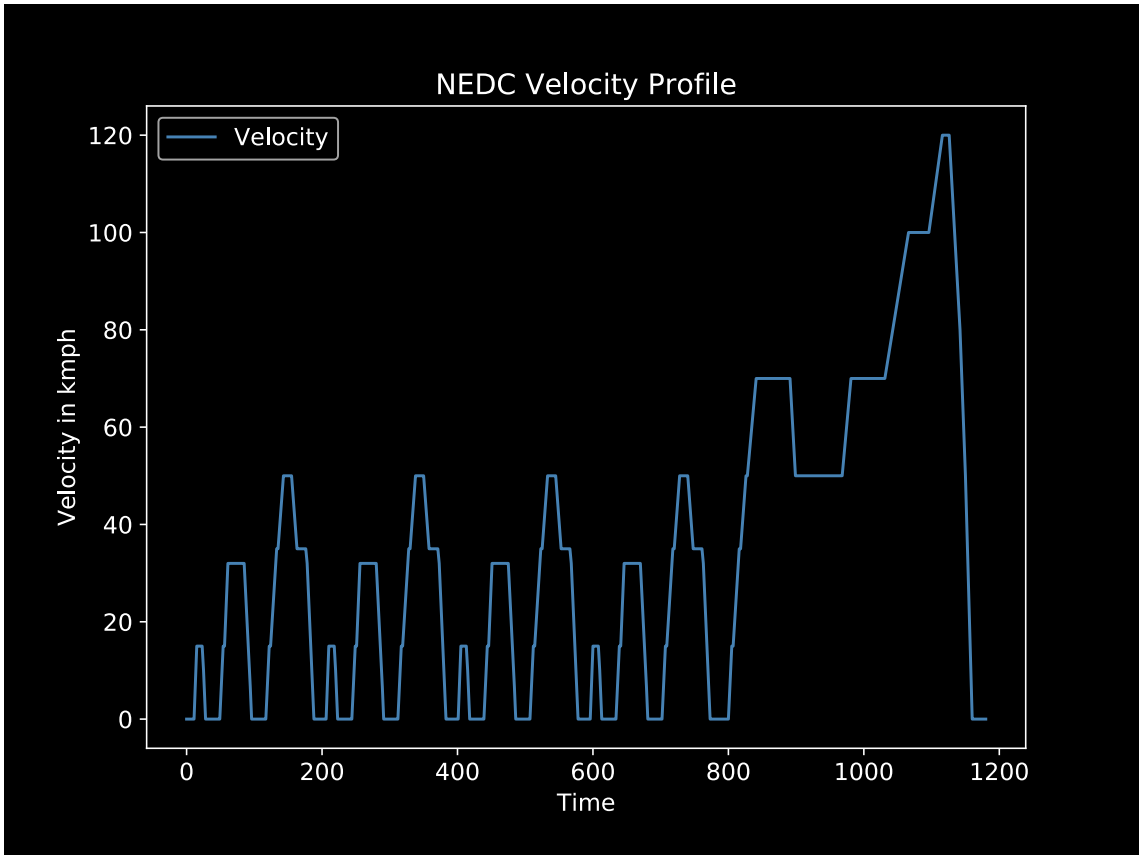


Figure 2.5: NEDC speed profile, where the maximum speed attained is 120 kmph and the average speed is 33.35 kmph.

2.2 Function Approximation

Function Approximation or *System Identification (SysID)* deals with the problem of building mathematical models of dynamical systems based on observed data from the system [2]. The first IFAC Symposium on Identification started in 1967, which more or less indicates the time that SysID became a mature research area [12]. The SysID has a widespread applications and implementations because of the abundance of the dynamical systems present in the nature. The SysID procedures come in handy when it is difficult to get all the dynamics into the mathematical model,

sometimes SysID is also referred to as Data Driven Control or Model Free Control. SysID techniques can utilise both input and output data (e.g. eigen system realisation algorithm) or can include only the output data (e.g. frequency domain decomposition). Typically an input-output technique would be more accurate, but the input data is not always available.

It is very difficult to get a mathematical model that captures the dynamics of the various sub-systems of the PHEV like ICE [13], EM, Battery, 7DCTH [11]. The mathematical models can be complicated considering the states, hidden sates, inputs and feedback and the hierarchy of the control system. Developing a controller for these mathematical models that ensures the stability, robustness and optimal behaviour can be extremely challenging [14].

2.2.1 The SysID Procedure

A mathematical model is a close replication that reflect the dynamics of the system. In reality, the rapidly changing dynamics, non-linearities, disturbances, limited prior knowledge and the incomplete availability of observed data prevent an exact mathematical model description of the system. The complexities involved in developing and analysing the mathematical model makes it less desirable. SysID is an alternative that approximates a model for a specific application on the basis of knowledge and data available [12]. The Figure 2.6 shows the SysID procedure for a problem.

2.2.2 A Priori, Knowledge and Objectives

A Priori

Model set selection is basically determined by the available information. The more information is available the better model can be constructed and the more similarity will be between the system and its model. Based on *prior* information we can speak about white box, grey box or black box models [15, 2].

Knowledge

The data of input signals and the output estimates plays an vital role in the identification of the system. The selection of input-output signals are also dependent on the prior knowledge of the system. Various constraints may be set on the data to make the data maximally informative to identify the system [2].

Objectives

It is always important to know the estimates or the signal that the system approximates. Sometimes, the exact input-output labelled data can also be solved using machine learning. The Approximation function can be selected based on the objective or the goal of the system.

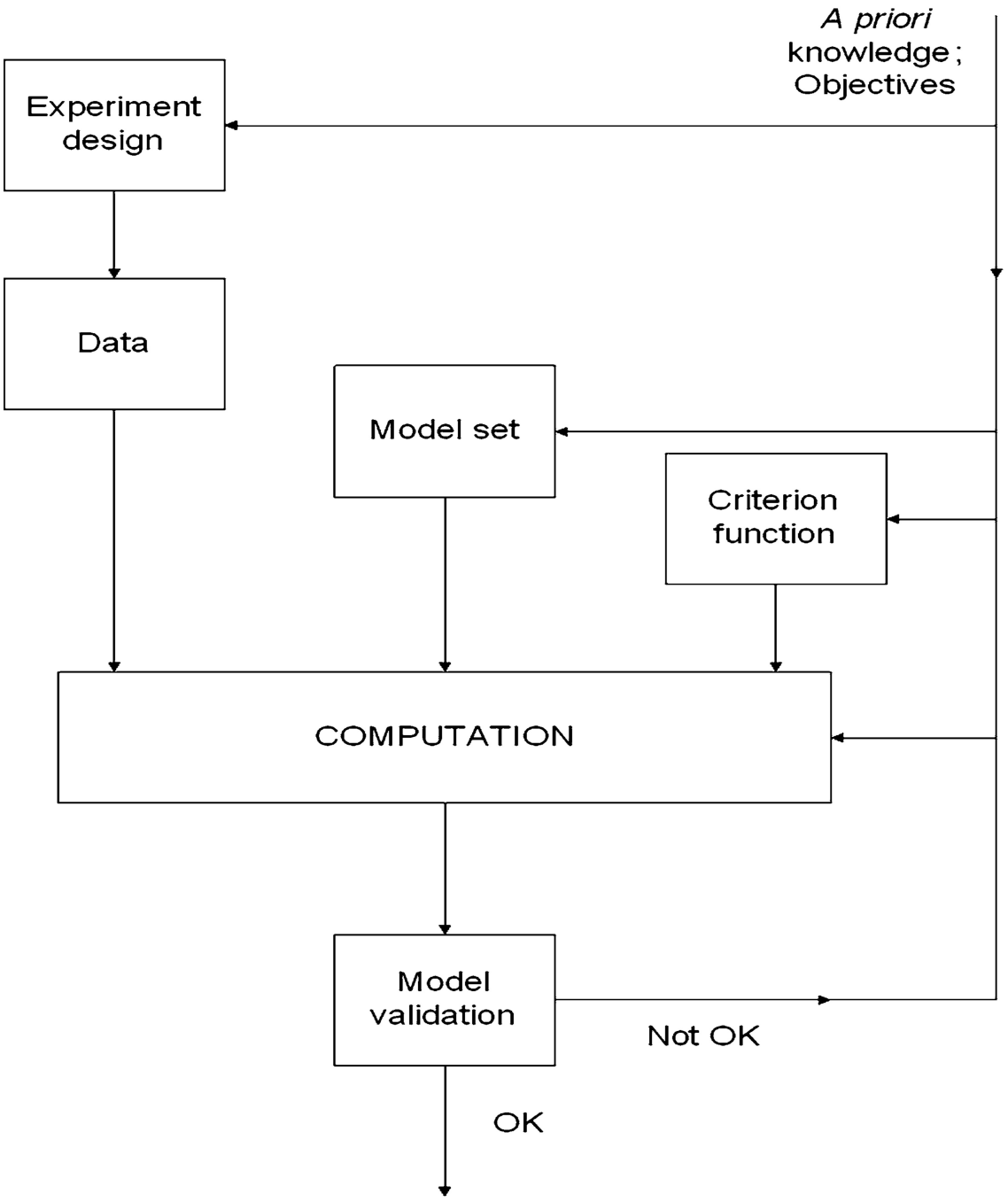


Figure 2.6: The flowchart [2] that shows the SysID procedure. Having a prior information of the data to a model that predicts the outputs from the priori.

2.3 Deep Learning

DML is a class of machine learning algorithms that learns multiple levels of feature representations that correspond to different levels of abstraction. DML has been a been a buzz word since a decade and has been evolving very rapidly since then. From the applications of classification to continuous regression it has a wide range of applications. DML sometimes specifically is also referred to as AI. The first wave of DML dates back to 1940s. The DML was referred to as different names like *Cybernetics* in 1940s - 1960s and *Connectionism* in 1980s -1990s, and current resurgence known as **Deep Machine Learning** or *Artificial Neural Networks (ANN)* [16]. The Figure 2.7 shows the evolution of how Deep Learning has evolved through the course of time.

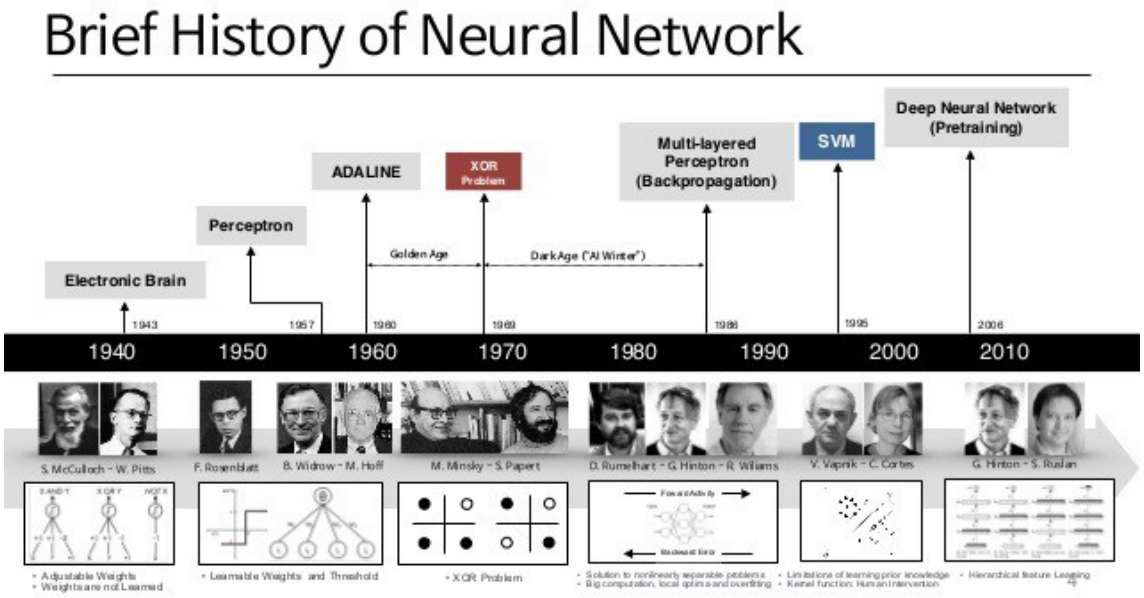


Figure 2.7: The figure shows the historical evolution of DML.

2.3.1 Artificial Neural Network

ANNs are biologically inspired computational models, but not identical to the biological brain. Mathematically speaking ANNs are non-linear function approximators. The corresponding DML models can be referred to as the engineered systems inspired by the biological brain. ANNs are functions of input (X), weights (w) and biases (b). The simplest and the first implementation of an ANN dates back to 1958 [17]. The definition of the perceptron is as follows:

$$f(x) = \begin{cases} 1 \;\; if \;\; w^T x + b > 0 \\ 0 \;\; otherwise \end{cases} \tag{2.6}$$

The below Figure 2.8 shows an illustration of a simple perceptron, the activation function is a simple step response function. The perceptron is used to classify linear objects. The predicted values are separated with a hyper-plane.

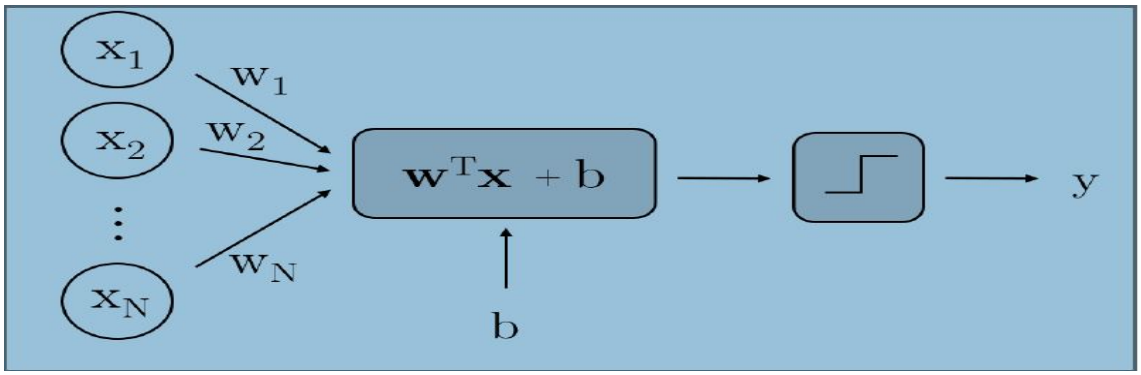


Figure 2.8: The figure shows the illustration of a perceptron implemented as a Linear Classifier.

$$\hat{y} = step(w^T x + b) \tag{2.7}$$

The Equation 2.7 shows the implementation of a simple perceptron (as shown in the Figure 2.8), the output predicted values($\hat{y}$) are with input (X), weights (w) and bias (b).

2.3.2 Activation Functions

The activation functions in the below Table 2.1 are some of the very common Non-Linear functions implemented in the NNs.

Activation Functions	Equations
Step	$f(x) = \begin{cases} 1 \ if \ \ x > 0 \\ 0 \ otherwise \end{cases}$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
ReLU	$f(x) = \begin{cases} x \ if \ \ x > 0 \\ 0 \ otherwise \end{cases}$
Leaku ReLU	$f(x) = \begin{cases} x \ if \ \ x > 0 \\ \alpha x \ otherwise \end{cases}$
Tanh	$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

Table 2.1: The Table shows some of the most common activation functions used in DML. Custom activation functions can also be used if the application demands it.

A very basic and simple activation function is like a switch, any value of a input which crosses a certain preset threshold is positive and the rest is zero. These hard threshold or the hard conditioned activation functions like step functions are generally not considered. The Step equation from the Table 2.1 says that the value of *f(x)* is always 0 if $x < 0$, this implies the gradient of the function is mostly zero, this also implies that the direction towards optimal is unknown since gradient

is 0 [18]. Hence learning is difficult when gradient is 0. To avoid such problems, more flexible and non-linear activation functions like Sigmoid, ReLU are used. DML models today widely use ReLU as their activation functions. However, recent studies has shown that the ReLU can be used for deeper networks considering the accuracy and the complexity involved in the computation [19, 20]. The ReLU looks like a linear function but exhibits non-linear behaviour, and in fact any function can be approximated by the combination of one or more ReLUs [21]. The main advantage using ReLU is that the output from the activation function becomes sparse if the input is normalised (0 mean and unit variance). The normalisation say max-min scaling normalisation restricts the inputs to be between 0 and 1. The normalised inputs and ReLU as activation function can help the NN to be much faster and lighter [19, 20, 21]. Unfortunately this can cause problems too. Since everything less than zero will not get activated it can cause the gradient to vanish. Then there will be no weight update during gradient descent and the network can end up in sub-optimal or local optimal. This is often referred to as *the dying ReLU* problem [22]. Today, there are a several extensions of the ReLU [20] that try to evade this problem, one of them is the Leaky ReLU. The idea is to have a small gradient αx where all the gradient goes to 0 [20, 21]. This helps in keeping the learning active.

2.3.3 Data

Data is the most integral part in implementing a NN. Data defines the type of DML and NN architecture for solving the task. For example, supervised learning is most suitable when you have input data and output labelled data. The Table 2.2 shows the selection of a few types of NN architectures and type of DML to be implemented to solve the defined problem considering the available data.

Input Data	**Output Data**	**NN Architecture**	**DML type**
Real Numbers	Real numbers	Feed Forward	Supervised Learning
Images	Image Labels	CNN	Supervised Learning
Data function of time	Varying size Data	RNN	Supervised Learning
Data(any type)	-	any	Unsupervised Learning
Markorov Decicion Process(MDP)	-	any	Reinforcement Learning

Table 2.2: The Table shows the selection of type of DML and NN architecture considering the input and output data and their format.

2.3.4 Loss Function

It is of prominent importance to have a good loss function that can help the NN capture the critical patterns in the data and learn the important features that affect the performance of the NN. The loss function minimisation can be shown mathematically as follows:

$$min \;\; \mathcal{L}(f(X, w, b), y) \tag{2.8}$$

$\mathcal{L}$ is some arbitrary loss function, where as w and b are weights matrix and biases matrix respectively.

Modular open source NN library Keras [23] developed on top of TensorFlow [24] offers a wide range of loss functions like hinge, mean squared error, mean absolute error, cross entropy, categorical cross entropy etc. Keras also offers a means to develop a custom loss function. For the purpose of the thesis, a custom loss function is developed that minimises the, fuel consumption and the battery energy of the PHEV the details of the developed loss function are explained in Chapter 3 (Methods).

2.3.5 Optimisation

The orthodox way of optimising a function is to use gradient based methods. The weights of the NN are updated by computing the gradient. Weights(w) are updated each time the gradient is calculated as seen the in the Equation 2.9 below:

$$w := w - \alpha \bigtriangledown_w \mathcal{L} \tag{2.9}$$

The parameter α is the learning rate, which controls the number of steps to convergence. A small learning rate gives a better convergence but the time taken to reach the minimum increases, consequently a large learning rate can yield a local optimal soon. Thus the algorithms usually have an adaptive learning rate, where the learning rate decreases as the learning progresses on. This process is known as deterministic gradient descent. The deterministic gradient descent performs poorly with bigger data sets as it can consume a lot of time for one iteration.
To overcome such problems gradients are computed for batches of data called as mini-batches. A mini-batch consists of a smaller fixed number of data samples that are drawn from the data set X. The weights(w) update procedure in the stochastic gradient descent is as follows, The gradient used for the weights update is the average of all individual gradients in the mini-batch as shown in the below Equations 2.10 and 2.11:

$$\hat{g} = \frac{1}{m} \sum_{i}^{m} \bigtriangledown_w \mathcal{L}(f(x_i, w)) \tag{2.10}$$

$$w := w - \alpha \hat{g} \tag{2.11}$$

The $\bigtriangledown_w$ is the computed gradient of the loss function $\mathcal{L}$ and $\hat{g}$ is the computed average gradient for a mini-batch of length m. The method is called batch gradient descent. The stochastic gradient descent is the most widely used optimisation algorithm in ML. It is called stochastic because the method uses randomly selected (or shuffled) samples to compute the gradients, hence SGD can be considered as a stochastic approximation of gradient descent optimisation.
Adam has also proved to be a better stochastic optimisation algorithm that consumes very less memory and yields better results. Adam is also well suited when large volumes of data is involved or for large NNs. Adam combines the advantages of AdGrad, which helps in handling the sparse and RMSProp that helps in handling the stationary parts of the function [25].
The weights(w) are recursively updated using the *backpropagation algorithm*. The input data is first propagated through the network first, then the loss is calculated and then back propagated through the network to update weights [18].

Evolutionary Algorithms (EA) also have proved be equally good over the gradient based methods. Population based Genitic Algorithms (GA) can be used to evolve the weights of the deep NN. The GAs can also perform well on the head deep learning problems like Atari games etc [26]. GAs also have proven to work in optimisation of multi-objective function structures [27].
The GA with Elitism is implemented to train the NN implemented in this case.

Genetic Algorithm

This meta-heuristic method basically works like laws of genetics. GA is a search heuristic inspired by Darwin's theory of natural evolution [4]. This algorithm resembles the natural selection process where the fittest individuals are selected for reproduction to make a better future generation. This is a solution when it comes to imperfect, missing input data and limited computational capacity. In this method, first some random solutions (individuals) are generated each containing several properties (chromosomes). Based on the laws of genetics, cross-over and mutations chromosomes produce a second generation of individuals with more diverse properties [28].
Darwin theory is comprised of three basic principles of natural evolution [4]:

- Heredity: A process in which children receive the properties of their parents in form of genes.
- Variation: A process to bring diversity in population.
- Selection: A mechanism which filters out the properties to be carried out in the next generations from their parents. This process explains the survival of the fittest generation.

GAs are also based on these evolution properties. The Figure 2.9 shows the optimisation process for GA, the two fundamental processes of GA are mating and mutation. The GA combines the fittest individuals from previous generation through mating in which off-springs get the properties of their parents in order to survive for the future. The loss function decides the fitness value of the selected parameters of new individuals and iterate over and over until the value of fitness function converges. GA explores all the cost surface with the help of these properties and do not get stuck in local minima like gradient methods. Then it exploits the best feature of the last generation to converge to increasingly better parameters sets.
In order to get the better understanding the main phases of GA are discussed below:

- **Initialisation of population**: In the first step, N random elements are initialised which is referred to as *population*. Each element is a solution to the given problem. An individual is comprised of set of parameters known as *genes*. Genes form a *Chromosome* by joining in a form of string. After initialisation each of the N chromosomes is decoded in order to form the corresponding next individual. There are many procedures to get the variables from the chromosomes, one of them is to divide the chromosome of length m into n equal parts consisting of $k = \frac{m}{n}$ [6]. Each part of the chromosome is a binary number which then can be converted into decimal values. Accuracy of a variable is determined by the number of bits used. The best depiction of population, gene and chromosome is depicted in the Figure 2.10. The binary values are used here in the formulation of population.

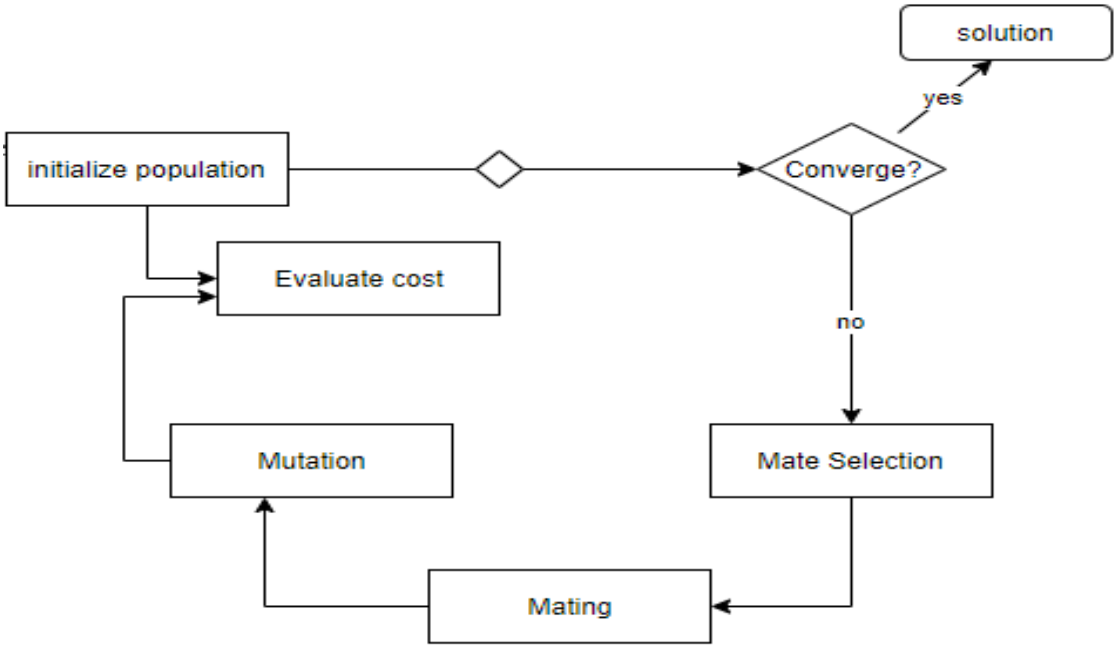


Figure 2.9: Flowchart of optimisation with GA [3]. The steps are explained briefly below the image.

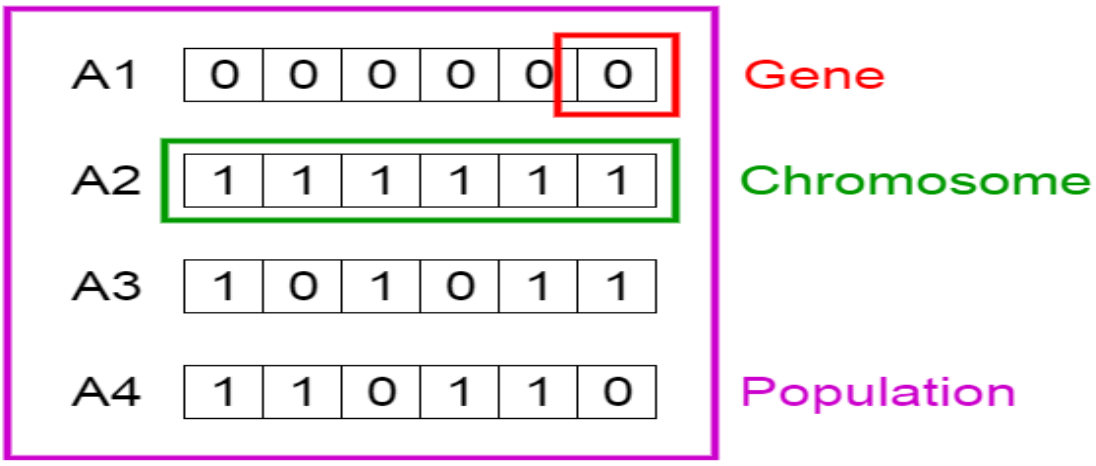


Figure 2.10: Population, Gene and chromosome [4, 5].

For encoding purpose both real and binary numbers can be selected, there is no systematic difference in the performance of GAs [6]. But it is worth noticing that the real numbers carry more information than the binary ones. On the other hand, the number of potential crossover points are reduced when the real numbers are used. In case of real numbers, a generic variable x is formed for genes g as giving a value range of $[-d, d]$ [6] as

$$x = -d + 2dg \tag{2.12}$$

and in case of binary encoding, a generic variable is formed from genes $g_1, ..., g_k$

as giving a value in range of $[-d, d]$ as follows:

$$x = -d + \frac{2d}{1 - 2^{-k}}(2^{-1}g1 + ... + 2^{-k}g_k). \tag{2.13}$$

- **Formulation of fitness function**: This function checks the fitness value for each individual, that make sure the competence of one individual against others. It assigns the fitness score to each individual. The fitness score decides whether the individual would be selected for reproduction or not. The fitness function aims for maximisation/minimisation of fitness scores. The procedure of decoding the chromosomes, evaluation of their respective individuals and assigning the fitness value scores will be repeated until all n individuals have been evaluated in a generation. The next step is to make the next generation with better individuals. And here comes the next step selection of individuals from last generations [6].
- **Selection of Parents**: The selection process determines the fittest chromosomes from the previous generation based upon the fitness scores [4]. Fitness scores can be both, maximisation and minimisation of the objective function. The selection process is stochastic, it means the selection of some individuals do not tend to have direct relationship with fitness scores. The selection of less fit individual can be prior than the best fit individual. The reason is to avoid the local minima, because it could happen that the best fit individual is far from global optimal, while the other less fit individuals may give the global optima after mating with other individual [6]. The two most common methods for selecting the chromosomes for new individuals are **roulette-wheel selection** and **tournament selection** [6]. In roulette-wheel selection process, individuals are selected using a fitness proportional method. This procedure carries out by forming the cumulative relative fitness value ϕ_j, as shown in the Equation 2.14,

  $$\phi_j = \frac{\sum_{i=1}^{j} F_i}{\sum_{i=1}^{N} F_i}, j = 1, ...N \tag{2.14}$$

  where F_i denotes the fitness of individual i. Then a random number $r \in [0, 1]$ is drawn and the selected individual is chosen by satisfying $\phi > r$. After selection new individuals are formed through reproduction. The method that creates new off springs from their selective parents after reproduction is crossover. The roulette wheel is not so feasible from biological point of view. An alternative approach is tournament selection. In this practice two individuals are randomly selected and from population and then selecting the best individuals of pair, i.e. the one with the higher fitness value with probability p_{tour}. p_{tour} is tournament selection parameter [6].
- **Crossover**: This is the most important step in optimisation using GA. New individuals are formed by exchanging the two parents genes. A crossover point cuts the genes of parents at specific point. Both individuals exchange their genes with each other from the crossover points in order to make an offspring. These new individuals inherit their parents properties but also keep

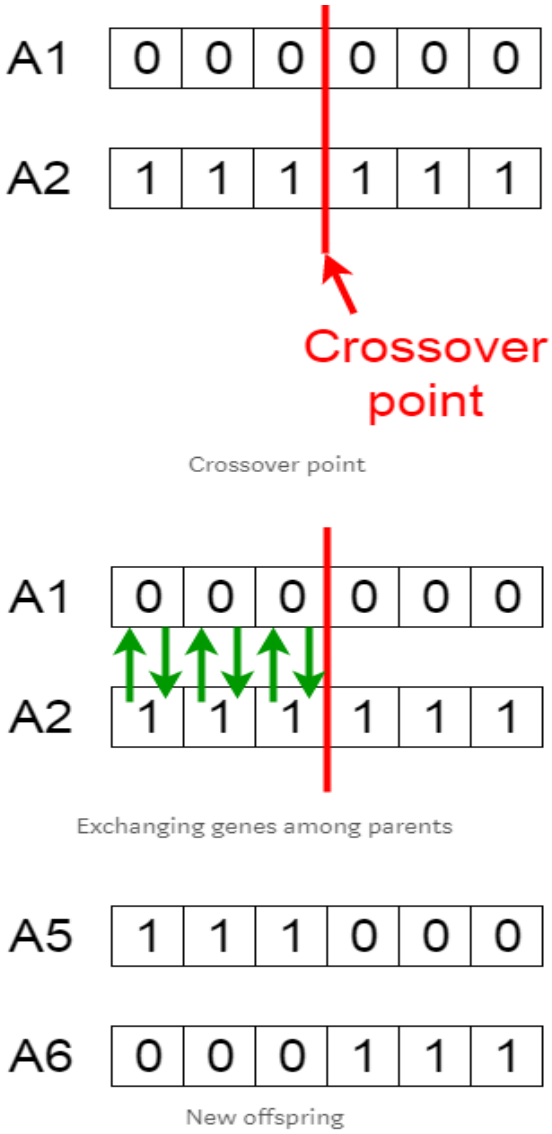


Figure 2.11: Crossover in binary genes [4]

the diversity to some extent. The crossover procedure is shown step by step in the Figure 2.11. At first the crossover point is selected and then both parents switch their genes with each other. Two new off springs adds in population after reproduction process by parents [6].

Crossover can spread a successful individual in population very fast that reduces the diversity, therefore crossover bit too efficient method and it can cause the population to get stuck in the local optima. It is a good practice to carry out the crossover only with a certain probability p_c, called as crossover probability. There are multiple ways to develop the crossover function such as single point crossover, uniform crossover, averaging crossover and length preserving crossover. In the first one a single point is randomly chosen among $m - 1$ possible points in a chromosome length of m. In uniform crossover the number of crossover points are equal to $m - 1$ [6]. Averaging crossover can b

applied to real valued individuals in which a gene g takes the value g_1 and g_2 in the two parents takes the values in the next individuals as in the Equations 2.15 and 2.16 and $\alpha \in [0, 1]$.

$$g_1 \longleftarrow \alpha g1 + (1 - \alpha)g_2 \tag{2.15}$$

$$g_2 \longleftarrow (1 - \alpha)g1 + (1 - \alpha)g_2 \tag{2.16}$$

The length preserving crossover is used when the structure of the process being optimised is known.

- **Mutation**: After crossover, some of off springs will go through mutation with a low predefined random probability p_{mu} [6]. The purpose of the mutation is to maintain diversity in the population and prevent the premature convergence. The mutation procedure is shown in the Figure 2.12.

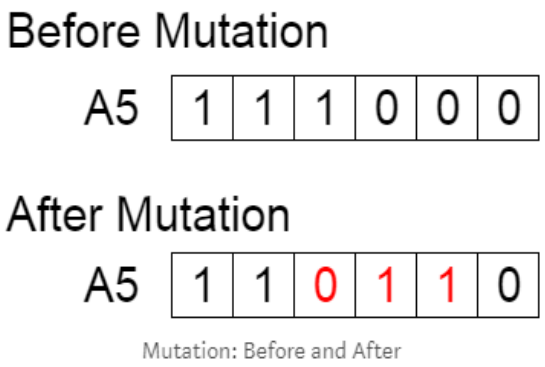


Figure 2.12: Mutation in binary genes [4]

Usually, the p_{mu} is set according to $\frac{c}{m}$ where c is a constant of order 1 and m is the chromosome length. When mutation takes place in real numbers, it changes the property of the individual a lot as compared to the binary ones. Because real numbers contains more information. To avoid such situation **real number creep** is frequently used [6]. In this mutation, the mutated value of a gene is centred of the previous value. The formula for creep mutation is as follows:

$$g' \longleftarrow \psi(g) \tag{2.17}$$

where ψ is a suitable distribution among the individuals, it can be normal or uniform distribution.

- **Replacement**: The final step is replacement. After *selection*, *crossover* and *mutation* there are two generations available, each with n individuals. In general replacement the first generation individuals are discarded and the new n individuals form the second generation are considered. The method continues over all the generations until a reasonable solution is found. Replacement can be characterised by its generational gap, that simply measures the fraction of the population has been replaced in each cycle.

- **Elitism**: For a given generation a best individual will be selected with best fitness value but there is a chance that it would not survive in the next generation in selection process[1]. In order to keep best individual in population for next generation, the best ones can be copied to the next generation without alteration and reproduction [6]. The final individuals would be having a new population formed by asexual reproduction, i.e. selection and mutation only.

Pseudo code of Genetic algorithm
A standard genetic algorithm is shown in the Figure 2.13[2] in order to make the previous theory understandable [6].

[1]More on Optimisation with Elitism and without Elitism is explained in Appendix.
[2]Courtesy of WIT Press from the book [Wahde, M. (Mattias): Biologically inspired optimization methods : An Introduction, 2008,page 56]

1. Initialize the population by randomly generating N binary strings (chromosomes) c_i, $i = 1, \ldots, N$ of length $m = kn$, where k denotes the number of bits per variable.
2. Evaluate the individuals:
   - 2.1. Decode chromosome c_i to form the corresponding variables x_{ij}, $j = 1, \ldots, n$.
   - 2.2. Evaluate the objective function f using the variable values obtained in the previous step, and assign a fitness value $F_i = f(\mathbf{x}_i)$.
   - 2.3. Repeat steps 2.1–2.2 until the entire population has been evaluated.
   - 2.4. Set $i_{\text{best}} \leftarrow 1, F_{\text{best}} \leftarrow F_1$. Then loop through all individuals; if $F_i > F_{\text{best}}$, then $F_{\text{best}} \leftarrow F_i$ and $i_{\text{best}} \leftarrow i$.
3. Form the next generation:
   - 3.1. Make an exact copy of the best chromosome $c_{i_{\text{best}}}$.
   - 3.2. Select two individuals i_1 and i_2 from the evaluated population, using a suitable selection operator.
   - 3.3. Generate two offspring chromosomes by crossing, with probability p_c, the two chromosomes c_{i_1} and c_{i_2} of the two parents. With probability $1 - p_c$, copy the parent chromosomes without modification.
   - 3.4. Mutate the two offspring chromosomes.
   - 3.5. Repeat steps 3.2–3.4 until $N - 1$ additional individuals have been generated. Then replace the N old individuals by the N newly generated individuals.
4. Return to step 2, unless the termination criterion has been reached.

Figure 2.13: A standard genetic algorithm applied to the case of function maximisation [6].

Note F_{best} is the best fitness of the individual, and it refers to global maximum/minimum fitness value i.e. the best value found in any generation so far. The Figure

2.14[3] is the result of a typical GA with elitism. It is seen the fitness value increases rapidly in the beginning and then a slow improvement. A point is important to note that is this figure is to maximise the fitness value function. For minimisation it will be a decreasing graph with negative slope.

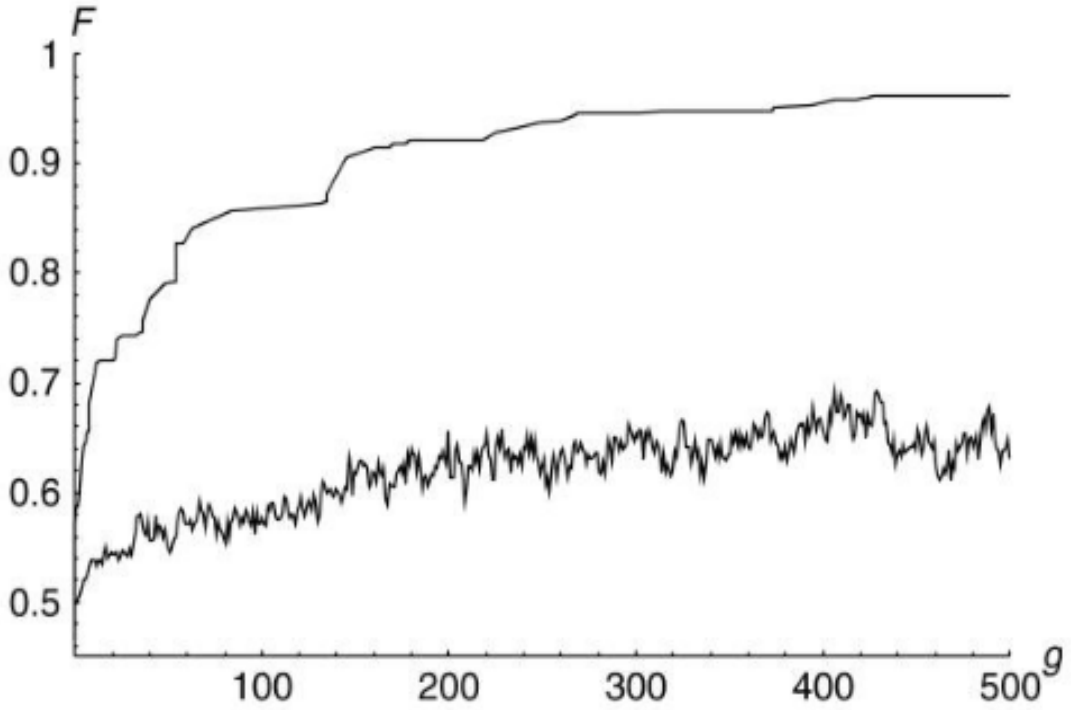


Figure 2.14: A standard genetic algorithm result for maximisation of fitness function over 500 generations, the upper curve shows the best fitness curve the below one is the average fitness curve. The population size is 100 here. F represents the fitness and g represents the generations [6].

2.3.6 Generalisation

A NN that has the ability to generalise makes it a powerful machine. This means that a trained net could classify data from the same class as the learning data that it has never seen before. In real world applications developers normally have only a small part of all possible patterns for the generation of a neural net. Generalisation is a term used to measure the performance of the NN on unseen data. Data sets are often split into 3 different parts, training data (usually 80% of the total data), testing data (usually 10% of the total data) and validation data (usually 10% of the total data). The test sets and validation sets are used to check the performance of the NN on unseen data. Biases in data set leads to bias in predictions and regressions. The first model in the Figure 2.15 fails to generalise the data, this might give a poor performance on unseen data. The right most model is too flexible or non-linear enough to specifically capture the dynamics and classify each and every data

[3]Courtesy of WIT Press from the book [Wahde, M. (Mattias): Biologically inspired optimization methods : An Introduction, 2008,page 57]

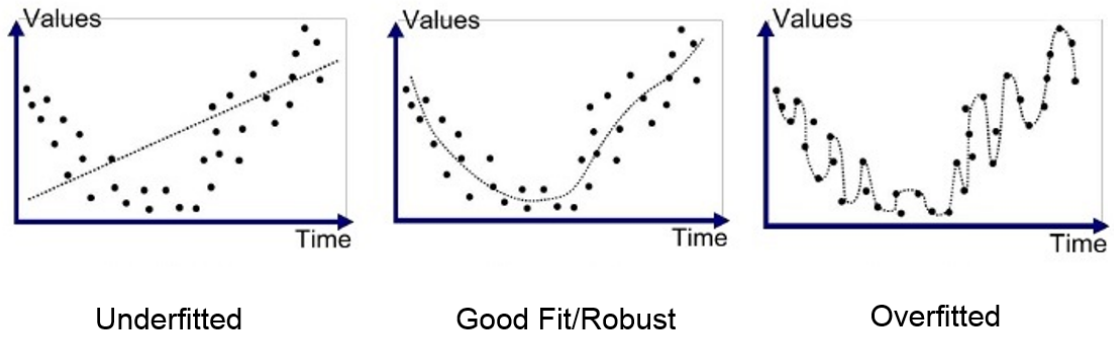


Figure 2.15: The figure shows the underfittind and overfitting of the training data. The middle model best fits and generalises the data into two different classes.

sample. The problem with overfitting is that the model learns to just classify the training data and fails to perform well on the test and validation sets or the unseen data. The model in the middle fits the data very well classifying the two classes of data. The middle model has a better ability to generalise than the other model shown in the Figure 2.15.

There are several means to overcome the potential problem of overfitting by underfitting by using *Regularisation* and *Dropouts*.

2.3.7 Dropouts and Regularisation

Dropouts

The traditional methods of improving the predictions by improving generalisation was to have multiple NN architectures to train on, and then by selecting the model with best performance. Multiple models can also be trained on the same data and the predictions of the average can be considered, such models or networks are known as committees [29]. These methods are relatively expensive and consumes a lot of time. The networks in the committee selected are of different architectures to get the best results.

Dropout is a simple means to avoid committees and deal with the problem of overfitting. The NNs with numerous amount of parameters are often vulnerable to problems like overfitting. Dropouts is a technique of dropping random units and their connections while training the network. This will prevent the neurons from co-adapting to the input data (training data) [30, 31].

The Figure 2.16 shows a fully connected network and a NN after applying dropout with certain probability. After applying dropouts some neurons and their links are randomly dropped during each step of training.

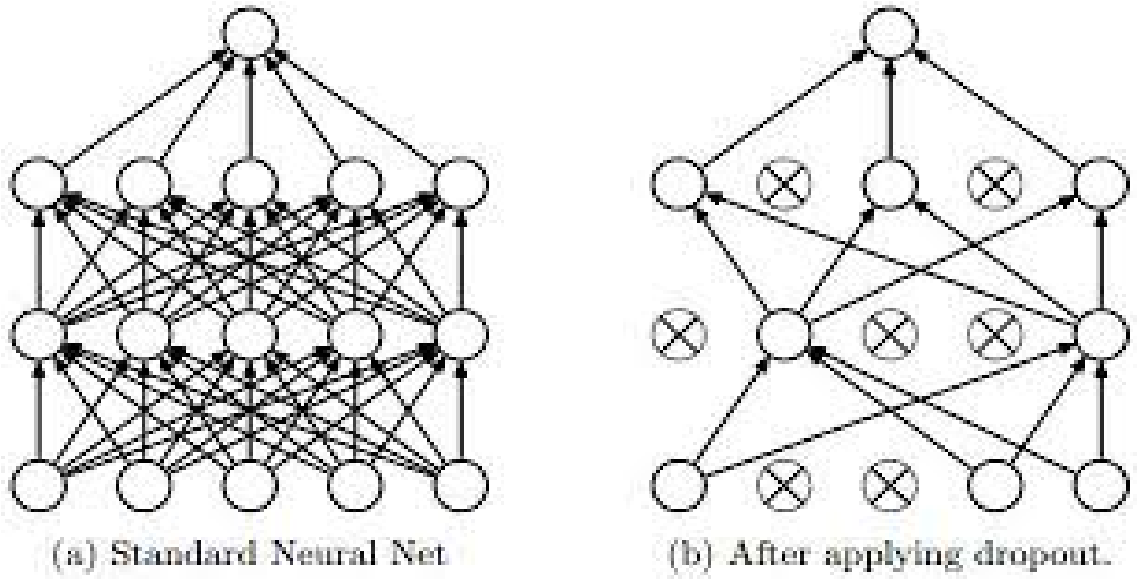


Figure 2.16: The image *a* shows the graph of a traditional feed forward NN and *b* shows the image of the same NN after implementing dropout with certain probability.

Regularisation

A problem in deep learning is how to make an algorithm that will perform well not just on the training data, but also on unseen inputs and data. Many strategies used in ML are designed to reduce the test error, possibly at the expense of increased training error. These strategies are known collectively as *regularisation*. There are many forms of regularisation that are available to the deep learning practitioner. Regularisation can be defined as any modification made to a learning algorithm that is intended to reduce the generalisation error but not its training error [16, 18].

Norm Based Regularisation

Regularisation has been used even before DML. Many regularisation methods are based on limiting the DML models. It is sometimes penalties and constraints are necessary to make an under determined problem determined.

$$\tilde{J}(\theta; x, y) = J(\theta; X, y) + \alpha\Omega(\theta) \tag{2.18}$$

J is the objective function, X,y are the inputs and outputs respectively, θ is the vector of weights(w) and regularisation parameters and $\alpha \in [0, \infty)$ is the hyperparameter that weighs the relative contribution of the norm penalty term Ω, relative to the standard objective function J [16]. To simplify the intuition, the bias term **b** is ignored, this implies that θ is just w.

- **L^2 Regularisation**

  L^2 regularisation norm penalty can be defined as,

$$\Omega(\theta) = \frac{1}{2}\|w\|_2^2 \tag{2.19}$$

The penalty term is also known as *weight decay*, and the regularisation is also called as *Ridge Regression* or *Tikhonov regularisation*. The Equation 2.18 now reforms to:

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y) \tag{2.20}$$

The corresponding parameter gradient is computed as follows,

$$\bigtriangledown_w \tilde{J}(w; X, y) = \alpha w + \bigtriangledown_w J(w; X, y) \tag{2.21}$$

The single gradient update step to update the weights is done as,

$$w \leftarrow w - \epsilon(\alpha w + \bigtriangledown_w J(w; X, y)) \tag{2.22}$$

The Equations from 2.18 to 2.22 shows the implementation of the L_2 regularisation. The addition of weight decay term has modified the learning rule to multiplicatively shrink the weight vector by a constant factor on each step, before performing the usual gradient update step [16].

- **L^1 Regularisation**

  L^1 regularisation on weights(w) can be defined as,

$$\Omega(\theta) = \|w\|_1 = \sum_i \mid w_i \mid \tag{2.23}$$

  The regularised objective function for L^1 is defined as follows:

$$\tilde{J}(w; X, y) = \alpha \|w\|_1 + J(w; X, y) \tag{2.24}$$

  The corresponding gradient is computed as follows,

$$\bigtriangledown_w \tilde{J}(w; X, y) = \alpha sign(w) + \bigtriangledown_w J(w; X, y) \tag{2.25}$$

  where sign(w) is simply the sign of w applied element-wise. The Equations from 2.18, 2.23 till 2.25 shows the implementation of the L_1.
  By inspection the Equation 2.25, the effect of L^1 regularisation is quite different from that of L^2 regularisation. Speaking specifically the regularisation contribution to the gradient no longer scales linearly with each w_i, instead it is a constant factor with a sign equal to sign(w_i). An important observation is that wit is difficult to obtain the clean algebraic solutions to quadratic approximations of J(w;X,y) as obtained in case of L^2 regularisation regularisation [16].

3
Methods

This chapter deals with the explanation and implementation of different methods to answer the research questions asked in Chapter 1 and the theory to support in Chapter 2. The first part of the chapter also explains about the implementation platform and tools used to achieve the results.

3.1 Platform Introduction

The ideally suited *general purpose programming* language **Python 3.6** [32] is used as a platform to develop the DML algorithm and the GA optimiser. The open source libraries like SciPy and Numpy [33] provides an edge to use python for implementing complicated mathematical algorithms. The language has well implemented open source libraries that makes the implementation of complicated algorithms easy and handy. *Jupyter Notebook* is used to code python programs.

The most of the data *CEVT AB* is all available in .mat format. The .mat files are read into pandas dataframes and then processed. The modular open-source NN library keras [23] is used for the implementation of deep NN. The predictions are the analysed and Matplotlib is used to plot the results.

3.2 Bench-Marking Methods

The goal of the thesis is dissolved into multiple simple problems and the complexities are later introduced upon the solved simple problems to reach the final outcome. Deep Learning is wide spread and offers multiple methods and models to tackle the problem. The DML methods tried are explained below. The one with the best efficiency is selected. To start simple, a pure ICE vehicle is considered for energy optimisation. This can later be included into our final optimisation problem as pure ICE is also a subsystem in a PHEV. The key idea is to improve the energy efficiency in the TCU by shifting gear optimally with respect to the efficiency of the engine. The method should deliver an optimal 2-dimensional shift map as a function of engine speed (N) and Engine torque (τ_{ice}). This is the problem to start with and further developed towards the final problem which is explained in the sections below.

3.2.1 Predicting Optimal Gears and Gear Shifts

Supervised Learning is a potential and most reliable means of DML. Supervised Learning is most efficient and yet most uncommon in problems where system dynamics are involved. The optimal gear number is predicted assuming all the data is available. The key ideas to start with supervised learning for gear shifting are:

- The efficiency can be compared against other DML methods.
- The size of the network can be estimated for the final problem by knowing size of the network for this problem.
- This can help to understand the technicalities of the system in a better way.
- The gear shifting also can save some amount of fuel.

Data

In the setup, the current power demand is assumed and is randomly generated data. This data is used and an optimal gear is predicted, that sets the engine speed and engine torque that gives the better efficiency. All the data is randomly assumed or say generated in this setup. The input data that is Power demand (Vehicle Speed, Demand Torque) is denoted as X while the output label that is the output label is denoted as (y). Each combination of an input data has an output label y. The predicted data is denoted as $\hat{y}$. The Vehicle Speed, Demand Torque and Engine Map are used to predict the optimal gear. The algorithm (1) listed in Appendix B shows how to find the gear optimal (y). The engine speed and engine torque are randomly generated uniform distribution of data as shown in the Figure 3.1.
The reason for starting with random data is that it is easily available and covers all the points between upper and lower limits. This helps the network to learn all the possible inputs in the space without any pattern. This can also help in transfer learning by taking the weights of the last layer, if the same predictions are to be expected later.

Network, Loss function and Optimisation

A fully connected feed forward NN is implemented with 4 hidden layers. Different NNs of different size are tried and one with the best performance is selected. For Case 1 all the neurons use relu as the activation function. The output layer is a neuron with relu as activation function that predicts a real number as output. The predicted number is rounded up to get the gear number. In Case 2 all the neurons in the hidden layers are relu and the output layer is a neuron with tanh as the activation function. Tanh is used in the output layer as the range of tanh predictions extends from -1 to +1. The real numbered outputs are rounded to get values of -1, 0 or +1. Dropouts and Regularisation is also implemented to improve the performance of the model. In this case the *l2* regularisation with a penalising factor of 0.1 gave the best performance.

The minimum mean square is the loss and ADAM is the optimiser with a learning rate of 0.01 is used to optimise the loss. The Equation 3.1 represents the loss function

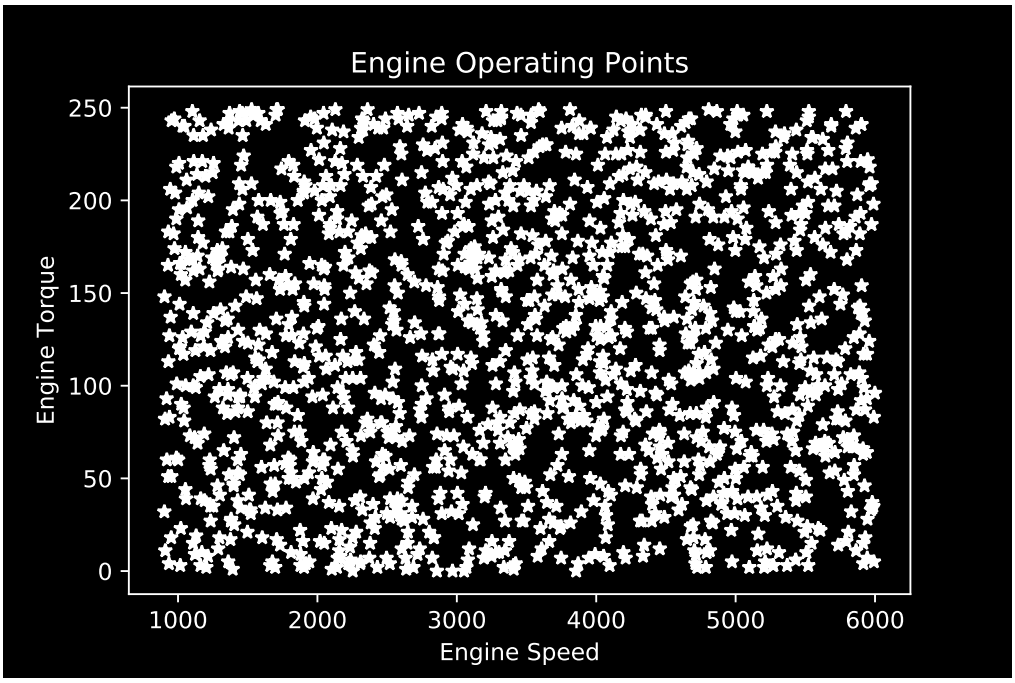


Figure 3.1: The Engine Speed along X axis and Engine Torque along Y axis in this 2D plot. The figure shows the engine operating points covered by the input Power demand to the NN.

that minimises the error in the prediction.

$$\mathcal{L} = \|\hat{y} - y\|^2 \tag{3.1}$$

Case 1: Predicting the Optimal Gear

The first case was to predict the optimal gear. This is the same as creating a 2D look up table or a shift map. The shift maps are generally obtained from test results and are not optimal with respect to fuel consumption in nature. The NN learns to rightly predict the optimal gear that helps the engine to run in a relative sweet spot for the power demand by learning the contours of Engine Map. The Block Diagram 3.2 shows the AI system for TCU.

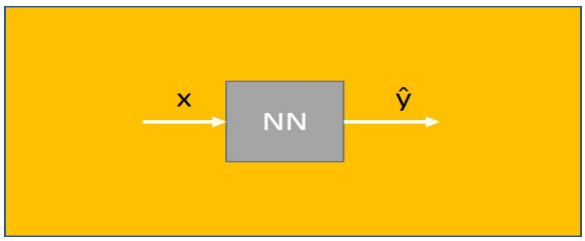
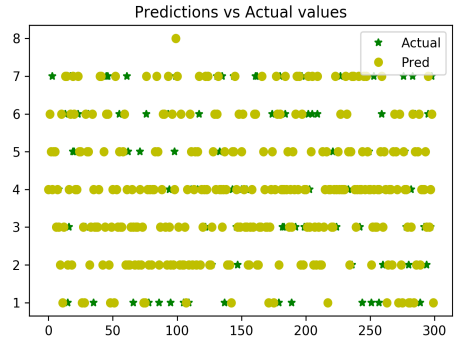


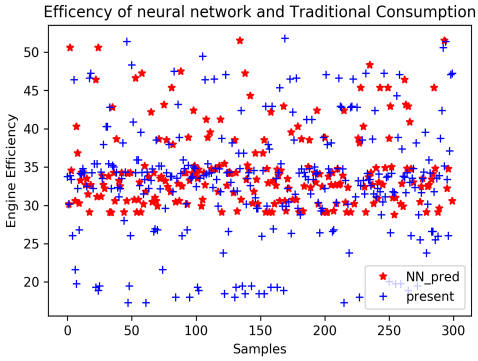
Figure 3.2: NN is like a block box that takes some input X and predict an output value $\hat{y}$.

The Figure 3.3a shows the predictions from the NN and the true output values. The NN is used to learn the optimal gear numbers for the power demands. The Figures

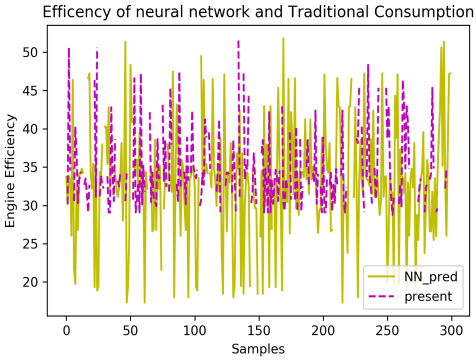
3.3b and 3.3c shows the comparison of engine efficiency points corresponding to the optimal gear and a random gear that can be used at that power demand. The Figure 3.3d shows the error in predictions from the network. The error is calculated as $hist(y - \hat{y})$. The NN has a fair performance considering the amount of the data used.



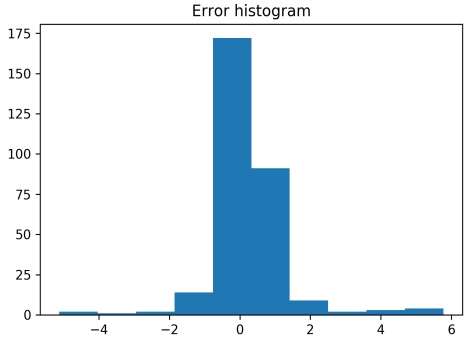
(a) The figure shows the plot of optimal gears and the predicted optimal gear from the network.



(b) The figure shows the comparison of Engine Efficiency for NN predicted gear to the engine efficiency of the inputs.



(c) The figure shows the efficiency of Engine for optimal gear and engine efficiency for other gears.



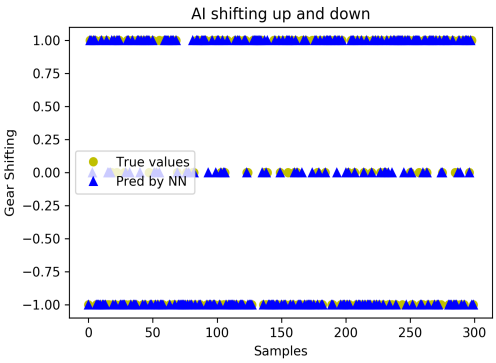
(d) The figures shows the performance of the predictions from the neural network. The error is calculated as $hist(y - \hat{y})$.

Figure 3.3: The figures shows the predictions and the performance from the neural network.

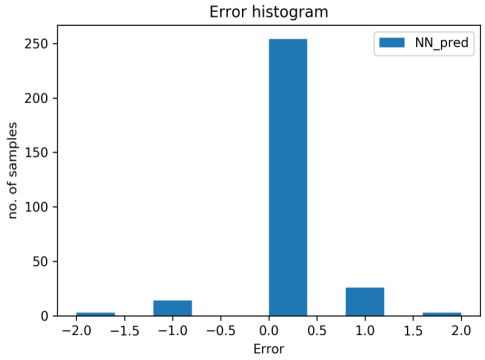
Case 2: Predicting the shift signal to reach the Optimal Gear

The inputs are the same as in *case 1*. The output in this case is the shift signal the shift signal takes the current gear (g_i) in the direction of the optimal. This is considered to be an extension of the first case. The first case predicts the optimal and second case considers the steps to reach the optimal. The first case is not practically feasible as it just says the optimal gear and may even jump the gears in between, which is not true practically speaking. While the second case says weather up shifting, down shifting or holding is better for the efficiency of the engine.

The Figure 3.4a shows the plot of the predicted shift signal versus the actual values. $+1, 0, -1$ says to up shift, hold and down shift respectively. The Figure 3.4b shows the error histogram of the network. The performance of the network is shown in the Figure 3.4. The error is computed as $hist(y - \hat{y})$.



(a) The figure shows the plot of shift signal predictions.



(b) The figure shows the error histogram for the NN. The error is computed as $hist(y - \hat{y})$.

Figure 3.4: The figures shows the performance and the predictions from the neural network. The predictions from the network are plotted against the true values.

However, there are certain problems associated with the implementation of this method even though the results are promising.

- Usually the optimal is unknown.
- This is too simple to resemble a practical TCU as no losses are considered and PHEVs are much more complicated with dual clutch transmission.
- Randomly generated data is easy and was used to just prove the working of the methods.
- The component and system losses are not considered in this methods.
- PHEVs are much more complicated than this.

3.2.2 Predicting the shift signal with GA

This is the same case as *case 2* discussed above with Genetic Algorithm as the optimiser. Here the parameters of the NN are optimised using GA. The procedure is more like exploring the feature space and trying out different options until a convergence criteria is met. The GA does not end up in the local optimal while other gradient descent methods might end up in local optimal. The GA was multi-threaded for faster execution and convergence of the optimisation.

3.3 Energy Optimisation in a PHEV

PHEV is a complicated system with many sub-systems interacting with one another. The approach to energy optimisation follows the same as a traditional control system. Firstly, the components and the sub-systems are modelled, and then are integrated to equations for vehicle dynamics. The controller in this case is the NN that gives input signals to the plant (the vehicle model) and receives vehicle fuel consumption and SoC level after completing the drive cycle as feedback. The NN model and training is addressed in the section *The Neural Network model* after *PHEV Modelling*.

3.3.1 PHEV Modelling

ICE Model

Engine models can be complicated when modelled with a microscopic view and that itself could be an interesting topic for a thesis. The fuel flow factors and its equation is of vital importance, and the other factors are neglected as they are not within the scope of the thesis. The efficiency of an engine or the best working point for an engine is decided by the engine efficiency map that is a function of engine speed and engine torque. The engine map also has the efficiency and fuel consumption contours that says the efficiency and fuel consumption for a given set of engine speed (N) and engine torque (τ_{ice}), shown in the Figure 3.5. The fuel flow rate formula is shown in the Equation 3.2 [34],

$$\dot{m}_f = \frac{N\tau_{ice}}{\eta_{ice}H_{lv}} \tag{3.2}$$

and the fuel consumption is expressed as litres per 100 km and can be calculated as:

$$FC = \frac{\int_0^t \dot{m}_f.10^3.10^2}{x_{total}} \tag{3.3}$$

where x_{total} is total distance travelled. The emissions from the engine are not considered during the optimisation process. The above equation shows that engine speed (N) and engine torque (τ_{ice}) directly affect the fuel consumption of the engine. Therefore by optimising engine speed and engine torque in better points on the engine map a certain amount of fuel can be saved.

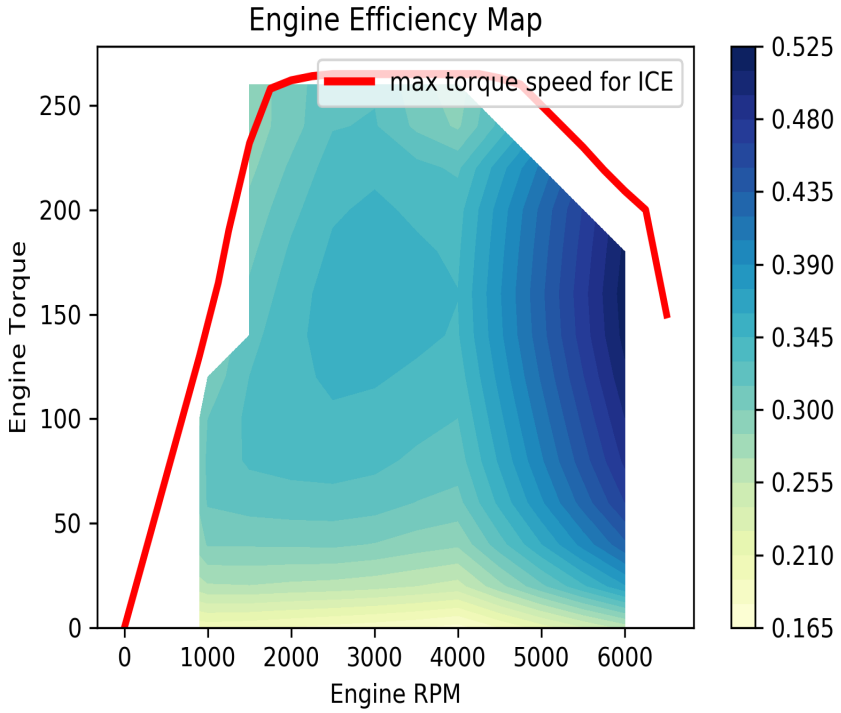


Figure 3.5: Engine Efficiency Map. The colour bar indicate engine efficiencies.

Electric Machine Model

The detailed electric machine model is ignored to reduce complications, and a task of the AI is to run the electric machine or the engine in its optimum points. Hence a specific controller or an equation that governs the motion of the electric machine is unnecessary. The negative torque means that the electric machine is being used as the electric generator. The below zero part of the graph corresponds to the electric machine being used as a electric generator and the above zero corresponds to the electric machine. EM and EG efficiency map is shown in the Figure 3.6. The electric machine map acts the same like engine map, a 2-dimensional look up table. Given the electric machine speed and torque it returns the efficiency of the electric machine at that point [34] as shown in the Figure 3.7. The efficiency is important as to not to loose more electrical energy while discharging and to gain more while charging.

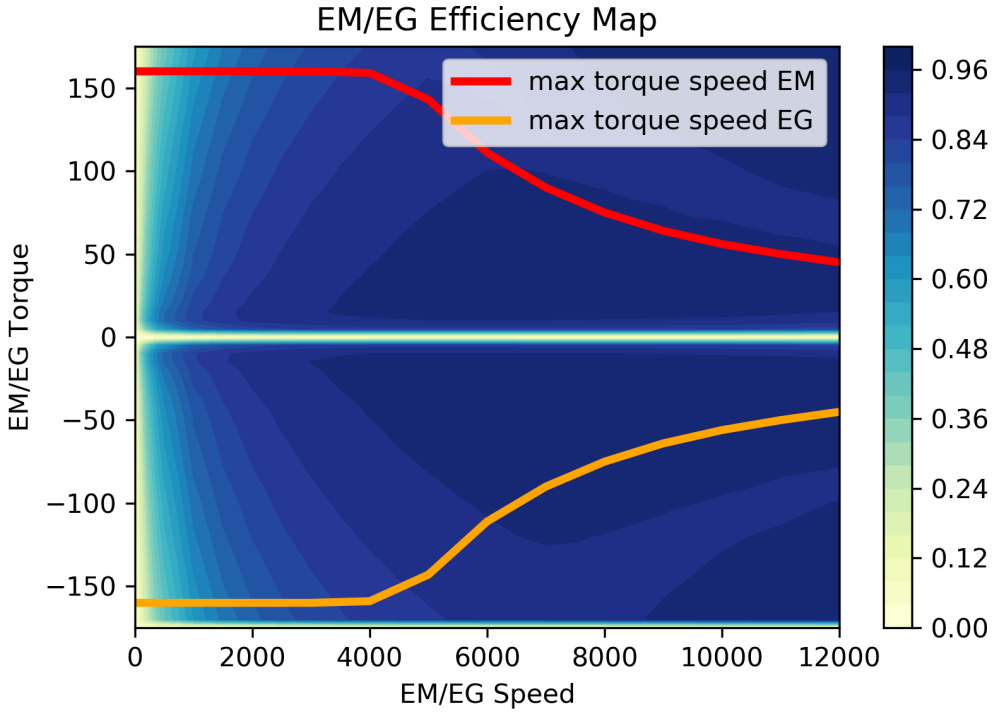


Figure 3.6: Electric Machine Map. The colour bar indicate electric machine/electric generator efficiencies.

Figure 3.7: The figure shows the block diagram of the EM map. The electric machine map is a function of electric machine speed and electric machine torque and returns the efficiency of the corresponding speed and torque.

When the electric machine operates on a demand of positive torque, the electrical energy from the battery is used and while on negative torque demand, charges the battery by converting the mechanical energy to electrical energy, the power consumption of electric machine is given by:

$$P_{em} = \frac{\tau_{em}.\omega_{em}}{\eta_{em}}. \tag{3.4}$$

While the electric machine generates electrical energy, this implies that the power flows from the electric machine to the battery, the generated power can be calculated as:

$$P_{eg} = \tau_{eg}.\omega_{eg}.\eta_{eg} \tag{3.5}$$

The Equations 3.4 and 3.5 are very crucial in calculating SoC level in battery. Battery charge is dependent upon the electric machine operation. If electric machine is used for propulsion then battery charge will be continuously decreasing and if it is acting as the electric generator then the battery charge will be increasing because the energy will be restored in the battery.

Battery Model

A 51 Ah battery is used in the PHEV model development. The energy stored in battery is calculated as :

$$E_{batt} = C_{batt} * \text{no. of battery cells} * 3600 \tag{3.6}$$

The battery state of charge (SoC) varies as the vehicle moves, the battery SoC is a function of open circuit voltage V_{OC} as shown in Equation 3.7:

$$SoC = F(V_{OC}) \tag{3.7}$$

The SoC and V_{OC} relation depends on the temperature, the temperature is assumed to be constant i.e 0 degree Celsius in this case. The V_{OC} is calculated for all the cells in the battery. Battery power demand depends on the operation between electric machine P_{em} and electric generator P_{eg}. The Equations 3.4 and 3.5 describe the EM system in terms of power. The current going out from the battery is described by the equation as:

$$I_{batt} = \frac{P_{em/eg}}{V_{OC}} \tag{3.8}$$

Change in SoC(ΔSoC) is the ratio of present battery energy at time t to the total battery energy. Battery SoC and delta SoC calculations are as follows:

$$\Delta SoC = \frac{P_{em/eg} * t + P_{battloss} * t}{E_{batt}} \tag{3.9}$$

where

$$P_{batteryloss} = \frac{\int_0^{t-1} I^2 dt * R_{int}}{V_{OC}} + Loss_{aux.load} \tag{3.10}$$

SoC is a dimension less unit, current (I) is measured in amperes, $P_{em/eg}$ is in kW, R_{int} in ohms (Ω), battery capacity is in Ampere-hour (Ah), $P_{batteryenergy}$ is in Joules J. And then the current SoC is updated as:

$$SoC_t = SoC_{t-1} - \Delta SoC. \tag{3.11}$$

An auxiliary load of 354 Watts and a temperature resistance loss of 0.17Ω for a cell is considered.

Transmission Model

A 7DCTH system is used in the PHEV model. ICE is connected to both even and odd gear shafts through two clutches C1 and C2, while EM is connected only the to shaft with even gears.
As Explained earlier electric machine can operate only on even gears only and engine can operate on all gears, in order to make them synchronised there is a physical rule that governs the system, *the electric machine gear should always be one up or one down with respect to the engine gear.* The power-train system equations are dependent on the PHEV architecture and configurations. The three kind of equations are *torque*, *angular speed* and *power flow* equations. And these parameters are influenced by gear ratios. The equations are listed below:

- Torque Equations:

$$\tau_{ice} = \frac{\tau_{dem}}{g_{ice}} \tag{3.12}$$

$$\tau_{em} = \frac{\tau_{dem}}{g_{em}} \tag{3.13}$$

  Where $\tau_{ice} < \tau_{ice,max}$ and $\tau_{em} < \tau_{em,max}$. Where τ_{ice} is engine torque, g_{ice} are the engine gear ratios, τ_{dem} is torque demand from the wheels, g_{em} is the electric machine gear ratios and τ_{em} is electric machine torque.
- Angular speed equations:

$$\omega_{ice} = \omega_{wheel} * g_{ice} \tag{3.14}$$

$$\omega_{em} = \omega_{wheel} * g_{em} \tag{3.15}$$

  Where $\omega_{ice} < \omega_{ice,max}$ and $\omega_{em} < \omega_{em,max}$. Where ω_{ice} is the engine speed, g_{ice} is the engine gear ratios, ω_{wheel} is the wheel speed, g_{em} is the electric machine gear ratios and ω_{em} is the electric machine speed.
- Power equations:

$$P_{ice} = \omega_{ice} * \tau_{ice} \tag{3.16}$$

$$P_{em} = \omega_{em} * \tau_{em} \tag{3.17}$$

  Where $P_{ice} < P_{ice,max}$ and $P_{em} < P_{em,max}$. Where P_{ice} is the engine power, P_{em} is the power of the electric machine.

PHEV Operating modes

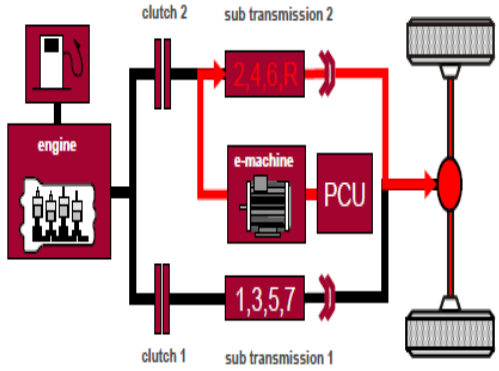
The four different modes that are considered in this work are the pure electric mode, the hybrid mode, pure engine mode and the power mode, the respective modes are explained below. The other operating modes for PHEV are ignored for simplicity.

- **Pure EV Mode:** In pure electric mode, only electric machine is used to propel the vehicle. Both clutches are open, and the ICE is isolated from the drive shaft. Regenerative braking can be used to charge the battery. The energy flow diagram is shown in the Figure 3.8a.

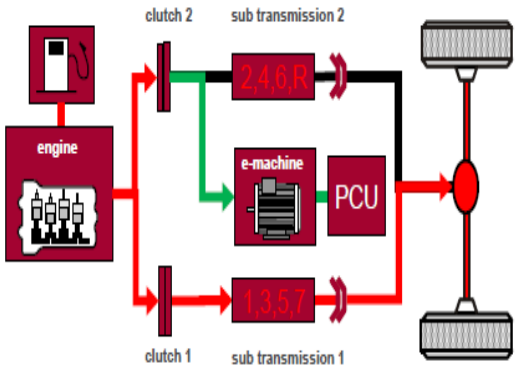
- **Hybrid mode:** In the hybrid mode the battery can be charged through ICE when demanded. In this mode the ICE is connected to all gears through both clutches. When the engine is operating on the odd shaft gear it act as in pure engine mode which means neither regeneration can take place nor engine can charge the battery. When the engine operates on the even shaft, the ICE operates on the best efficiency points, and provides the energy for propulsion and the remaining energy is used to charge the battery. The electric machine acts as the electric generator when the operating gears are even. The main purpose of this mode to consume less fuel by operating in optimum point for both operations, battery charging and vehicle propulsion. The power flow diagram of this mode is shown in the Figure 3.8b.

- **Engine mode:** In this mode only Clutch 1 is open and the battery does not charge from any source. PHEV in this mode behaves as a conventional ICE. *Assumption: An assumption is made that both the clutches are closed and engine can operate on all gears.* The power flow diagram of this mode is shown in the Figure 3.8c.

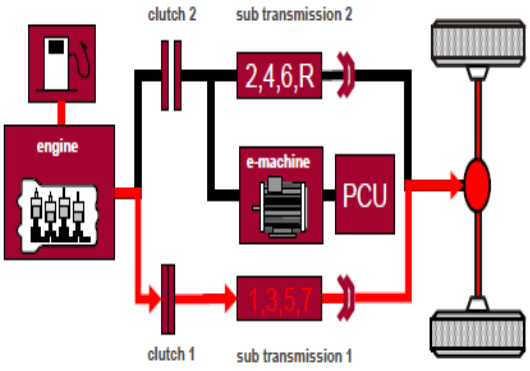
- **Power mode:** In this mode both the engine and the electric machine propel the vehicle in conjuncture. Both the clutches are closed. Battery energy can be restore by regenerative braking. The power flow diagram of this mode is shown in the Figure 3.8d.



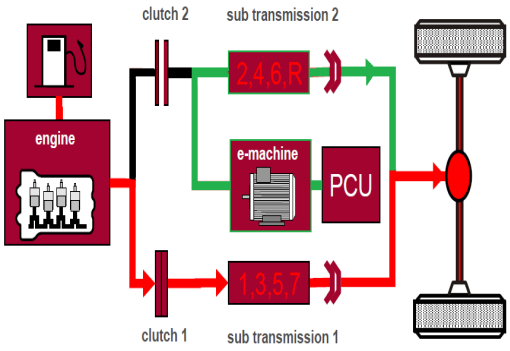
(a) The working of EV mode is depicted in the figure. The red filled lines show the flow of energy. This mode is also referred to as Charge Depletion mode.



(b) The hybrid working mode is shown in the figure. The engine powers the vehicle simultaneously charging the battery, the machine in this mode acts as electric generator.



(c) The engine drives the vehicle. The working of this mode is the same as a conventional vehicle working.



(d) The power mode uses both sources to propel the vehicle. This mode is engaged when the power demand is high for a single power source.

Figure 3.8: The figure shows the four modes considered for the purpose of the thesis. The sub-figures also show the flow of energy from different power sources.

Vehicle Free Body Dynamics

There are many forces acting on the vehicle while driving and a vehicle can move forward if driving force is greater than all of the resistant forces. The vehicle model with all the exerted forces is shown in the Figure 3.9.

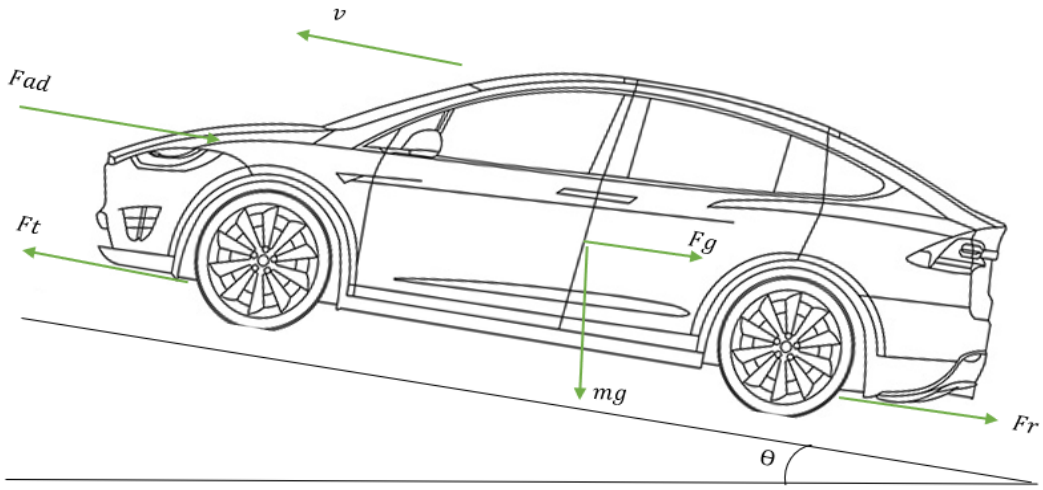


Figure 3.9: Vehicle Model

The important drag forces [34] that can be seen in the above Figure 3.9:

- F_{ad} aerodynamics forces ,
- F_g slope and
- F_R rolling friction.

These forces are shown below in the Equations 3.19, 3.20 and 3.21 respectively. The total force F_t exerted on vehicle can be calculated through the Equation 3.18 [35].

$$F_t = ma + F_{ad} + F_g + F_R \tag{3.18}$$

$$F_{ad} = 0.5\rho C_d A_F v^2 \tag{3.19}$$

$$F_g = mg \sin\theta \tag{3.20}$$

$$F_R = mg \cos\theta . C_R \tag{3.21}$$

Where $m, a, \rho, C_d, A_f, v, \theta$ and C_R are mass of the vehicle, acceleration, air density, air drag coefficient, vehicle front cross-sectional area, velocity, slope and the rolling resistance coefficient, respectively.

Road Load Curve (RLC)

Road-load is the force acting upon vehicle while driving at a speed v. These forces are the summation of tire rolling resistance, drive line losses, and aerodynamic drag. RLC is described as follows:

$$F_t = RLC + ma \tag{3.22}$$

Where RLC is defined as:

$$RLC = 140.15 + 0.4135v + 0.03826v^2 \tag{3.23}$$

RLC is a function of vehicle speed (km/h). And the torque demand (τ_{dem}) and wheel speed (ω_{wheel}) is updated using the equations:

$$\tau_{dem} = F_t r \tag{3.24}$$

$$\omega_{wheel} = \frac{v}{2n\pi r} \tag{3.25}$$

Where r is the wheel radius.

PHEV Configuration and specifications

The electrical specifications of PHEV is listed in the Table 3.1. The mechanical specifications are represented in the Table 3.2. Basic vehicle design parameters are listed in the Table 3.3.

Description	values	Units
Max torque of electric machine	255	Nm
Max Power of electric machine	55	kW
SoC_{init}	[95, 15]	CD mode,CS mode
SoC_{target}	[55, 15]	CD mode,CS mode
SoC_{limit}	[12, 12]	CD mode,CS mode
Battery capacity	51	Amh
Battery energy	17	kWh
EM fix ratio	1.59375	–
No of battery cells	96	–
Nominal Voltage	3.6	$V/cell$

Table 3.1: Electrical specifications.

Description	values	Units
Max Torque of engine	265 @1350-3900 RPM	Nm
Max Power of engine	132@5700 RPM	kW
Q heat of fuel	41	$\frac{MJ}{kg}$
fuel density	0.748	$\frac{kg}{m^3}$
odd shaft gear ratios	[16.041, 6.697, 3.729, 2.492]	–
even shaft gear ratios	[9.65, 4.836, 2.95]	–

Table 3.2: Mechanical specifications.

Description	values	Units
Official Curb Weight	1930	kg
cd	0.36	m
cross section area	2.53	m^2
Radius	0.35814	m

Table 3.3: Vehicle Attributes

Powertrain systems	type
EM	AC Synchronous PM
Fuel	Gasoline 95 RON
Engine	Turbo charged
Gear box	7DCT

Table 3.4: Models specifications.

3.3.2 The Neural Network Model

A feed forward NN model is implemented as the controller for the modelled plant in the thesis. The Table 3.5 shows the architecture of the NN implemented. The NN model is a sequential feed forward network with 3 hidden layers, a input layer and a output layer. The output layer has 3 neurons with tanh as activation function that predicts shift signal for electric machine gear, engine gear and modes respectively. 1 neuron with relu as activation function for predicting the value of α, the power distribution factor between engine and electric machine in hybrid mode. The value of α dictates the power to be generated from the engine in Hybrid mode. Quoting an example say if the predicted value of α is 0.2, then 0.2 kW is generated from ICE, that can be used either for propulsion or for charging the battery depending on the Power demand and battery SoC levels. 1 neuron with expit or sigmoid is used as an activation function, because the value of $\beta \in (0, 1)$. The β is used as the power distribution factor in the power mode that says the percentage of power demand to be delivered by the engine. The Table 3.7 defines the parameters α and β. The NN is non-linear function approximator that predicts the outputs with the given inputs. The NN predicts the shift signal for the electric machine gear, engine gear, modes, α and β. The Table 3.6 shows the type of activation function implemented, and the range of the omitted output prediction.

NormalisedSOC	NormalisedSOCTarget	WheelSpeed	DemandTorque	emGear	engGear
1.25	0.15789473684210525	0.0	0.33462214	0.0	0.0

Figure 3.10: The figure shows the input to the NN. The input data is a numpy array and stored also as a pandas dataframe for visualisation.

Layer(type)	Output Shape	Param#
dense_1	(None,8)	56
dense_2	(None,6)	54
dense_3	(None,5)	35
dense_4	(None,5)	30
Total params : 175 Trainable params : 175		

Table 3.5: The table shows the summary of the NN model implemented.

Activation function	# Neurons	Range	Output
Tanh	3	(-1,1)	$g_{em}, g_{ice}, modes$
Relu	1	(1,∞]	α
Sigmoid	1	(0,1)	β

Table 3.6: The table shows the purpose and type of neurons implemented in the output layer.

Variable	**Definition**
α	Power split factor that decides the power to be generated from the engine in hybrid mode. $\alpha \in (0, \infty]$
β	Power split factor that sets the *ratio* to split the power between EM and engine. Any value of β is the power demand multiplied β to be provided by the engine. $\beta \in (0, 1)$

Table 3.7: The table shows the definition and the value range of power split factors.

3.3.3 Loss Function

The loss function forms the most integral part of the system. The loss function is not a part of the NN. The loss function is used to find the parameters of the NN during training. The code included in appendix B (1) shows the loss function implemented. The loss function is divided into individual losses as *lossDistance, lossFuel* and *lossSoC*. The losses are defined as follows:

$$lossDistance = \frac{1000 - \int_t^{t+1} \dot{x}}{11000} \tag{3.26}$$

$$\tag{3.27}$$

$$lossFuel = \frac{\int_t^{t+1} \dot{m}_f * 100 * 1000}{\int_t^{t+1} \dot{x}} \tag{3.28}$$

$$\tag{3.29}$$

$$lossSoC = argmax(SoC_{target} - SoC[-1], 0) * 1 \tag{3.30}$$

$$\tag{3.31}$$

$$\mathcal{L} = min(lossDistance + lossFuel + lossSoC) \tag{3.32}$$

Note: The summation($\sum$) can also be used instead of integration($\int$), if the intervals are discrete and equally timed or spaced. Summation is used in the thesis for simplicity and faster execution.

- ***lossDistance:*** The most important thing while training is that the NN learns to make such predictions that the complete NEDC is run or simulated. *lossDistance* expression takes care that the entire drive cycle is simulated or the NN is penalised for the unfinished distance if the simulation is incomplete. The Equation 3.26 shows the implementation, where $\dot{x}$ is the speed integrated between the two time steps to get the distance travelled for a complete cycle, that is t=0 till t=1186 seconds. The NEDC cycle is approximated to cover 10,932 meters, hence the distance travelled is normalised by a factor of 11000. The constant 1000 is used to keep all the losses on the same numerical scale. The algorithm tends to get greedy with smaller numbers as they indicate the loss is minimum.
- ***lossFuel:*** The lossFuel function is used to minimise the fuel consumed by the PHEV for a NEDC. This Expression 3.28 trains the NN to operate in engine points with minimum fuel consumption. The cumulative sum of fuel consumed at each time step returns the total fuel consumed at the end of the cycle. The fuel consumption decreases as the number of generations in the GA progresses on. The Equation 3.28 shows the implementation, $\dot{m}_f$ is the fuel consumed at the time t, the integration of $\dot{m}_f$ gives us the cumulative fuel consumed for the cycle. The fuel consumed is multiplied by 100 and then 1000 and then normalised with the total distance covered to convert the fuel flow rate to litres/100 Kms.
- ***lossSoC:*** SoC levels play an important role in PHEVs. Ending up in the desired target SoC level is important in a PHEV, if not the difference electrical energy is charged from the electric grid and can cost some amount of money or it comes at the expense of some fuel energy by consuming additional fuel. The Equation 3.30 shows the implementation, SoC[-1] gives the SoC final SoC when the simulation is complete or stopped. Here the difference in SoC is accounted for and the SoC[-1] is expected to be very close to the SoC_{target} when the model is completely trained.

The Equation 3.32 shows the complete loss function $\mathcal{L}$. The loss is minimised as the training progresses.

3.3.4 Training

The feature space of the NN is $\mathcal{R}^6$, implies that feature space is spread across 6 dimensions. This also means that the input to the NN is an array of size (n,6), where n represents the length of the input data. The input data to the network is vertically stacked vectors of *Normalised SoC, Normalised* SoC_{target}*, Normalised Wheel Speed* ω_{wheel}*, Normalised Demand Torque* τ_{dem}*, initial engine gear and initial electric machine gear.* The Figure 3.10 shows the inputs to the NN. The initial SoC and the target SoC are set before the training is initialised. The NN is trained by simulating a complete NEDC and the demand torque for the NEDC is obtained from the RLC. The NN is trained is using the GA as optimiser, and as said above

the loss function helps to set the parameters of the NN during training. This means that the output predictions are made during the execution of a NEDC and each time the NEDC is run, the NN parameters are consequently updated. The training or optimisation can also be based on certain convergence criterion. In this case the NN was trained for 700 generations with 100 genes population and 2 elite copies[1].

The loss function was developed for optimising the fuel consumption, the attributes of the loss function like convexity, smoothness or others are unknown. Gradients are difficult to be found in these type of loss functions. Hence the gradient free Genetic Algorithm is selected for optimisation or training. The GA also helps with the missing values and is robust with the loss function.

3.3.5 The Optimisation

The GA is used as an optimiser, the stochastic GA algorithm optimises the parameters of the NN by using a suitable variable space for the parameters.

- The objective function is the NN and and the simulation environment for the NEDC cycle.
- The variable space $\in [-1, 1]$. This means that the NN parameters can take any arbitrary values ranging from -1 to 1.
- The weights and biases are randomly initialised from the variable space for every individual in a generation.
- The optimisation is run for fixed number of generations and individuals or it can also be run until a convergence criteria is satisfied.
- The assigned set of weights array is called is an individual. The $\mathcal{L}$ is computed for every individual in the generation. The one with least value of $\mathcal{L}$ is saved as an *elite* copy, so that the best individual that returns the best performance is not lost in the stochastic process. It should also be noted that as the number of elite copies are increased the exploration of the variable space is restricted because the mutation mostly happens around the elite copies.
- The individual with the least loss $\mathcal{L}$ gives the best performance of the NN. That individual returns the best set of weights and biases for the NN seen so far.
- The process is repeated until all the generations are evaluated or a certain convergence criteria is fulfilled.

The Algorithm 2.13 shows the detailed and step-by-step optimisation process of a GA.

The implemented code for the objective function that is to be minimised or optimised is listed in appendix B(2). The weights of the NN are optimised here, by computing individual losses. The function returns the optimised NN parameters array after the optimisation or training is completed.

[1] Training is visualised in Appendix.

3.3.6 Model Validation

The model is now coded and implemented in python. The model is validated by comparing the model inputs and signals with the controllers and plant models used for simulations in the FEoP team at CEVT AB.

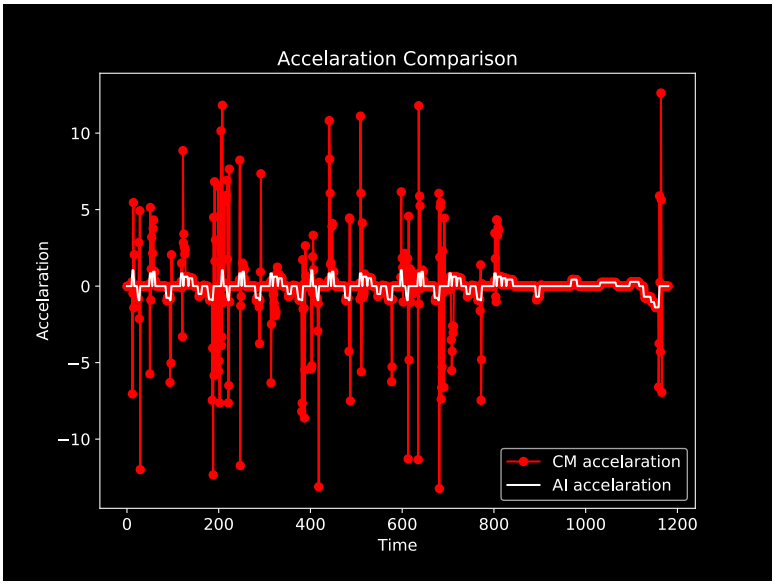


Figure 3.11: The figure shows the acceleration comparison between plant models used at CEVT and the one implemented here.

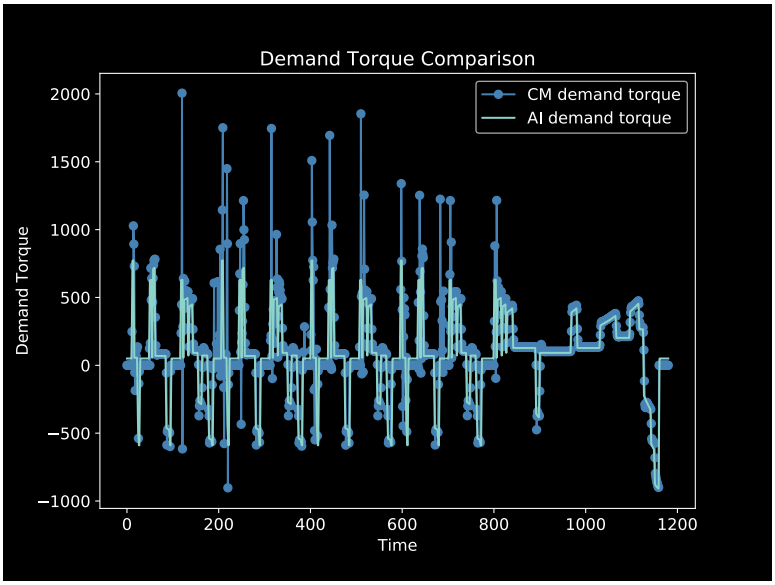


Figure 3.12: The figure shows the demand torque comparison between plant models used at CEVT and the one implemented here.

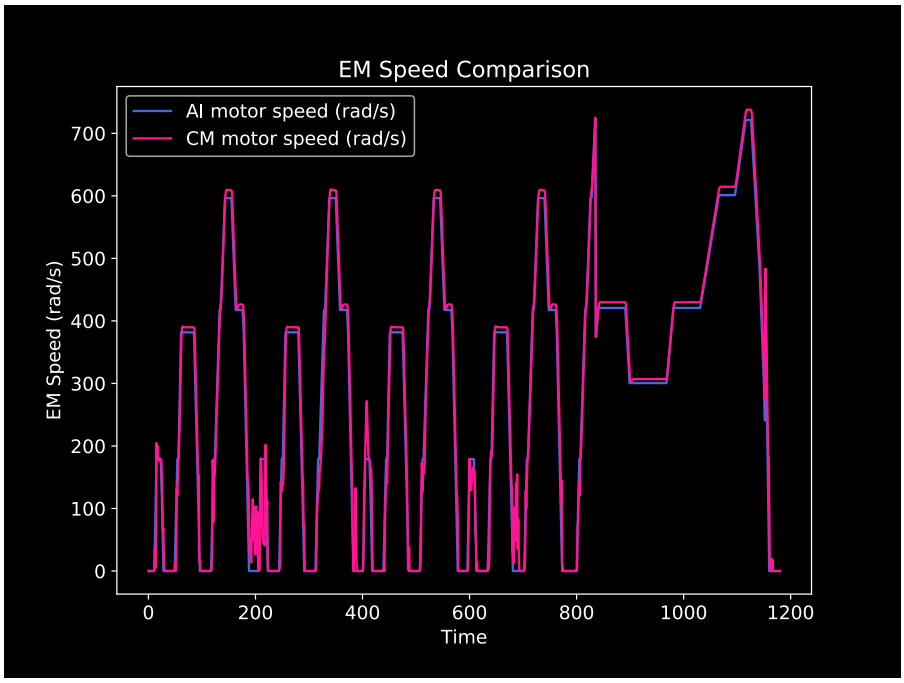


Figure 3.13: The figure shows the electric machine speed comparison between plant models used at CEVT and the one implemented here.

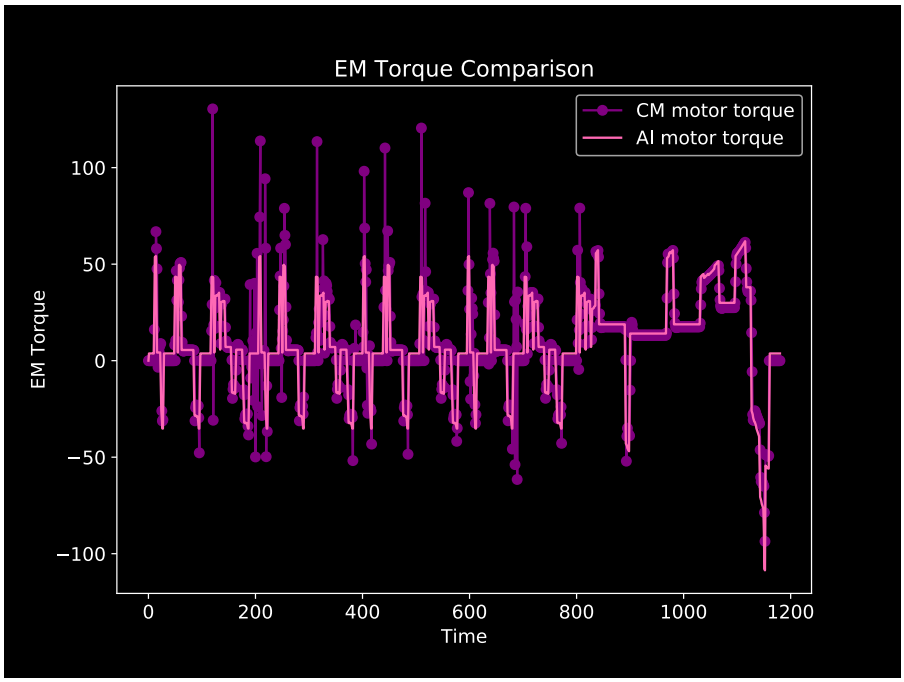


Figure 3.14: The figure shows the electric machine torque comparison between plant models used at CEVT and the one implemented here.

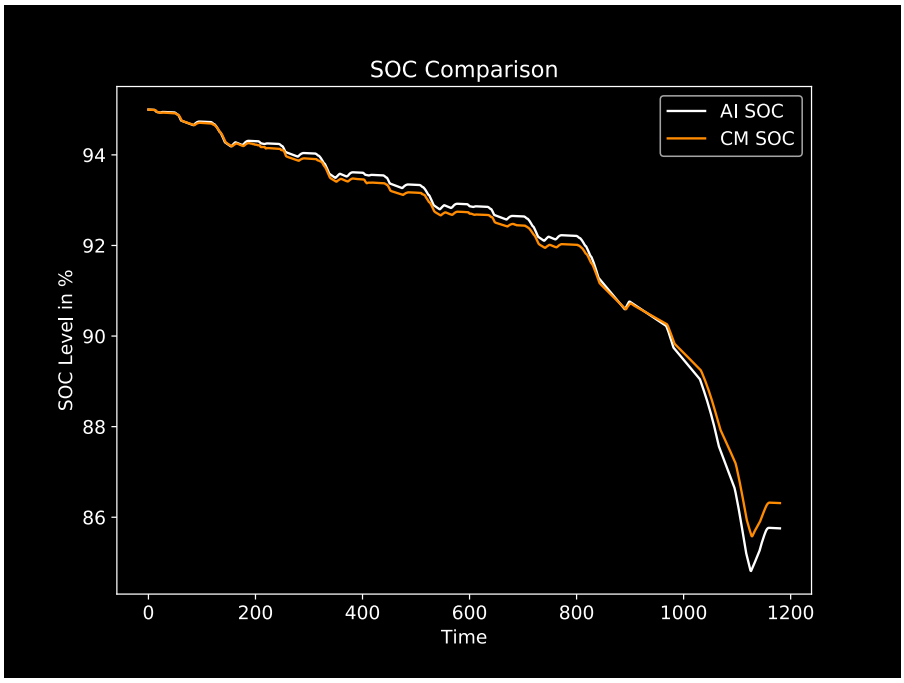


Figure 3.15: The figure shows the SoC comparison between plant models used at CEVT and the one implemented here.

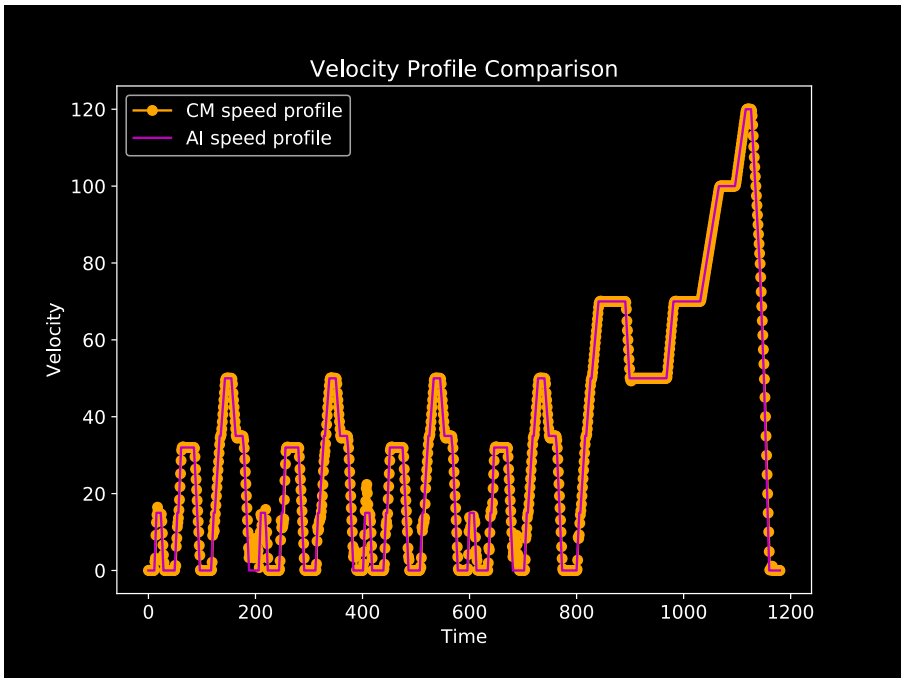


Figure 3.16: The figure shows the velocity profile comparison between plant models used at CEVT and the one implemented here.

Lets assume that **Plant A** is the plant model implemented in the thesis and **Plant B** is the plant model used by the FEoP team at CEVT AB. The plant A is conservative with respect to plant B as the electric range obtained was 6km less and

the electrical power consumption in terms of SoC as seen in the Figure 3.15. The electric range for Plant A is 89.5 kms while the electric range for Plant B is 95 kms. The initial SoC limit considered was from 95%. A heating temperature loss for the battery was assumed to 0.17Ω for one cell and an auxiliary load of 354 watts. The acceleration of Plant B has peaks when compared to the acceleration of Plant A as shown in the Figure 3.11. The peaks are due to the clutch signals while shifting gears. The clutch signals are neglected in Plant A for simplicity. The demand torque in Plant B is due to the consideration of acceleration pedal position, while the demand torque in Plant A is obtained from the road load curve (RLC) as depicted in the Figure 3.12. The small offset in the demand torque of Plant A is due to the RLC factor. The RLC gives a very small yet negligible amount of torque demand even at zero velocity. The electric machine torque and electric machine RPM are decided from the RLC curve in Plant A while the same is decided in Plant B using pedal shift maps, The Figure 3.14 and the Figure 3.13 shows the comparison in electric machine torque and electric machine RPM signals respectively. The plant A model is simulated in Python 3 while the plant B is simulated in IPG CarMaker.

The peaks in the velocity profile of Plant B are due to the Drivers behaviour that is an attribute to be selected in the CarMaker. While the velocity profile for the Plant A is smooth as shown in the Figure 3.16.

The Plant A was tested against the Plant B with same inputs. The idea is to be on the same page as the behaviour of the Plant A is considered with Plant B.

3.4 The AI System: A short summary

The below Figure 3.17 shows the AI system implemented for the fuel economy optimisation in the dissertation. The system acts as a traditional control system with a plant and a controller. The NN is the AI controller here and gives optimised inputs to the plant model. The shift signal updates the current gear $(g_{i_{em}}, g_{i_{ice}})$ for both the Electric Machine and Internal Combustion Engine, the current gear and the SoC is used as the feedback feedback signals to set the Engine Speed, Engine Torque, Electric Machine Speed and the Electric Machine Torque are continuously updated. The plant model on receiving the inputs from the NN runs the simulation for a NEDC.

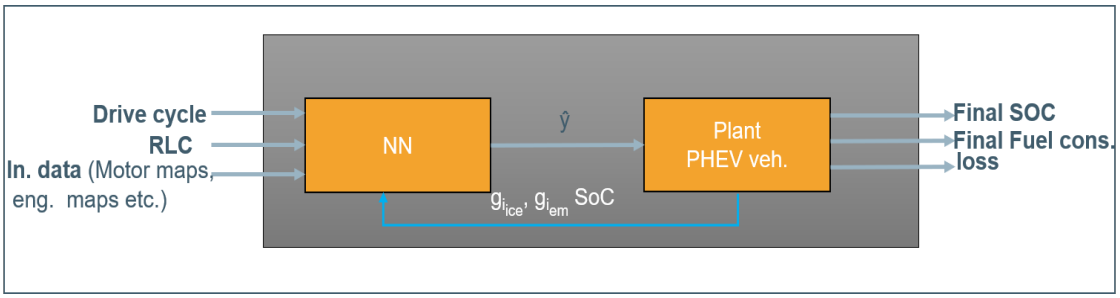


Figure 3.17: The block diagram shows the implemented AI control system for minimising fuel consumption.

The results are discussed in the next chapter while the recommendations and future discussions are talked further on in the paper.

4

Results

In the first half of the chapter the results are visualised and analysed. Later the future recommendations to continue the work is spoken about.

4.1 Results and Inferences

Gear Predictions

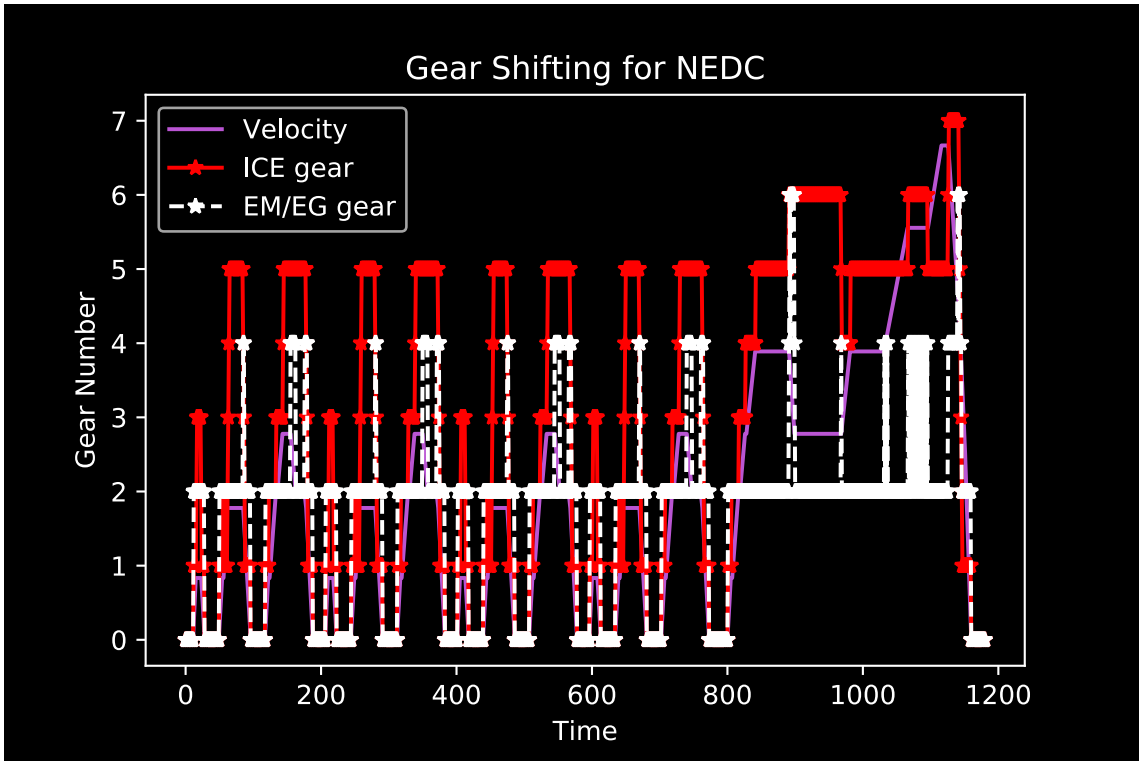


Figure 4.1: The figure shows the gear shift pattern predicted by the NN for both ICE and EM for a NEDC in a charge sustaining mode.

The Figure 4.1 shows the gear shifting pattern for the NEDC produced by the NN. As explained earlier the EM can only operate on even gears while the power from ICE can be transferred using both even and odd gear numbers. The NEDC has repetitive patterns of velocity (**160s-800s on x axis**) in it before there is a larger acceleration at the end of the cycle. The velocity patterns are also reflected in the

gear shift pattern as shown in the Figure 4.1. The predictions from the NN gets kind of messy between 1130s-1160s. The EM gear can take a gear above or below the gear number of the ICE gear, the NN also learns to predict in the similar manner. This is probably due to the fact that the loss function is similar for both gears. This can be avoided by adding a cost each time a gear shift is made.

SoC Neutrality

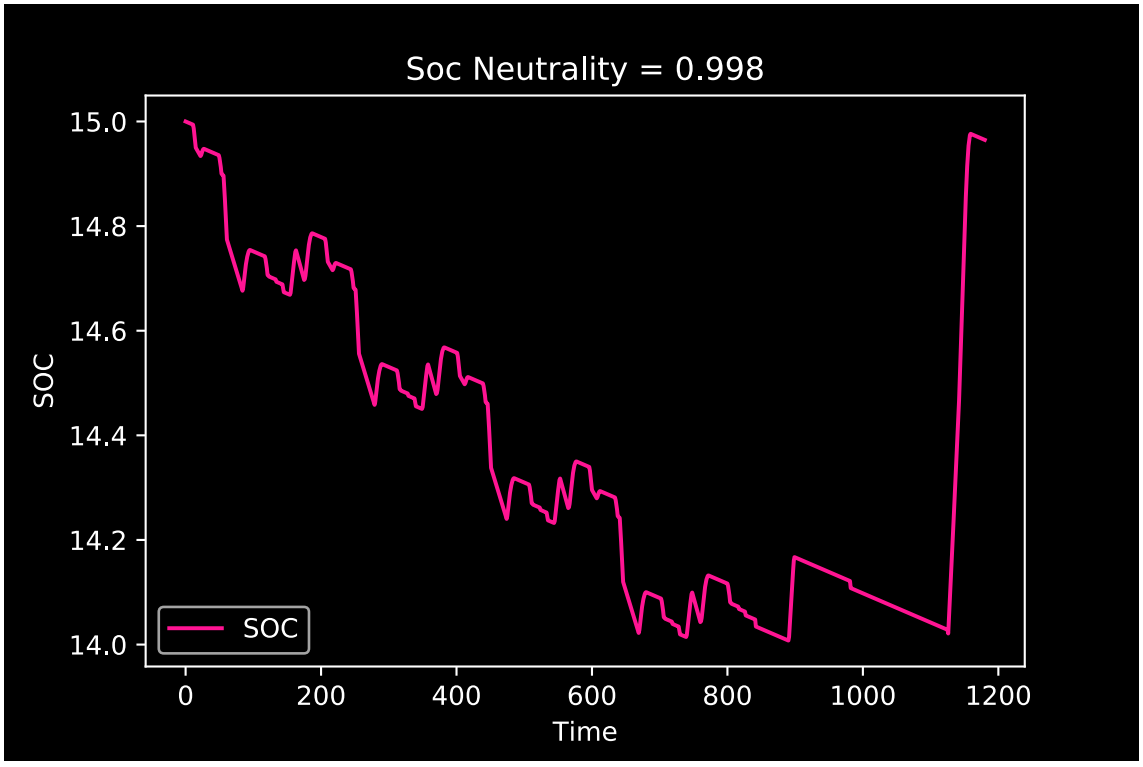


Figure 4.2: The figure shows the plot of SoC level for the NEDC in charge sustaining mode.

The Image 4.2 shows the consumption and recharging of electrical energy for a NEDC. The initial SoC is 15% and the SOoC at the end of the cycle is 14.967%. The SoC neutrality is satisfied with 99.76% accuracy. The NN has learnt to consume and restore the electrical energy to sustain the SoC levels for a NEDC. The plot also follows the similar pattern as a NEDC while loosing SoC while accelerating and gaining SoC while decelerating.

Modes

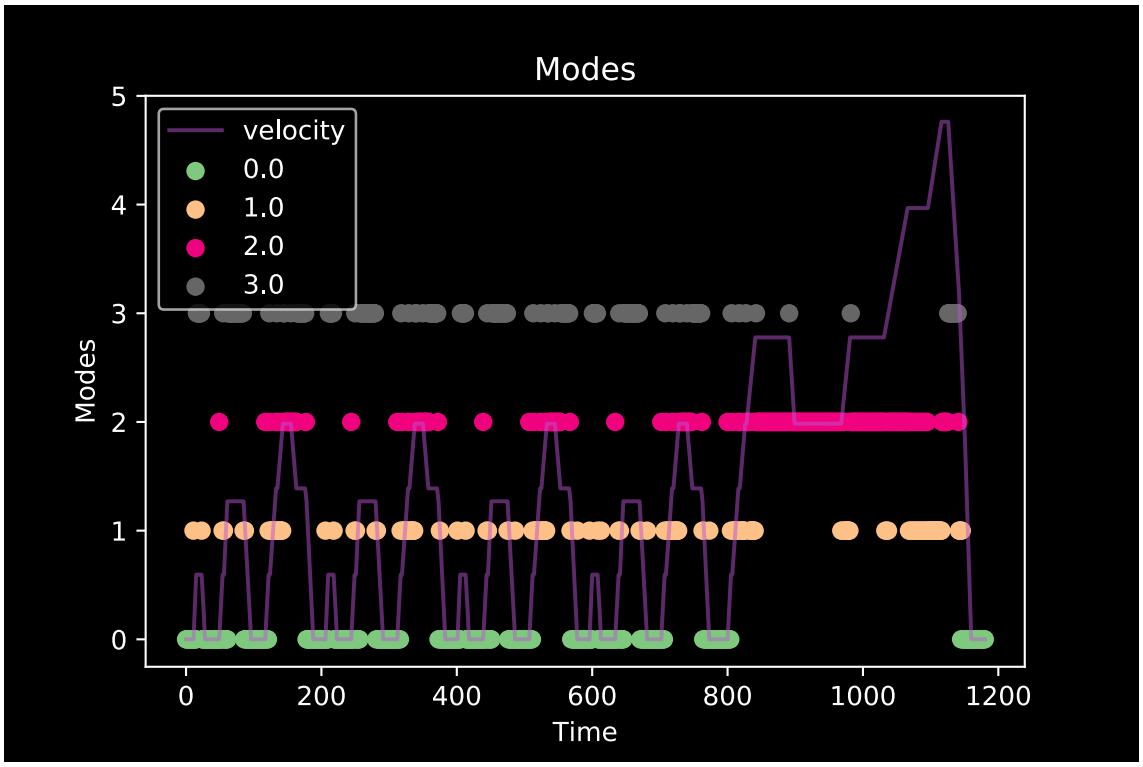


Figure 4.3: The figure shows the modes predicted by the NN for a NEDC.

Predictions	Modes
0	Pure Electric Mode
1	Pure ICE mode
2	Hybrid Mode
3	Power Mode

Table 4.1: The table shows the predicted real numbers corresponding to the actual modes as implemented in the thesis.

The Figure 4.3 shows the optimal modes to be used for the NEDC in charge sustaining mode. The Table 4.1 shows the integer numbers corresponding to the modes implemented in the PHEV. The NN learns that when the power demand is high or say when there is an acceleration demand the power demanded cannot be solely given by electrical energy as the initial SoC is 15% and the minimum is 12%. It also makes sense that the used electrical energy cannot be regenerated to that extent in the very near future. Hence the NN decides to operate the PHEV in power mode, where a very small fraction of power demand is fulfilled by electric energy and the remaining is supplied by the ICE. Thus the NN predicts Power mode (m=3) when there is an acceleration demand from the drive cycle. Sometimes the NN predicts that it might be wise to use hybrid mode to charge the battery during acceleration

or more importantly while cruising as observed between **800s-1100s** in the Figure 4.3.

Power Distribution Factors(α, β)

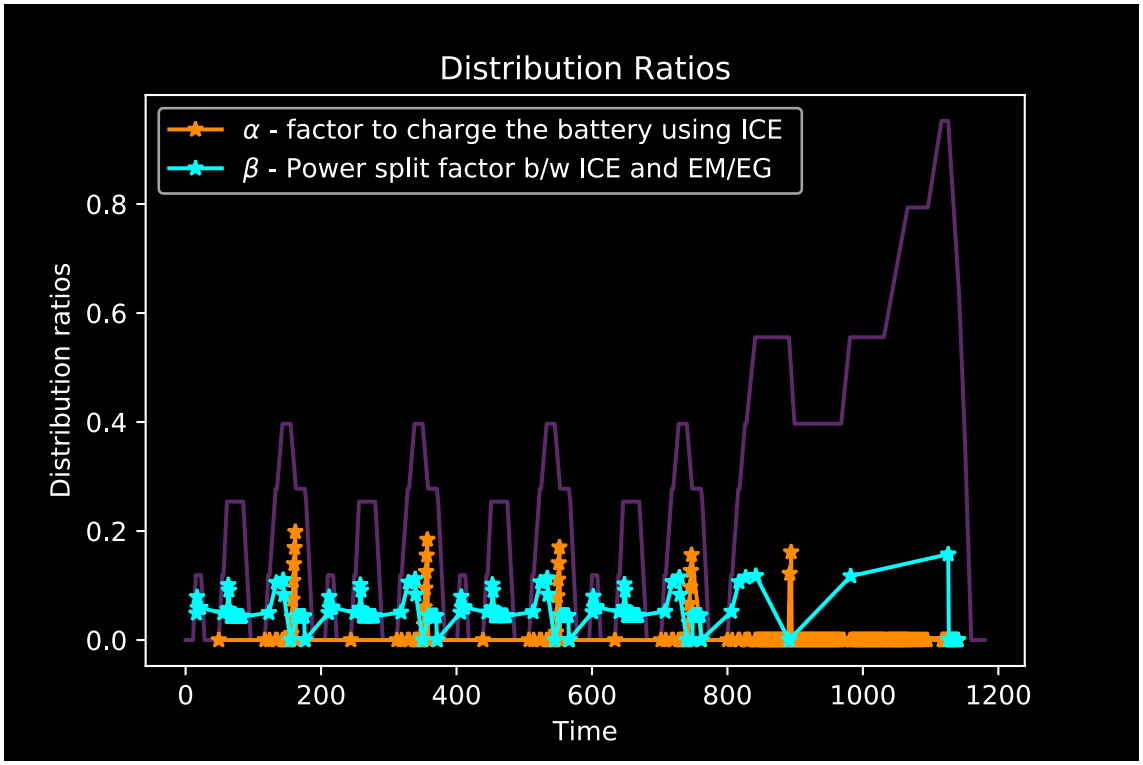


Figure 4.4: The plot shows the usage of power split factors of α, β for charge sustaining mode for a NEDC.

The Figure 4.4 shows the plot of the power distribution factors during the hybrid mode and the power mode. The definitions of power distribution factors are shown in the Table 3.7. The β values stay zero as they only get predictions in power mode. The β value remains zero for all other modes. The β value shoots up when there is an acceleration demand and the battery SoC levels are close to the minimum. The $\beta * P_{dem}$ will the power supplied from the engine and $(1-\beta) * P_{dem}$ will be the power supplied from the EM. While the α is the power decided to be generated from the engine in the hybrid mode to charge the battery. The α predictions occur only when the vehicle is running in the hybrid mode.

Fuel Consumption

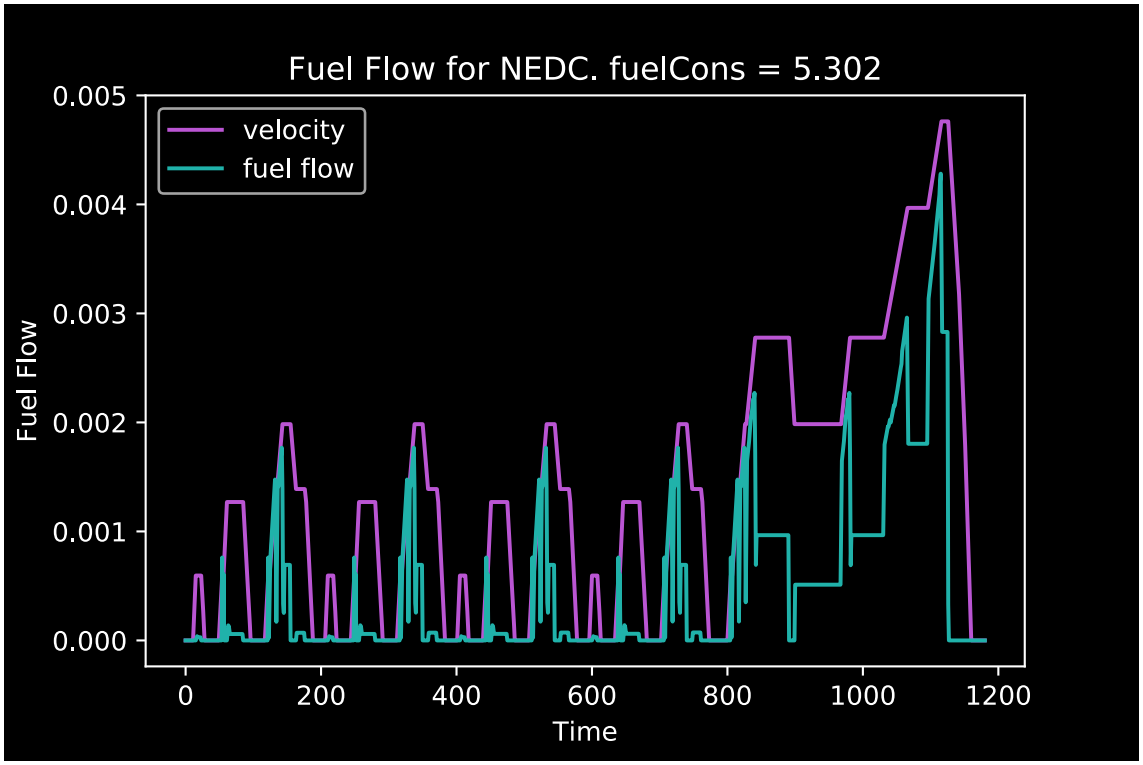


Figure 4.5: The images shows the fuel consumed by the PHEV for a NEDC in Charge Sustaining mode.

The Figure 4.5 shows the fuel consumed by the PHEV for a NEDC in a charge sustaining mode. The plot shows the similarity in the NEDC. When there is an acceleration demand there is a fuel consumed. The bigger the acceleration the bigger spike in the fuel consumed. The fuel consumed at the end of the NEDC for charge sustaining mode is **5.3 liters for every 100 kms**.

4.2 The AI controller vs Traditional Controller

For simplicity in understanding, lets call the implemented *AI controller* as **Controller A** and the *traditional controller* as **Controller B**.

Transmission Control Unit Performance

The Gear Shifting pattern of both the controllers are discussed and visualised here.

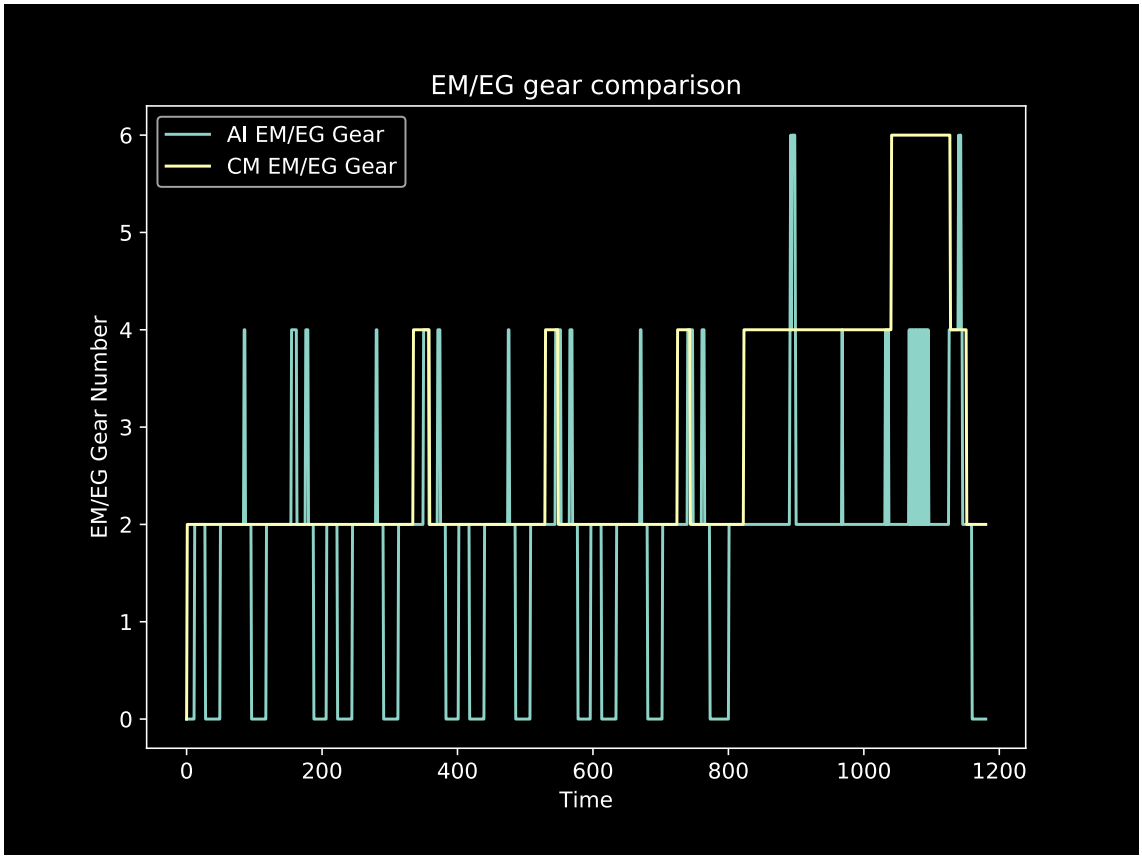


Figure 4.6: The figure shows the EM gear shifting comparison of Controller A and Controller B.

The Figure 4.6 shows the comparison between the predictions of Controller A and the behaviour of Controller B for EM gear shifting. The Controller A learns to come down, and goes to neutral whenever the velocity of the vehicle is zero. This results in a lot of transitions in gear shifting. The added advantage to this is that the Controller A uses lower gears to recharge while decelerating this means that with a larger gear ratio higher the torque and the better the EM efficiency this helps to regenerate better than Controller B. The transitions may also result due to the sudden change in operation modes. While the Controller B is very smooth and goes to top gear only when the acceleration demand is high and remains in the lowest possible gear (second gear) even when the vehicle velocity is zero.

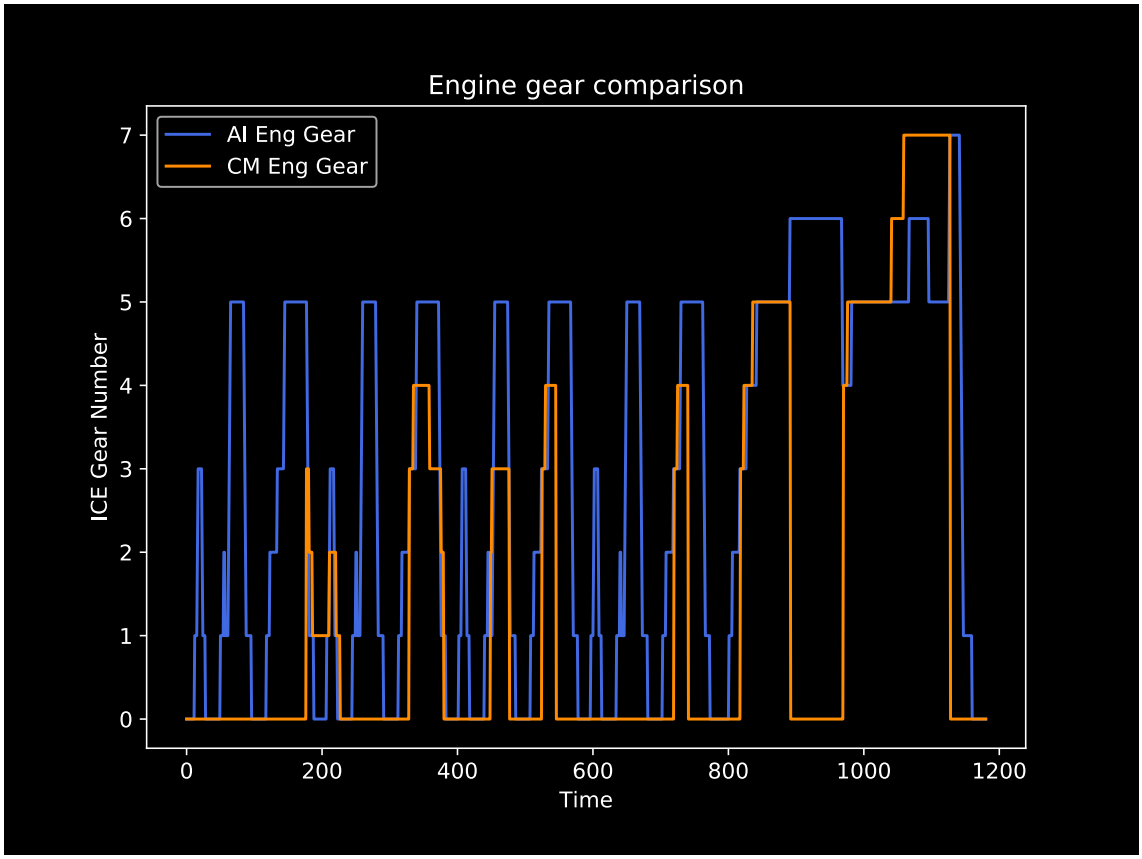


Figure 4.7: The figure shows the behaviour in Gear shifting pattern for engine gears.

The Figure 4.7 shows the comparison between the predictions of Controller A and the behaviour of Controller B for engine gear shifting. The Controller A operates at higher gears than that of Controller B. The reason is because of the lower fuel consumption and better engine points. The pattern between of both the controllers are more or less the same, but the Controller B behaves in a bit conservative manner and could take up higher gears when there is an acceleration demand. The most interesting fact is that Controller A increments step by step considering even the EM gear number into consideration. The behaviour of Controller A resembles the practical situation, every time the vehicle is at rest, the gear shifting happens from neutral or say gear 0 and increments linearly. The gear jumping in Controller B might be mostly due to the usage of different combinations of operating modes.

Electric Machine and Generator Performance

The Figure 4.8 shows the EM and EG operating points on the EM efficiency map for the modes when the EM or generator is being used. The figure shows that the majority of the EM points are in the sweet spot region of the EM where the efficiency of the EM is high. The EM operating points of the Controller A is better than the operating points in the Controller B. The NN has successfully learnt to operate the EM at the higher efficiency points where the electric energy losses are minimal. The regeneration points are fairly better in controller A than controller B.

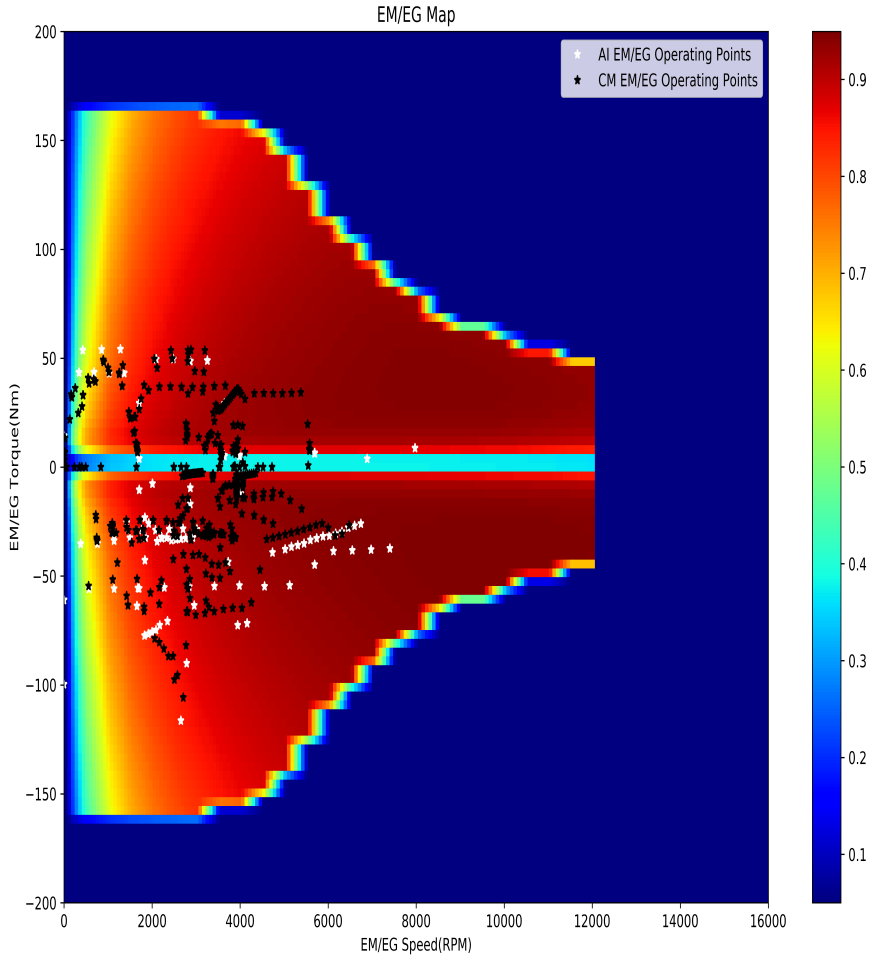


Figure 4.8: The figure shows the performance of the Electric Machine by plotting the operating points on the EM efficiency map.

Engine Performance

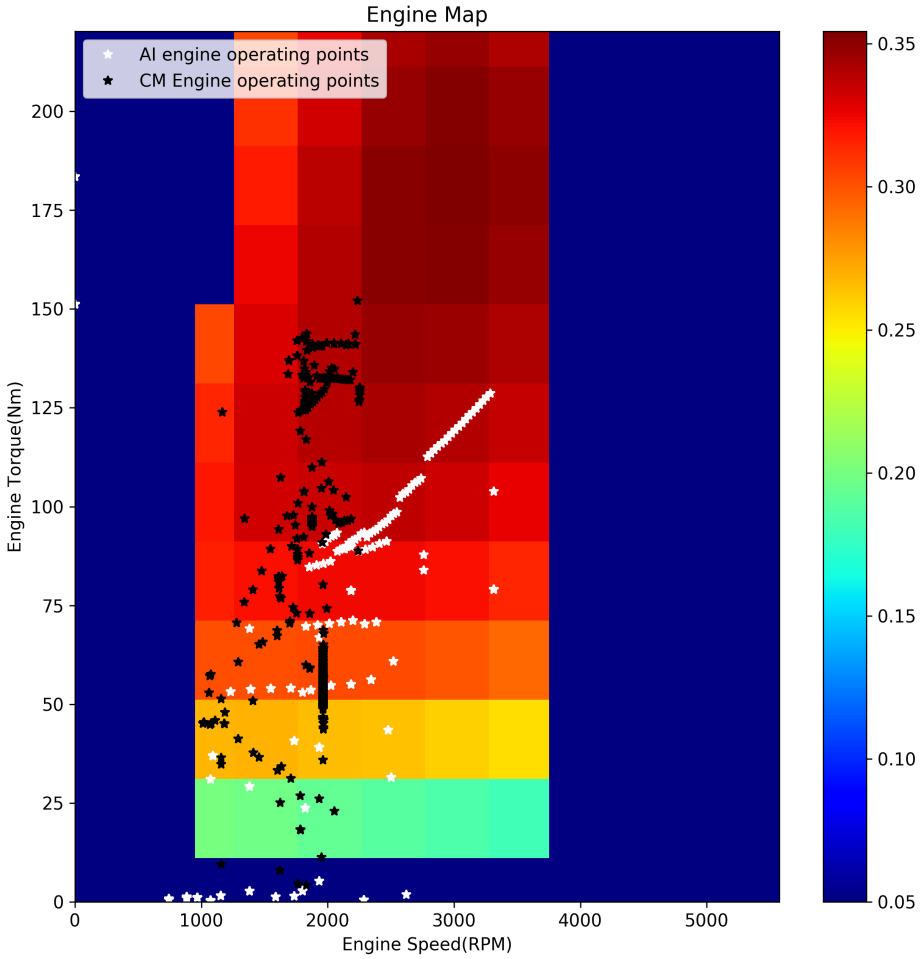


Figure 4.9: The figure shows the performance of the engine by plotting the operating points on the engine map.

The Figure 4.9 shows the engine operating points on the engine efficiency map for the modes when the engine is switched on. The figure shows that the majority of the engine points are in the sweet spot region of the engine where the efficiency of the engine is the highest. The engine operating points of Controller A are better than that of Controller B. The NN has successfully learnt to operate the engine at higher efficiency points where the cost of fuel consumed is minimum.

Fuel Consumption Comparison

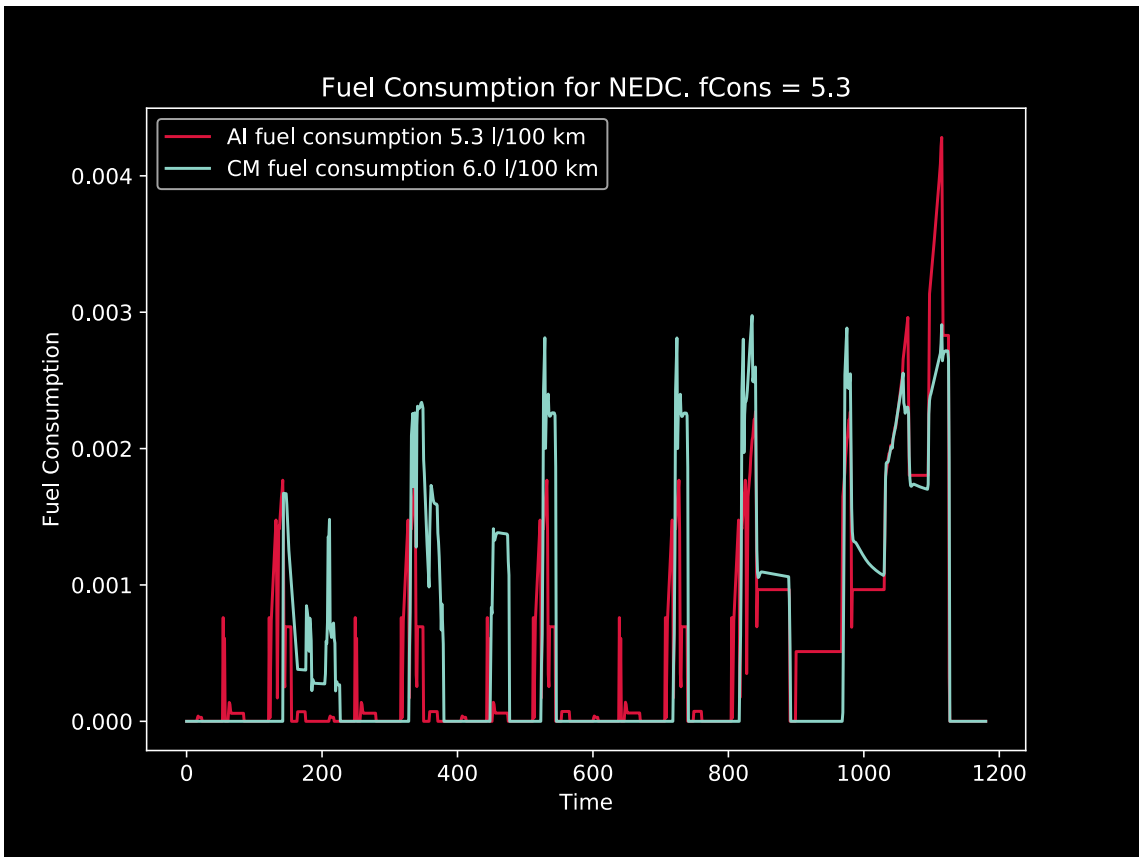


Figure 4.10: The figure shows the fuel consumed in both the controllers.

The Figure 4.10 shows the fuel consumed by the Controller A and Controller B. The fuel consumed by Controller A is 6.9% less than that of Controller B in Charge Sustaining mode for a NEDC. The image shows that the Controller A uses the engine more than that of Controller B, but the Controller A uses the engine wisely by running in points where the fuel consumed is relatively less. The Controller B has accounted for cold start losses and idle engine losses. The optimal fuel consumption for the Controller B is **5.7 l/100km**. Hence the Controller A has 11.6% less fuel consumption than Controller B.

4.3 Future Recommendations and Discussions

The results achieved during the study has been satisfactory. A few future recommendations have been discussed here that can be considered to improve the results further.

Recurrent Neural Networks

RNNs are a super set of feed forward NNs. RNNs can be seen as or visualised as computational graphs. The RNNs are mostly used for text classification, Neural

Machine Translation, Sentiment Analysis etc. The RNNs have the ability to store the previous state of the input and use that information to predict some useful outputs.
The state update equation in an RNN cell is defined as:

$$h_t = f_w(h_{t-1}, x_t) \tag{4.1}$$
$$y_t = W_{hy}h_t \tag{4.2}$$

The f_w can be any non-linear function, h_{t-1} is the previous state and the x_t is the input at the time stamp t. The current state is updated using the previous state of the input and the current input.
The state update with a *tanh* as the non-linear activation function is defined as follows:

$$h_t = tanh(W_{hh}h_{t-1}, W_{xh}x_t) \tag{4.3}$$
$$y_t = W_{hy}h_t \tag{4.4}$$

Where W_{hh} is the weight matrix that updates the internal states of the RNN and the weight matrix W_{xh} is multiplied with the input. The weight matrix W_{hy} extracts this hidden state and transforms it to a meaningful predictions at eact time step t. This is the general working of a RNN. Where the *tanh* function can be replaced by more complicated cells or functions, like LSTM, GRU etc.
The RNNs provide an advantage of storing some valuable pieces or patterns from the inputs that can be used as an advantage to improve the quality of predictions. In this case a simple means could be that RNNs can store the time stamps where there are sudden accelerations and deceleration. RNNs can provide an edge in gear shifting by remembering the previous state or previous predictions before predicting the current value. The information of previous states can be important in predicting the next values. This could be used as an advantage to regenerate in a better way and consume fuel minimally. However RNNs comes up with challenges like training, back-propagation with time, vanishing gradients to name a few.

Different Drive Cycles

The input can be any drive cycles or a combination of drive cycles. This helps the NN to observe patterns and actually learn to simulate to physical resemblances. Different drive cycle has different patterns or they are made up for different driving styles. This can help the NN to learn different driving styles like driving in cities, highways, stop & go etc. This can also help the NN to generalise better and perform better on unseen data.

Test Data from Vehicles

The data can be collected from the sensors or from the control units in the PHEV. This data can be noisy and also can correspond to the actual and practical driving scenarios in the world. This can help the NN to be robust to missing values, noisy and corrupt data.

Multi-Agent Reinforcement Learning

The Multi-Agent Reinforcement learning can be a very good option if computational complexity and resources are neglected. The multi-agent reinforcement learning explores the complete state space by trying different actions and collecting rewards or penalties for the action. The method can take you to the local optimal, while different means could be tried out to reach the global optimal solution. The multi-agent reinforcement learning can be very expensive and could take some time to develop and days to train the model.

Reducing time stamp in Drive Cycle

The timestamp reduction between two points in the drive cycle can help to improve the results. The NEDC used is for 1186s where data is logged at each second. By reducing the time stamp to say about $\frac{1}{10}s$ or $\frac{1}{5}s$ the data logged is more and precise. This can help the NN to focus on the tiniest details and therefore improve the results.

5
Conclusion

To answer the main research questions, the hierarchical control system was understood and an AI controller was implemented for the developed PHEV model. It was a bit tricky to set up the inputs, outputs and the feedback signals. The performance of the AI controller is dependent on the number of inputs used and various other factors. The signals affecting the fuel consumption were studied carefully and classified them into inputs, outputs and feedback signals. Initially different Deep Learning Methods were studied and then were bench-marked and the one the one with best results was implemented. The most integral part of the project, the loss function that optimises the fuel consumption satisfying various physical constraints was developed. The optimal sized NN, the normalised input signals, the output predictions of the NN (input signals to the plant) and the loss function makes the AI controller complete. The system was trained and the results found were satisfactory.

5.1 Key Findings

The section sums up the key results achieved in the thesis:

- Fuel Consumption is minimised.
- A Python platform for vehicle simulation was also developed.
- SoC neutrality is achieved with 99.67% accuracy.
- The NN has learnt to operate the power sources within its mechanical or physical limits meaning the all the predictions made are valid for either of the power sources.
- The NN has successfully learnt to operate both the engine and the electric machine in better points.
- The NN has successfully learnt to simulate a PHEV in a NEDC by selecting appropriate modes and shifting gears.
- GAs can also be a potential means to train the NN.
- Energy Efficiency of the PHEV is increased with a more conservative plant model.
- The NN model was explicitly coded in NumPy to parallelise the training process.

Bibliography

[1] Lino Guzzella and Antonio Sciarretta. *Vehicle propulsion systems : introduction to modeling and optimization*. Springer, 2005.

[2] Lennart Ljung, editor. *System Identification (2Nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[3] Sue Haupt. *Introduction to Genetic Algorithms*, pages 103–125. 01 2009.

[4] Vijini Mallawaarachchi. Introduction to genetic algorithms-including example code. `https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3`, 07 2017.

[5] Shubham Jain. Introduction to genetic algorithm their application in data science. `https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/`, 07 2017.

[6] M. (Mattias) Wahde. *Biologically inspired optimization methods : an introduction*. Southampton, UK ; Boston, MA : WIT Press, 2008. Formerly CIP.

[7] Viktor Larsson Bo Egardt Mitra Pourabdollah, Lars Johannesson. Phev energy management: a comparison of two levels of trip information. Technical report, Chalmers University of Technology, Gothenburg, Sweden.

[8] Yujia Zhai. Advanced neural network based control for automotive engines. Technical report, Liverpool John Moores University, Liverpool, United Kingdom, 2009.

[9] Md. Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Mahmudul Hasan, Brian C. Van Esesn, Abdul A. S. Awwal, and Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR*, 1803.01164, 2018.

[10] David Cid Fernandez. Model Building and Energy Efficient Control of a Series-Parallel Plug-in Hybrid Electric Vehicle. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2016.

[11] Muddassar Zahid Piracha. Improving gear shift quality in a phev dct with integrated pmsm. Technical report, Chalmers University of Technology, Gothenburg, Sweden., 2017.

[12] Karel J. Keesman. *System Identification*. Springer, 2011.

[13] Christian Winge Vigild. The internal combustion engine modelling, estimation and control issues. Technical report, TECHNICAL UNIVERSITY OF DENMARK, LYNGBY, DENMARK, 2001.

[14] Yang-Yu Liu and Albert-László Barabási. Control principles of complex systems. *Rev. Mod. Phys.*, 88:035006, Sep 2016.

[15] Gábor Horváth. *Neural Networks in System Identification*. Budapest University of Technology and Economics, 2002.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[17] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain [j]. *Psychol. Review*, 65:386 – 408, 12 1958.

[18] Lennart Svensson. Lecture notes in deep machine learning (ssy340). Signal Processing Group, Chalmers University of Technology, Sweden, August 2018.

[19] Comparison of Activation Functions for Deep Neural Networks. https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a. Accessed: 2019-05-30.

[20] F. Ahmad, N. A. M. Isa, M. K. Osman, and Z. Hussain. Performance comparison of gradient descent and genetic algorithm based artificial neural networks training. In *2010 10th International Conference on Intelligent Systems Design and Applications*, pages 604–609, Nov 2010.

[21] Understanding Activation Functions in Neural Networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. Accessed: 2019-05-30.

[22] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv e-prints*, page 1903.06733, Mar 2019.

[23] François Chollet et al. Keras. https://keras.io, 2015.

[24] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page 1412.6980, Dec 2014.

[26] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, 1712.06567, 2017.

[27] G. R. Vossoughi and S. Rezazadeh. Optimization of the calibration for an internal combustion engine management system using multi-objective genetic algorithms. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1254–1261 Vol. 2, Sep. 2005.

[28] Manish Kumar. Understanding genetic algorithms in the artificial intelligence spectrum. https://medium.com/analytics-vidhya/

understanding-genetic-algorithms-in-the-artificial-intelligence-spectrum-7021b7
09 2018.

[29] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *2011 International Conference on Document Analysis and Recognition*, pages 1135–1139, Sep. 2011.

[30] Nitish Srivastava. Improving Neural Networks with Dropout. Master's thesis, University of Toronto, Toronto, Cannada, 2013.

[31] Jimmy Lei Ba. Adaptive dropout for training deep neural networks. Master's thesis, University of Toronto, Toronto, Cannada, 2014.

[32] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[33] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[34] Yanhe Li. Optimization of phev power split gear ratio to minimize fuel consumption and operation cost. *Electronic Theses and Dissertations. 4750.*, page 225, 12 2013.

[35] X. Lin, P. Bogdan, N. Chang, and M. Pedram. Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 627–634, Nov 2015.

A

Definitions

A.1 Definitions

A.1.1 Mathematical Model

The Mathematical models are usually classified between a band of colours. *White Box* models being completely descriptive to *Black Box* being a model which is assumed or randomly chosen.
White Box Models - The mathematical models developed based on the first principle or the Newton's equations of motion are called as White Box Models. These type of models are called as White Box models because of the access to the control parameters. When both the structure and the parameters of the model are completely known – complete physical knowledge is available - it is a white box model. White box models can be constructed from the prior information without the need of any observations[15].

Black Box Models - The model development is dependent on the input-output data. The input-output model if often known to as empirical model or Black Box model. The black box model does not replicate the structure of the physical system. The model structure is usually flexible enough to represent a large class of systems. The Black Box Models and White Box models usually represent the two extremes of modelling the system[15].

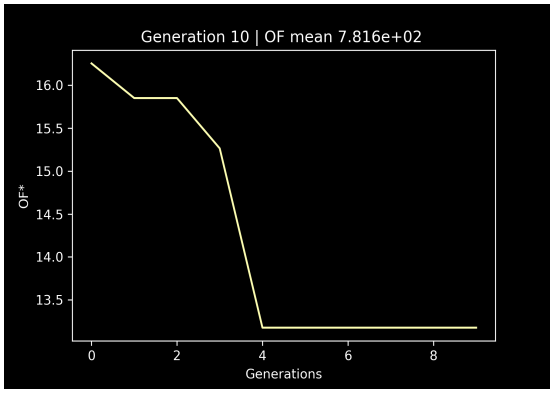
A.2 Stochastic Optimisation with GA

The section explains and visualises the training of NN with GA's. The section also highlights the difference between the usage of **Elitism** and without the usage of **Elitism**.

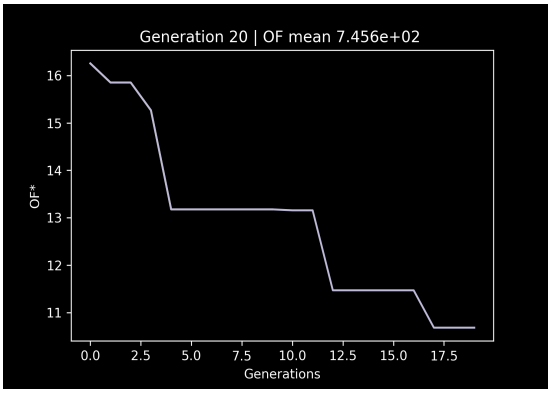
A.2.1 GA with *Elitism*

The below Figure A.1 visualises the training process of the NN or say the optimisation of the loss function. The *Elitism* helps the objective function to be *strictly increasing* or *strictly decreasing* during the optimisation process. As the Figures A.1a to A.1f shows the strictly decreasing optimisation progress. The x-axis shows the generation as the optimisation progresses and the y-axis says the value of the optimisation function. The mean indicated in the plot title represents the number of

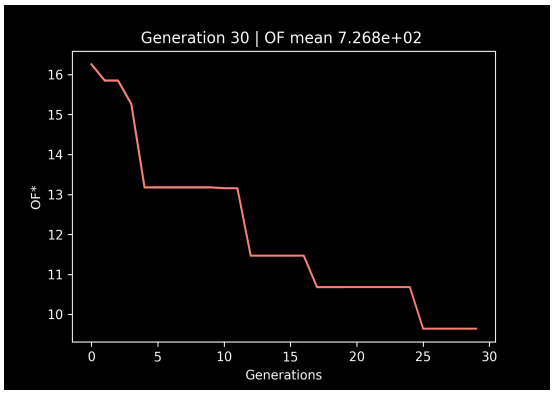
individuals survived during the optimisation process. More specifically speaking, *the mean indicates the set of NN parameters that completed the vehicle simulation for a NEDC. For example in the Figure A.1a there are 781 individuals with respective NN parameter arrays that completes the simulation. The individual with the least loss is the Elite Copy.*



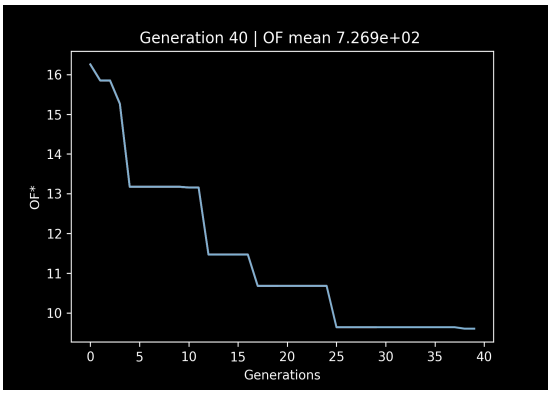
(a) The objective function optimisation for 10 generations with 1 elite copy.



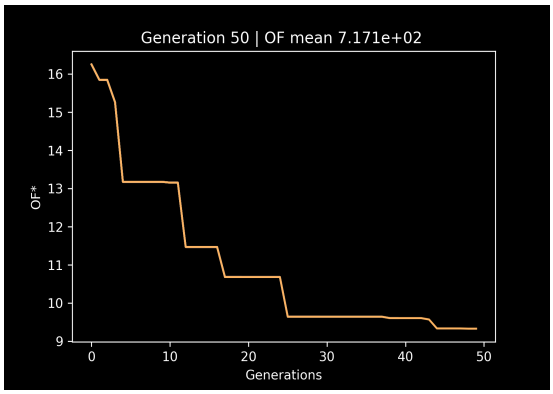
(b) The objective function optimisation for 20 generations with 1 elite copy.



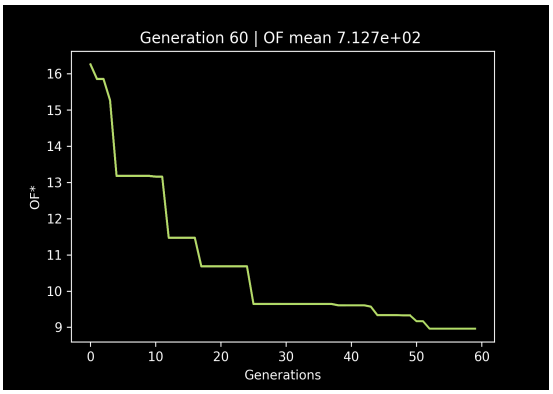
(c) The objective function optimisation for 30 generations with 1 elite copy.



(d) The objective function optimisation for 40 generations with 1 elite copy.



(e) The objective function optimisation for 50 generations with 1 elite copy.

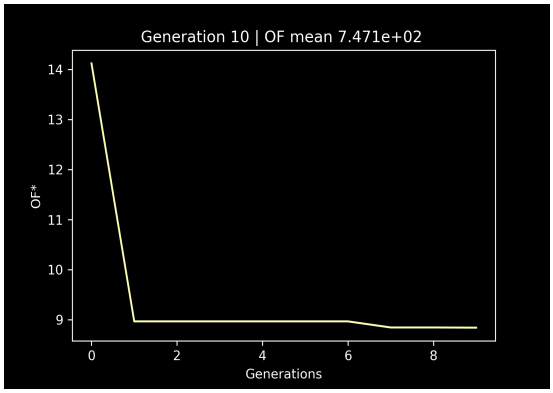


(f) The objective function optimisation for 60 generations with 1 elite copy.

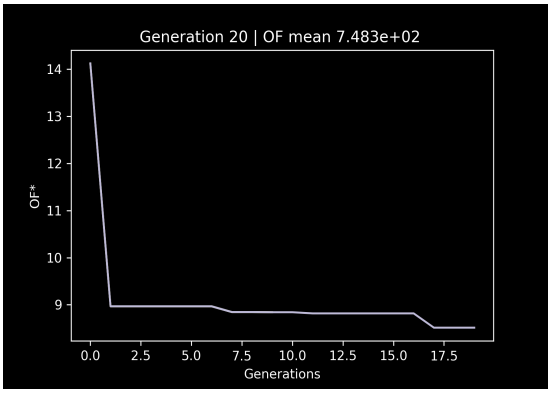
Figure A.1: The figure shows the optimisation of the loss function or par say the training of the NN using GA with Elitism.

A.2.2 GA *without* Elitism

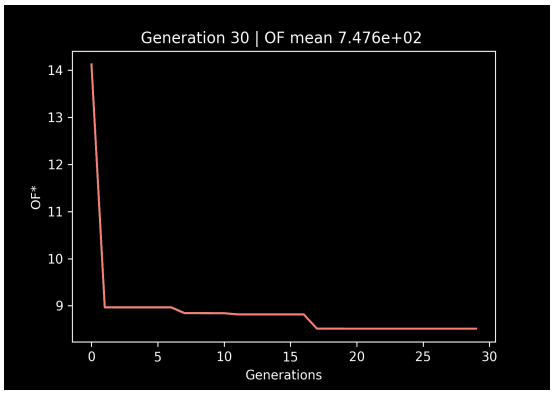
The below Figure A.2 visualises the training process of the NN or say the optimisation of the loss function. The optimisation without elitism will not be strictly increasing or strictly decreasing during the optimisation. The stochastic optimisation can some time hover up and down during the optimisation. It is recommenced to have a convergence criterion while using GA's without elitism. By comparing the Figures A.1 and A.2, having elite copies results in a much smooth optimisation while having no elite copies at all. The best individuals which guarantees less loss are not lost in the stochastic process, while having no elite copies the best individuals can be lost in mutation or while crossing over or in any other stochastic process.



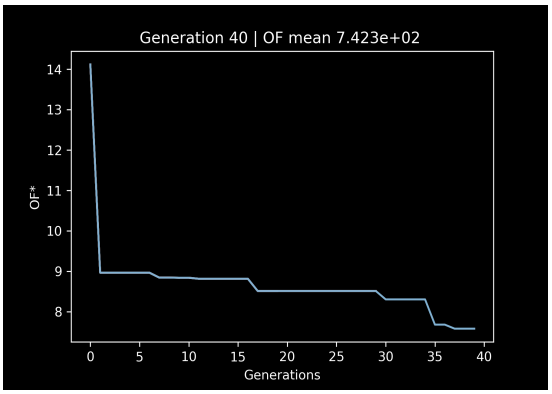
(a) The objective function optimisation for 10 generations with no elite copies.



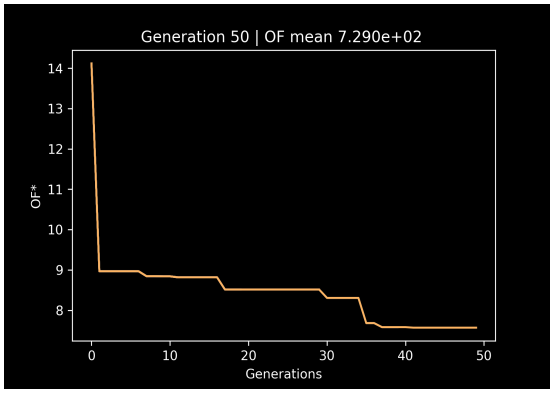
(b) The objective function optimisation for 20 generations with no elite copies.



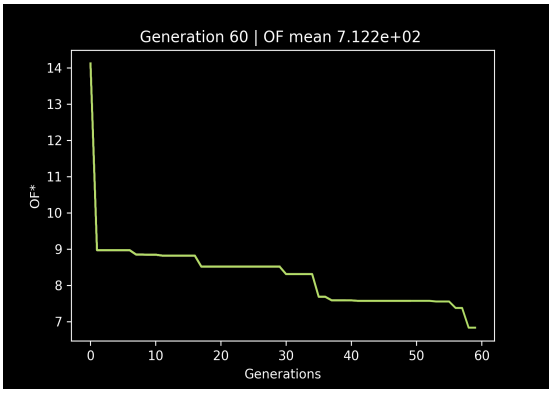
(c) The objective function optimisation for 30 generations with no elite copies.



(d) The objective function optimisation for 40 generations with no elite copies.



(e) The objective function optimisation for 50 generations with no elite copies.



(f) The objective function optimisation for 60 generations with no elite copies.

Figure A.2: The figure shows the optimisation of the loss function or par say the training of the NN using GA *without* Elitism.

B

Utilities

B.1 Methods

The section deals with the methods that are needed to support the optimisation problems. The block of python method is inserted and explained below. And in some parts the algorithms are listed and explained.

The below algorithm takes the wheel speed, demand torque, current engine speed, engine torque and the current gear to find the optimal gear with knowledge of gear ratios and engine map.

Algorithm 1: Algorithm to calculate gear optimal

Result: Gear Optimal y
x $\in \{\omega, \tau_{dem}\}$;
i = 0 ;
while i != $len(\tau_{dem})$ **do**
    $N_{new} = \omega * g_r$;
    $\tau_{new} = \tau_{dem}/g_r$;
    eff = $EngineMap(N_{new}, \tau_{new})$;
    $\hat{y} = argmax$(eff) ;
    return y
end

```
def LossFunction(vehicle):
    lossDistance = 1000 - vehicle.x[-1]/11000
    lossFuel = vehicle.fuelFlow.sum()*100*1000/vehicle.x[-1]
    lossSOC = np.maximum(vehicle.SOC_target - vehicle.soc[-1],0)*1
    if vehicle.completed:
            extraLoss = 0
            if vehicle.emRPM.max() > 12e3:
                    extraLoss += vehicle.emRPM.max()/12e3
            loss = lossSOC + lossFuel + extraLoss
    else:
            loss = lossDistance
    return loss
```

Listing 1: The loss function implemented that optimises the fuel consumption of the PHEV.

```
def VehicleToOptimizeForGA(vehicle):
    def OptimizeFunction(wPopulation):
        lossList = []
        for wNN in wPopulation:
            vehicle.ControllerNN = SetWeightsIntoNN(wNN,vehicle.ControllerNN)
            vehicle.RunSimulation()
            lossList.append(LossFunction(vehicle))
        loss = np.array(lossList)
        return loss
    return OptimizeFunction
```

Listing 2: The complete loss function that optimises the weights of the NN by optimising the loss function (1) recursively.