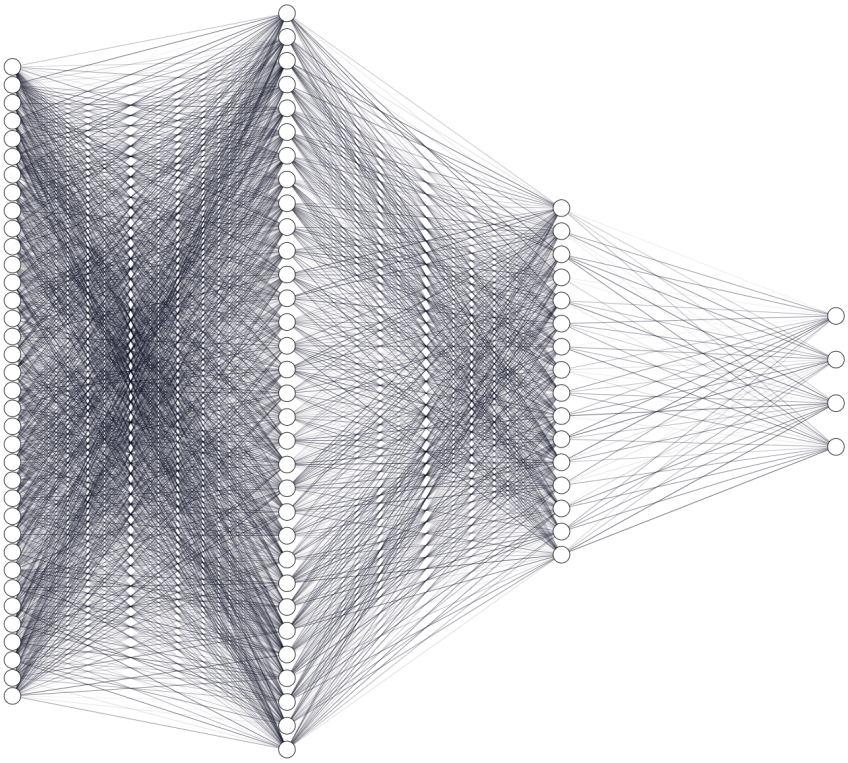
CHALMERS
UNIVERSITY OF TECHNOLOGY



Detection, Semantic Segmentation & Generation of Engine Drive Cycles Using Machine Learning

Master's thesis in Systems, Control and Mechatronics

MATTIAS JOHANSSON
KARL-FREDRIK ZINGAROPOLI

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Detection, Semantic Segmentation & Generation of Engine Drive Cycles Using Machine Learning

MATTIAS JOHANSSON
KARL-FREDRIK ZINGAROPOLI

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Detection, Semantic Segmentation & Generation of Engine Drive Cycles Using Machine Learning
MATTIAS JOHANSSON, KARL-FREDRIK ZINGAROPOLI

Supervisor: Martin Nilsson, Volvo Penta
Advisor: Maryam Lashgari, Department of Electrical Engineering
Examiner: Erik Agrell, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Feed forward artificial neural network used for drive cycle detection, semantic segmentation and generation.

Typeset in LaTeX
Printed by TeknologTryck
Gothenburg, Sweden 2021

Detection, Semantic Segmentation & Generation of Engine Drive Cycles Using Machine Learning
MATTIAS JOHANSSON
KARL-FREDRIK ZINGAROPOLI
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Data that are consequently stored within data lakes are extremely valuable to identify future abnormalities or behaviors for developing products constantly. The data analysis can be done using supervised machine learning to perform predictions of user defined behaviors in time series. The goal of this thesis is to explore real engine data and use applied machine learning as a proof of concept for advanced analytics. The engines may vary in size, area of application, and operation. A general form of an engine operational cycle for a given area of application is referred to as an engine drive cycle. We have proposed three approaches to explore the potential of the engine data. These approaches utilize an Artifical Neural Network (ANN) as a classifier. The dataset is based on a sequentially windowed feature extraction of statistical, temporal, and spectral features. The first approach, called detection, shows the separability of the features between the engine size and applications with 98% accuracy. The second approach, called semantic segmentation, proved that by adopting a similar ANN architecture as used for the detection approach, we can provide an automated way to semantically classify manual annotated data. With an imbalanced dataset, the ANN obtained 98% accuracy of the classifications. The third approach, called generation of engine drive cycles, utilizes the ANN as a fitness function for a genetic algorithm, which illustrated the lack of robustness of the ANN. Only 90% accuracy was achieved because the number of feature representations was lower due to using only one input signal to the ANN. A realistic drive cycle could not be generated, which was concluded to be due to the lack of feature representations in the ANN and few constructional constraints in the genetic algorithm during the generation process.

Keywords: Machine Learning, Artificial Neural Networks, Engine Drive Cycle, Genetic Algorithm, Semantic Segmentation, Feature Engineering, Time series, Combustion Engines, Volvo Penta

Acknowledgements

During this extensive work, we have been provided great support from personnel at Volvo Penta. Mostly we would like to express our greatest thanks to Martin Nilsson and Andreas Nyman, providing more than sufficient aid making this thesis a possibility. A special thanks to Martin Nilsson, our provided supervisor at Volvo Penta, who brought with him his ambitious support and knowledge throughout this thesis. Together with Martin, important discussions were held regarding the different implementations and methods. Also, with his great knowledge in computer programming, machine learning and cloud computing, Martin was able to provide this thesis with his greatest support.

Again, we would like to thank Andreas Nyman, being the coordinator of this master thesis. He is the manager at the field test department at Volvo Penta and is responsible for the collected and structured data. Andreas provided this thesis with all the necessary aid needed for us as students to realize this project in the greatest way possible. He has shown great interest for this project and continuously supported us to ensure that we have all the necessary tools, and contacts, to be able to provide Volvo Penta with a high-quality project.

We would also like to show our greatest thanks towards our advisor, Maryam Lashgari, PhD student at the department of Electrical Engineering. She has provided us with great and ambitious support, giving us feedback regarding the project and mainly the thesis report, and have been available during the semester giving her support for the thesis. Maryam has made sure that the report maintains high quality in terms of both structure and formalities.

Lastly, we would like to give our thanks to our examiner Erik Agrell, Full Professor at the department of Electrical Engineering. During the initial time and during a half time presentation of the thesis, Erik has shown his interest in our work and provided us with valuable feedback. Thanks to his engagement, Erik's feedback has both increased the overall quality and presentation of the thesis.

Mattias Johansson, Karl-Fredrik Zingaropoli, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

Data collection and handling of data have been some of the most important assets for organisations all over the globe for a long time. The reason that data contains such high value is due to the endless possibilities that are brought together with Big Data, Machine Learning (ML), Artificial Neural Networks (ANN), and other topics within Artificial Intelligence (AI). In a combustion engine perspective, there are ways to collect essential data through sensors, which can help to define or characterize an engine's behavior or operation cycle. Most likely, in an organisation that has this type of data and develops these kinds of products, it is important to use the data wisely and try to come up with suitable ways to best take advantage of its great potential. Not only can the data describe how the engine is running, in the form of *Time Series* called ***Engine Drive Cycles***, but it can also be used within fields such as *Predictive Maintenance*, where a neural network can help to detect anomalies by studying a continuous drive cycle. This thesis will study the potential of the structured data obtained by Volvo Penta and how a neural network can detect and identify an engine drive cycle.

1.1 Background

Recent research within ML has received attention not only from the academia, but also from a vast variety of organisations that want to evaluate their own data, which has been collected throughout the years with the intention of increasing statistical knowledge. Volvo Penta, a global supplier of diesel engines and power solutions for industrial and marine applications, is one of these companies that wants to evaluate their acquired data. More specifically, Volvo Penta wants to be able to identify engine drive cycles by using ML and hopefully being able to construct their own drive cycle for a specific application which would hold an accurate representation.

The term "*drive cycle*" refers to the generally repeated (or cyclic) behavior of an engine. Due to random and uncontrollable events during the usage of an engine, it never (or rarely) behaves in the exact same way, and no single cycle is ever exactly repeated. However, the general behavior can still be represented as repetitions of a pattern that is a close approximation of the actual behavior. An example of this is that a mine hauler will start at the top of the mine at a low load, then drive down the mine, stand still as it is loaded with mined materials, run at a high load as it transports the materials up from the mine, and finally stand still as it is unloaded. The machine will then repeat this process throughout the day, every working day.

Slight deviations as to the exact weight of the materials transported and the exact length of the trip can exist, but the general behavior (or cycle) will still be the same. It is this repeated pattern that is a "drive cycle", and it can have a varying length depending on the application and what kind of behavior one wants to study. These drive cycles are used for analytical purposes and helps understand the operational behavior and characteristics of an engine. These behaviors can, e.g., describe either fuel consumption or degree of emissions to name a few, but there is a lot more that an engine drive cycle can describe. However, at the end, it all comes down to the application where the engine is installed since different applications have various purposes, hence different engine behavior and characteristics. There are multiple ways to represent a drive cycle, the above example being a very inexact way. For the purpose of the analysis of an engine behavior, a more exact representation is often needed. Such a representation can be created from time series of engine signals. Any set of signals could theoretically represent a drive cycle, but some signals are more representative (in general) than others. Among the most representative and useful signals are the Engine Speed, Engine Torque and Selective Catalytic Reduction (SCR) Temperature.

Volvo Penta faces numerous challenges when it comes to predicting faultiness in their engine product line. One of the challenges is to be able to identify an engine drive cycle for an application and determine if the cycle describes the general case for a type of application. Identifying a general engine drive cycle is important since it can help to predict faulty behavior which deviates from the regular drive cycle. These deviations could possibly answer questions such as: "Is the engine oversized for a specific application? Is there a risk of an engine breakdown due to critical failures?". Being able to answer these types of questions will allow Volvo Penta to be ahead of the competition and being able to provide key information to their customers, thus, increasing Volvo Penta's own product value and quality of customer service. Most importantly, with this information, Volvo Penta can truthfully fit an engine of the right size for the customer and provide the right conditions to minimize potential risks of engine failures.

Through the years, Volvo Penta's field test and data management unit has collected and structured engine data from applications used by the end user, so called field tests, where the collected data consists of the signals such as engine torque and NO_x emissions, among others. These signals are vital, and mostly used to be able to define an engine drive cycle. Thanks to these field tests, the data will bring Volvo Penta important knowledge about their engine product line and allow Volvo Penta to explore the potential and opportunity that comes with this type of structured data.

1.2 Aim of the Thesis

The aim of this thesis is to thoroughly analyze the data obtained and structured by Volvo Penta from their engine product line, and implement ML by constructing an ANN to detect and identify engine drive cycles from different types of industrial

applications. After the ML model detected and identified a cycle, it should be possible to extract a time span of an engine drive cycle which represents the general operational behavior of that specific application. In addition to the engine drive cycle detection and identification, semantic segmentation is also performed to label engine states. These types of engine states are: *Working*, *Idle*, and *Running*, which describes the current operation of the engine in an application. The thesis will also conduct a brief study by generating a synthetic drive cycle using a so called *Genetic Algorithm* and evaluate if this type of generated cycle is sufficient to represent general engine behavior. With this information about a drive cycle, discussions can be made about its potential, e.g., within predictive maintenance, or to see if an engine installed in an application is correctly fitted for the type of work conducted by a specific application.

1.3 Problem Description

This thesis implements ML on real engine data for the purpose of identifying engine drive cycles for different applications. An engine drive cycle can be represented by signals such as engine speed, SCR temperature, and engine torque, which will be used for analysis throughout this thesis. The problem formulation is stated as follows:

"How can the data be used for classification purposes? Based on data features, how can a neural network identify separate classes? How can the neural network be utilized throughout different scenarios, e.g., detection, segmentation, and generation?"

In detail the thesis will cover:

1. Feature extraction for separate engines to describe their characteristics
2. ML models, taking the extracted features as input, in order to perform:
    - Extraction of top five engine drive cycles representing a general operation for an application
    - Semantic segmentation of engine states
    - Generation of a synthetic engine drive cycle using the genetic algorithm [1]

1.4 Limitations and Challenges

The scope of this thesis is to use an ANN to identify different types of engine drive cycles considering different applications and engine sizes. With this knowledge and by using the acquired structured data, a representative engine drive cycle should be extracted for an application's general operating routine.

While the aim of the thesis is to identify different engine drive cycles, the thesis is limited to evaluate four of Volvo Penta's test engines instead of all existing engines due to time limitations. For training the ANN, a period of two months is used

while a separate third month is used for validation of drive cycle detection. The thesis is meant to be a proof of concept to show the potential of the data that Volvo Penta is in possession of, hence it is decided that it is sufficient to only study a few test engines instead of all of them. Another limitation is the number of signals that are used throughout the thesis which are engine speed, SCR temperature and engine torque. After detecting a cycle, the top five best cycles (considered by the neural network) will be extracted and one hour operation of each one will be shown.

When applying semantic segmentation on engine drive cycles, the thesis is limited to evaluate three engine states: Working, Idle, and Running, for one single engine size and application. The three states are easy to distinguish within a drive cycle and the limited amount of states is to make the study simpler which will not affect the general result of this work. The feature input will only be provided by two signals: Engine speed and engine torque. The SCR temperature was excluded, allowing the labelling process to be more convenient. The actual labeling is conducted over a limited time of five minutes, covering 3000 samples, both saving time and reducing manual labor.

Regarding generation of a synthetic engine drive cycle using genetic algorithm, the scope of the thesis is limited to study only the genetic algorithm approach and will not evaluate other possible methods for generation of synthetic cycles. The thesis will provide possibilities of future work and other useful areas for engine drive cycle detection, segmentation and generation.

1.5 Outline of the Thesis

The report is divided into six chapters with introduction being Chapter 1. Further, Chapter 2 contains theoretical background regarding the tools and theoretical concept used throughout the thesis.

Chapter 3 presents the proposed methods and flowcharts used to answer defined research questions. The presented methods explain how the theoretical concept was applied in each case, along with the process of obtaining the results.

In Chapter 4, the results from applied machine learning and the related discussion are presented. The discussion is presented after each method implementation.

Chapter 5 presents conclusions, which is based on the results and discussions carried out in the previous chapter.

Finally, future work is presented in Chapter 6 that gives suggestions on how to proceed with the thesis work further and additional improvements that were not covered in this thesis.

2
Theoretical Background

This chapter regarding theoretical background provides insight for all the tools used throughout the thesis. The chapter covers thoroughly; *Time Series*, *Data Preparation*, *Feature Analysis*, *Classification Algorithms*, *Neural Networks* and its properties, and lastly the fundamentals of *Genetic Algorithm*.

2.1 Time Series

A time series is a set $\boldsymbol{T}$ containing samples $\boldsymbol{g} = (g_1, g_2, ..., g_N)$ together with corresponding time samples $\boldsymbol{t}_N = (0,\ \Delta t,\ 2\Delta t, ...,\ (N-1)\Delta t)$. Here N is the number of samples used in $\boldsymbol{T}$, and Δt [s] is the time between the samples, also known as sampling time. This leads to the definition of sampling frequency defined as $\frac{1}{\Delta t}$ [Hz]. Further, a time series is a set of discrete samples where the magnitude of the samples is represented by a unit, e.g., engine torque [Nm], engine speed [RPM], or temperature [°C]. As time series are sequential, methods have been developed for analysis of time series to help understand patterns and characteristics of the data. As an example, seasonal trends are patterns that could exist within yearly measurements (e.g., temperature in a city) opposed to characteristics that can simply be described by a statistical mean or median for a segment of an arbitrary time series.

The above explanation is for *univariate time series* which are single measurements sampled from a physical system. *Multivariate time series* are two or more measurements sampled from the same system simultaneously and can be simultaneous measurements of, e.g., engine speed, and SCR temperature through time. Multivariate time series may not be sampled at the same frequency and can differ depending on the configuration of the sensor module. In this thesis, an *engine drive cycle* is a time series measured by engine signals, as previously stated in 1.1, which can be both univariate or multivariate.

2.1.1 Data Preparation

Annotating datasets is an important part when working with classification algorithms. When using annotated data as a training set for the ANN, it is called *Supervised Learning*, opposed to *Unsupervised Learning* that has no indication of classes [2]. This thesis only presents a supervised learning approach for the classification problem.

Manual data annotation is a method to create self-defined classes, that data annotators define themselves, with the purpose of training an ANN. The class representations are based on the features from the training dataset. The quality of the annotations are highly dependent on the expertise of the data annotators, defining and labeling the data, as they must be able to have a clear understanding of how to interpret the data. The interpretation of the data is related to its corresponding features, and understanding that these can be related to the annotations is vital for data annotators. If the annotations are executed perfectly, this will increase the separability between the classes, thus, increasing the accuracy of the ANN during its classification process.

Time series segmentation is a method to divide a time series into segments of samples. The data annotators must select each segment, defined as the windowed signal $\boldsymbol{s}$ with length t_s, of the time series and assign this segment to a class. The work is complete when the whole length of a time series $\boldsymbol{T}$, which consists of N samples, is subsequently divided into a number of N_d equally spaced, contiguous segments of length t_s, with a class correspondence. These equally spaced segments are referred to as the data points. The total amount of data points in the annotated dataset will therefore be equal to

$$N_d = \frac{N}{t_s}. \tag{2.1}$$

The data points are now represented as a segment of samples, but will be further prepared to be used as an input to a classification algorithm. Features from the segments, $\boldsymbol{s}$, from the time series can be extracted, this is called a *feature extraction*. The main purpose is to create a dataset that the ANN is compatible with. The feature extraction creates a new vector, for every segments, which consists of N_f features. This feature vector is defined as $\boldsymbol{x} = (x_1, x_2, \ldots, x_{N_f})$. The class that corresponds to the feature vector is encoded as a one-hot vector defined as

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_{N_c}), \text{ where } y_j = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \boldsymbol{C}_j \\ 0 & \text{if } \boldsymbol{x} \notin \boldsymbol{C}_j \end{cases}, j = 1, 2, \ldots, N_c \tag{2.2}$$

where $\boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{N_c})$ is a set that consists of N_c number of classes. Further, the data points are defined as $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_d})$ with the class points $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_d})$. The dataset that is created by this process is defined as $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{Y})$ and will be used as an input for the classification algorithm. The features that will be extracted from the segments are presented in the next section.

2.1.2 Features

This section covers every feature used, defining and describing them in detail. A segment, $\boldsymbol{s}$, of samples from $\boldsymbol{T}$ has various features within the statistical, temporal and spectral domain that are able to represent the characteristics of the samples. The features have to be separable with the trivial explanation that two segments of two time series may represent two different classes. In order to distinguish, e.g., *class 1* from *class 2*, the features have to be separable. The features may have similar values for some samples, thus creating the need for a larger amount of features that have the attribute of being separable.

The *statistical domain* consists of statistical properties, for a set of samples, that have the purpose of estimating a population or describing a sample set. Statistical features contain simple calculations such as the mean or median of the amplitudes of a time series.

The *temporal domain* is described as ratios or relative events between events through the samples and has no information about the ordering of the sequence. Basically these are features that have an intuitive physical interpretation, just as the statistical domain, such as zero crossing rate or autocorrelation.

The *spectral domain* is accessed by transforming a time series to the frequency domain using the *Fourier Transform* in order to extract features in the frequency domain. The features included in this domain are, e.g., the maximum frequency and fundamental frequency. Table 2.1 presents a set of features that has been extracted from the domains, where $\boldsymbol{s}$ is the windowed signal. The windowed signal has a predefined length, t_s, that is a segment of a time series. The windowed signal is sequentially extracted from a time series $\boldsymbol{T}$ and subsequently used as input to perform feature extraction.

Table 2.1: The selected features to be extracted from the window $\boldsymbol{s}$ [3].

Autocorrelation	$x_1 : \sum_{n=1}^{t_s-l} s_n \overline{s_{n+l}}$, where $\overline{s_n}$ is the complex conjugate of s_n, and l is a lag set to $l = 1$.
Max	$x_2 : \max(\boldsymbol{s})$
Maximum Frequency	x_3 : Computes the Fast Fourier Transform and returns the frequency of 95 % of the cumulative sum from the magnitude.
Mean	$x_4 : \mu_s = \text{mean}(\boldsymbol{s})$
Mean Absolute Deviation	$x_5 : \frac{1}{t_s} \sum_{n=1}^{t_s} \lvert s_n - \mu_s \rvert$
Mean Absolute Differences	$x_6 : \mu_{\lvert\Delta \boldsymbol{s}\rvert} = \text{mean}(\lvert\Delta \boldsymbol{s}\rvert)$, where $\Delta \boldsymbol{s}$ is the discrete difference along the vector $\boldsymbol{s}$ defined as: $\Delta \boldsymbol{s} = s_{n+1} - s_n$ for $n = 1, \ldots, t_s - 1$.
Mean Differences	$x_7 : \mu_{\Delta \boldsymbol{s}} = \text{mean}(\Delta \boldsymbol{s})$
Min	$x_8 : \min(\boldsymbol{s})$
Slope	x_9 : Fits a linear equation to $\boldsymbol{s}$ to fit a regression line $s^{lin}(n) = \bar{m}n + \bar{b}$, returns the coefficient $\bar{m}$. $\bar{b}$ is the y-intercept.
Peak to Peak Distance	$x_{10} : \lvert \max(\boldsymbol{s}) - \min(\boldsymbol{s}) \rvert$
Standard Deviation	$x_{11} : Std(\boldsymbol{s}) = \sqrt{Var(\boldsymbol{s})}$
Variance	$x_{12} : Var(\boldsymbol{s}) = \text{mean}(\lvert \boldsymbol{s} - \mu_s \rvert^2)$

Table 2.1 defines the feature extraction of one signal where each row corresponds to one feature. Each row of the individual features corresponds to each element in $\boldsymbol{x} = (x_1, x_2, \ldots, x_{12})$, as an example. Note that if the feature extraction is done on a multivariate time series, i.e., multiple signals sampled at the same time, the same feature extraction is performed for each signal and concatenated in to $\boldsymbol{x} = (x_1, x_2, \ldots, x_{N_f})$.

2.2 Feature Analysis

When selecting features, it is important to choose the ones that are more significant for a time series, since those features will help the classification algorithm (section 2.1.1) to more easily determine the correct class that the features describe. The selected features in this thesis are based on a large number of features from a library known as *Time Series Feature Extraction Library* (TSFEL) [3]. There are different ways to evaluate feature importance for classification algorithms, a common method is to iteratively take a random subset of features as input to the network and subsequently log the metric score. The subset that yields the best metric score will be

the one that is more important, thus, potentially reducing the dimensionality of the input by removing less significant features. This is called the *wrapper method* and is known for being computationally expensive [4].

Throughout this thesis, an alternative feature importance evaluation method is used called *Integrated Gradients*, proposed in [5] in 2017, and is specifically designed for ANN's. The integrated gradients are accumulating the gradients that are calculated between a feature input $\boldsymbol{x} \in R^{N_f}$ and a baseline $\boldsymbol{x'} \in R^{N_f}$ as a path integral

$$\text{IntegratedGrads}_i(\boldsymbol{x}) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial F\left(\boldsymbol{x'} + \alpha\left(\boldsymbol{x} - \boldsymbol{x'}\right)\right)}{\partial x_i} d\alpha. \tag{2.3}$$

The baseline, $\boldsymbol{x'}$, is a vector of zeros while $\boldsymbol{x}$ is the extracted input features and $\frac{\partial F(\boldsymbol{x})}{\partial x_i}$ is the gradient of the model's prediction function given each feature x_i. The parameter α is an interpolation constant, with respect to the feature x_i from the baseline x_i' up to 1, as the features are normalized. To compute the analytical function, it is needed to approximate equation (2.3) as a discrete expression. This leads to the expression

$$\text{IntegratedGrads}_i^{approx}(\boldsymbol{x}) = (x_i - x_i') \sum_{k=1}^{m} \frac{1}{m} \frac{\partial F\left(\boldsymbol{x'} + \frac{k}{m}\left(\boldsymbol{x} - \boldsymbol{x'}\right)\right)}{\partial x_i}, \tag{2.4}$$

where k is a scaling term and m is the number of steps in the Riemann approximation. The purpose of analyzing the path integral is to help to understand how significant the features are at the classification stage. However, even though some features are a good addition to the network and provide better classification accuracy for one class, the features can also be less significant or even affect the network negatively in general. The interpretation for this attribution method applied on each input is that a higher numerical value would increase the score of that class, while a lower numerical value would decrease the score of that class. This numerical value is henceforth referred to as *significance score*. Further, the baseline input $\boldsymbol{x'}$ may be an important parameter in the interpretability of the integrated gradient which is highlighted in [6].

2.3 Neural Networks for Classification

Classification of a time series is a strategy to classify one or more well defined classes based on features. To classify something is to assign, in this case segments of time series, to a class which we interchangeably refer to as "mapping". This thesis utilizes a feature based time series extraction to annotate the classes. The annotations are based on manual annotations and are defined by the user as classes. The classes are arbitrary, but aims to map a set of features $\boldsymbol{x}$ to the classes $\boldsymbol{y}$ defined as $F(\boldsymbol{x}) : \mathbb{R}^{N_f} \rightarrow \mathbb{R}^{N_c}$. Here $F(\boldsymbol{x})$ is a function that represents the classification algorithm. The performance of the classification algorithms will vary depending on the complexity, linearity or non-linearity of the relationship between features and the class. An ANN has been proven to be a classifier that models the non-linearity of a relationship in a simple and straightforward manner [7] and will therefore be used as the classifier algorithm in this thesis.

Recently, when talking about different ML concepts, neural networks has been in the center of attention. The concept of neural networks dates back to 1943 where McCulloch and Pitts introduced the first computational model of a neuron [8]. A single neuron does not define a neural network but it is the collection of multiple neurons that defines the network. This is what Rosenblatt introduced in 1953, a method of linking multiple neurons together in multiple layers. This was to be called *Multilayer Perceptron* (MLP). A simple MLP network is illustrated in Figure 2.1.

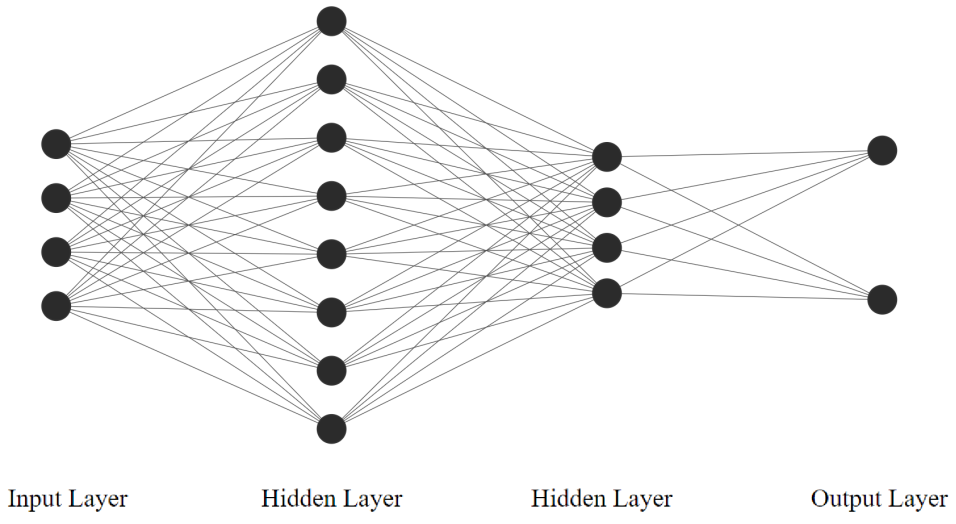


Figure 2.1: Illustration of an MLP with input layer, two hidden layers of various dimensions and one output layer.

The MLP has three categories of layers and consists of the *input layer*, *hidden layer*, and lastly the *output layer*. The input layer accepts $\boldsymbol{x} \in \mathbb{R}^{N_f}$, and consists of a set of neurons and represents the features. The output layer dimension corresponds to the classes $\hat{\boldsymbol{y}} \in \mathbb{R}^{N_c}$, where $\hat{\boldsymbol{y}}$ is the predicted output from the neural network. The number of hidden layers can be chosen arbitrarily and these define how "deep" a

neural network is. The hidden layers consists of multiple sets of neurons where each neuron is sequentially connected between the layers. The neurons within the hidden layers apply a transformation of the values from the previous layer and subsequently apply an activation function $f(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$, in order to help the network to learn complex patterns of the mapping.

In practice, the MLP is widely used for classification and regression problems when building a network with more neurons and more hidden layers. More specifically, this is called a *Deep Neural Network* (DNN) and utilizes deep learning for training a network for a specific application. Stating in terminology terms: An MLP is a subset of DNN, as DNN can have loops in its layers whereas MLP is a finite acyclic graph, i.e., a feed forward neural network. The term ANN that has been used so far is a general form of a neural network that covers a wide set of neural network structures. Throughout the rest of this thesis, the neural network will be referred to as the ANN and has the same structure as an MLP. The output of the ANN is defined as

$$\hat{\boldsymbol{y}} = F(\boldsymbol{x}, \boldsymbol{\theta}), \tag{2.5}$$

where $\hat{\boldsymbol{y}}$ is a vector of the predicted outputs of the ANN, $\boldsymbol{x}$ are the input features and $\boldsymbol{\theta}$ contains the weights ($\boldsymbol{W}$) and biases ($\boldsymbol{b}$) for the neurons. $F(\boldsymbol{x}, \boldsymbol{\theta})$ is the ANN in which all the layers and activation functions are stored and the whole network, with one hidden layer, is defined as

$$\hat{\boldsymbol{y}} = F(\boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{z}(\boldsymbol{x}, \boldsymbol{\theta})) = f(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}). \tag{2.6}$$

An illustration of a simple ANN with only one hidden layer is presented in 2.2.

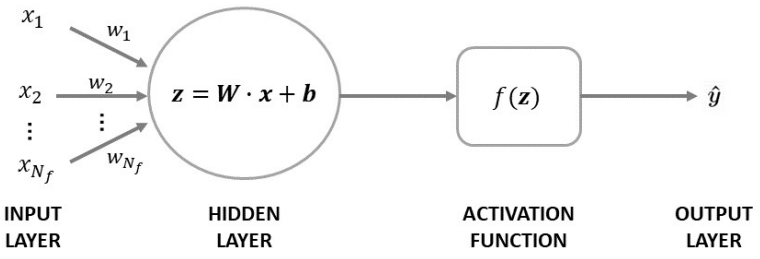


Figure 2.2: Visualization of the network function. Note that this network consists of only one hidden layer, which is only one neuron, and one subsequently applied activation function. The network may also have several hidden layers with activation functions subsequently applied after each neuron.

To create a score distribution between the classes in the output layer, another activation function is applied after each neuron in the output layer. The scalar with the largest value of the vector $\hat{\boldsymbol{y}}$ at this point, which is selected as $argmax(\hat{\boldsymbol{y}})$, belongs to one of the classes in $\boldsymbol{C}$. The coming sections will cover: linear transformation, activation functions and loss function, followed by metrics for evaluating the performance of the ANN.

2.3.1 Linear Transformation

The neurons for a single layer that makes up the neural network are stored in a so called linear transformation which is defined as a single layer feed forward network. The layer applies a linear transformation to the incoming features as

$$\boldsymbol{z} = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}, \tag{2.7}$$

where $\boldsymbol{W}$ is a set of weights of the adjacent edges between all the neurons in the layer, $\boldsymbol{b}$ is a set of all the biases for each neuron, and $\boldsymbol{x}$ is the input features. The ANN input is a layer that consists of N_f number of neurons, the hidden layers consists of an arbitrary amount of neurons and layers and the output layer consists of a linear transformation that outputs N_c number of classes. The number of hidden layers are highly dependent on the separability of the data points. If the data points were completely linearly separable, the usage of several hidden layers is not motivated [9]. A set of linear transformations is also referred to as *Fully Connected Layers* (FCL).

2.3.2 Rectified Linear Unit Activation Function

The *Rectified Linear Unit* (ReLU) activation function is a piecewise linear function that thresholds values at 0 when $\boldsymbol{z} < 0$, but stays positively linear when $\boldsymbol{z} \geq 0$. Note that f_{ReLU} is an activation function that operates independently on each element of $\boldsymbol{z}$. The mathematical expression of the ReLU function is expressed as

$$f_{ReLU}(\boldsymbol{z}) = \begin{cases} \boldsymbol{z}, & \text{if } \boldsymbol{z} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{2.8}$$

and Figure 2.3 visualizes the function.

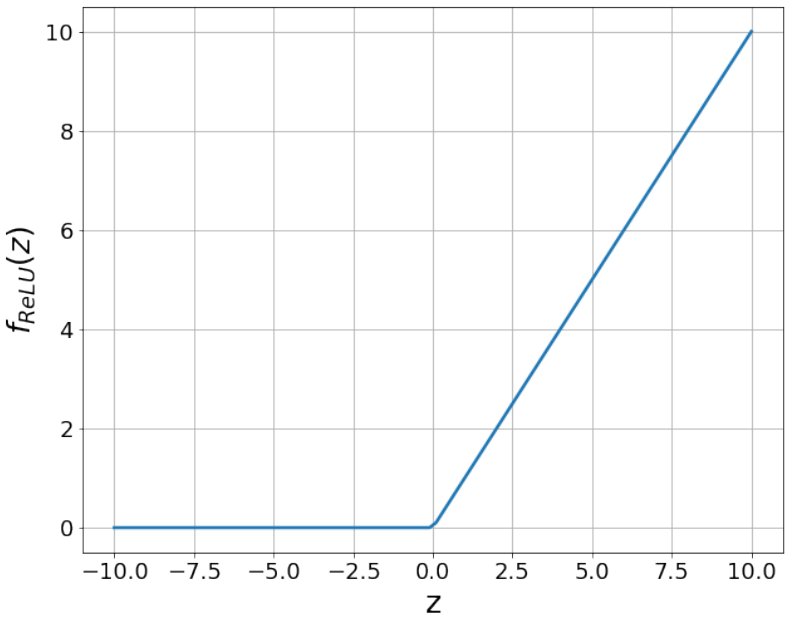


Figure 2.3: Visualization of the ReLU activation function.

The activation function in a neural network defines the complexity on the mapping of the features to classes. If the majority of activation functions in the neural network are linear activation functions, then the network will struggle to learn non-linear mappings. The ReLU as a function is nonlinear, due to the threshold, but piecewise linear as initially stated.

The main advantage with a ReLU activation function is that it is computationally inexpensive as opposed to non piecewise linear activation functions since the calculations are only limited to logical operators and not exponentials or divisions [10]. As the function rectifies values below zero, the vanishing gradient problem is eliminated. The vanishing gradient problem occurs when using gradient based optimization algorithms and backpropagation in an ANN. As the weights during training are updated with respect to the partial gradient of the loss function for previous nodes, the gradient may be small in some cases, therefore rendering the training to be negligible for that epoch [11].

2.3.3 Sigmoid Activation Function

For multi-label classification, the output layer applies a nonlinear activation function $f_{Sigmoid}(\boldsymbol{z}) : \mathbb{R} \Rightarrow \mathbb{R}$ which creates a distribution of scores for each output. The sigmoid activation function, $f_{Sigmoid}(\boldsymbol{z})$, is responsible for transforming the sum of the last layer into a score distribution between the defined classes. This function suppresses the output of the function to values between 0 and 1, the mathematical expression is shown as

$$f_{Sigmoid}(\boldsymbol{z}) = \frac{1}{1 + \exp(-\boldsymbol{z})}, \tag{2.9}$$

and Figure 2.4 visualizes the function.

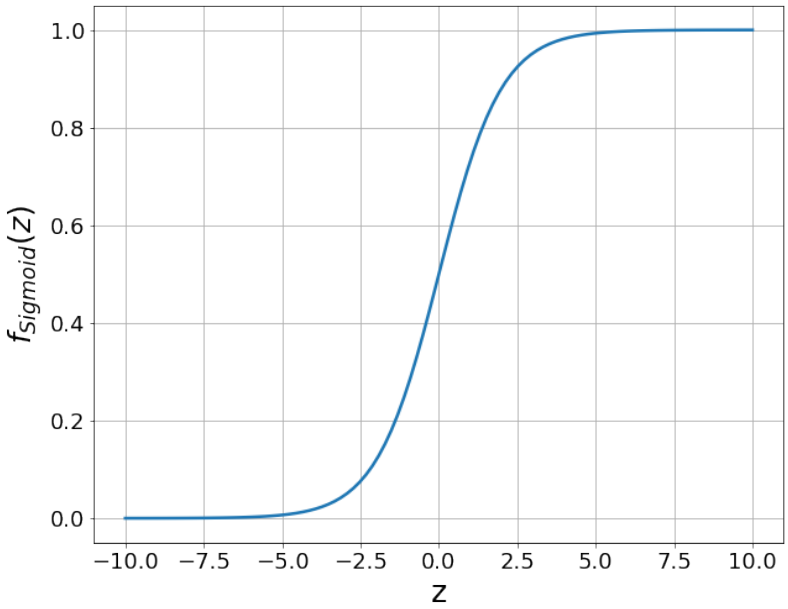


Figure 2.4: Visualization of the sigmoid activation function.

2.3.4 Binary Cross Entropy Loss

The *Binary Cross Entropy* (BCE) is a loss function and is normally used for binary classification [12]. The BCE, together with the sigmoid layer which is at the end of the ANN, calculate the error of reconstruction for the encoded predicted value. Mathematically, the loss function can be expressed as

$$\ell\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right) = -\frac{1}{N_c}\sum_{i=1}^{N_c} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log\left(1 - \hat{y}_i\right), \tag{2.10}$$

where $\hat{y}_i$ is the i-th predicted value and y_i is the true value of the i-th label. The value of y_i is either 0 or 1, while $\hat{y}_i \in [0, 1]$ which corresponds to the output of the sigmoid activation function. By using the BCE there is a degree of freedom for adding several categorical classes to the model. The BCE will act as an average of the *Categorical Cross Entropy* (CCE) loss result when there is a discrete class distribution. Further, the loss function $\ell\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right)$ presents the loss given by the training dataset. The total loss term for the network can be expressed as

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{T}) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}} \ell\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}} \ell\left(F\left(\boldsymbol{x}, \boldsymbol{\theta}\right), \boldsymbol{y}\right), \tag{2.11}$$

where $\hat{\boldsymbol{y}} = F\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ is the predicted output of the neural network. By using the training data, $\mathcal{T} \subset \boldsymbol{D}$, the aim is to find the variables $\boldsymbol{\theta}$ that minimize the loss function. The objective function is expressed as

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{T}). \tag{2.12}$$

2.3.5 Stochastic Gradient Descent

The loss function measures how accurate the networks predictions are, the next step is to use an optimization algorithm with the intention to minimize the loss. There is no guarantee to find a function for global minimization, therefore, the solution is to rely on local optimization.

The *Stochastic Gradient Descent* (SGD) optimization algorithm aims to differentiate the loss function and moves towards the negative gradient where the value increases more rapidly. As the algorithm is stochastic with respect to the training set, the algorithm may encounter fluctuations during the training. To counter this, a version of a weighted moving average is introduced as *momentum* to help the SGD algorithm be directed towards the optimum value efficiently [13]. The pseudocode for the SGD algorithm with momentum is presented in Algorithm 1.

Algorithm 1: Stochastic Gradient Descent with Momentum [12]

Require: Training Set: $\mathcal{T}$; Learning Rate μ; Normal Distribution Std: σ; Momentum Term Weight: ρ

Ensure : Model Parameter $\boldsymbol{\theta}$

1 Initialize parameter with Normal distribution $\boldsymbol{\theta} \sim N(0, \sigma^2)$
2 Initialize Momentum term $\Delta \mathbf{v} = \mathbf{0}$
3 Initialize convergence tag = False
4 **while** *tag == False* **do**
5 Shuffle the training set $\mathcal{T}$
6 **for** *each data instance* $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ **do**
7 Compute gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}, \boldsymbol{y}))$ on the training instance $(\boldsymbol{x}, \boldsymbol{y})$
8 Update term $\Delta \mathbf{v} = \rho \cdot \Delta \mathbf{v} + (1 - \rho) \cdot \nabla_{\theta} \mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}, \boldsymbol{y}))$
9 Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \mu \Delta \mathbf{v}$
10 **end**
11 **if** *convergence condition holds* **then**
12 tag = True
13 **end**
14 **end**
15 Return model variable $\boldsymbol{\theta}$

To ensure a stable local optimum, the value for the learning rate, μ, must be tweaked. By setting high values for the learning rate, it will influence the optimization in an uncontrolled pattern with a risk of missing the local optima. However, by setting low values for the learning rate, it will make computations slower and brings the risk of converging towards the local optima prematurely. The momentum term weight, ρ, is chosen to dampen the oscillations for the optima to be reachable. If the momentum term weight is set too low, the functionality of the momentum will be neglected, while if the momentum is set too high, it forces the stochasticity of the SGD optimization to diminish.

2.3.6 Backpropagation

Within ML, algorithms such as *backpropagation* are used for calculating the gradients of a loss function in a neural network. The algorithm computes the gradient of the defined loss function (as seen in Algorithm 1, line 7), which is the BCE, with respect to the weights within the network. The purpose is to update the weights such that the total loss is minimized. The gradient is computed for each layer starting from the last layer and working itself backwards through the neural network [14] [9].

2.3.7 Metrics

The performance of an ML algorithm depends on how well the algorithm predicts the target label (classification) or value (regression). This thesis is dealing with predictions, i.e., classifications, of the target classes. The metrics for evaluating the performance have a large impact on the interpretability on how well the algorithm behaves, thus also help discovering the weakness of predicting a certain class or value. This weakness is referred to as the mispredictions of the classes, often

a result due to lack of data representations for the classes. With a given dataset, the predictions may be good for some of the classes, and bad for others. This may depend on the complexity of the feature mapping for the latter case together with the low amount of data points that the network is trained with. The case where a class have a lower amount of data points than the other classes is referred to as an *imbalanced dataset*.

Below, relevant metrics are presented to evaluate a multi-class classification ANN. The metrics presents the performance of the classifications by comparing the predicted classes, $\hat{\boldsymbol{y}}$, with the true classes, $\boldsymbol{y}$. The ANN iterates over the dataset $\boldsymbol{D}$ and outputs the values for a prediction dataset defined as

$$\hat{\boldsymbol{Y}} = (argmax(\hat{\boldsymbol{y}}_1), argmax(\hat{\boldsymbol{y}}_2), \ldots, argmax(\hat{\boldsymbol{y}}_{N_d})). \tag{2.13}$$

The dataset to compare the predicted classes with, known as the true classes, is defined as

$$\boldsymbol{Y}^{true} = (argmax(\boldsymbol{y}_1), argmax(\boldsymbol{y}_2), \ldots, argmax(\boldsymbol{y}_{N_d})). \tag{2.14}$$

The metrics that are used in this thesis to evaluate the performance of the ANN are the following: *Accuracy*, *Balanced Accuracy* and *Confusion Matrix*.

2.3.7.1 Accuracy

Inherently, when discussing classification performance, the accuracy of a prediction is the most fundamental and intuitive metric for a balanced dataset [15]. However, for an imbalanced dataset where a class with a minimal amount of data points might be fully mispredicted by the network, while other classes containing the majority of data points will more likely be correctly predicted. This result will present an accuracy that is not a fair representation of the class with minimal amount data points. Hence, when using the accuracy metric it is preferred to have a balanced dataset. The accuracy is defined as

$$\text{accuracy}(\boldsymbol{Y}^{true}, \hat{\boldsymbol{Y}}) = \frac{1}{N_d} \sum_{u=1}^{N_d} \mathbf{1}\left(\hat{Y}_u = Y_u^{true}\right), \tag{2.15}$$

where $\hat{Y}_u$ is the u-th predicted value and Y_u^{true} is the true value of the u-th label. Further, $\mathbf{1}\left(\hat{Y}_u = Y_u^{true}\right)$ is an indicator function that is defined as

$$\mathbf{1}(Y_u^{true}, \hat{Y}_u) = \begin{cases} 1 & \text{if } \hat{Y}_u = Y_u^{true} \\ 0 & \text{if } \hat{Y}_u \neq Y_u^{true} \end{cases}. \tag{2.16}$$

2.3.7.2 Balanced Accuracy

The balanced accuracy is a metric that takes class imbalance into account, and therefore, gives a realistic performance metric for imbalanced datasets [16].

The balanced accuracy is defined as

$$\text{balanced-accuracy}\left(\boldsymbol{Y}^{true}, \hat{\boldsymbol{Y}}, \boldsymbol{q}\right) = \frac{1}{\sum \hat{\boldsymbol{q}}} \sum_{u=1}^{N_d} \mathbf{1}\left(\hat{Y}_u = Y_u^{true}\right) \hat{q}_u, \tag{2.17}$$

where $\hat{\boldsymbol{q}}$ is the sample weight. The terms "sample" is not to be confused with the samples inside a time series or a segment. The initial sample weight, $\boldsymbol{q}$, is calculated to assign weights to the predictions in each class. The calculations are based on how large proportion of each class consists within the dataset, and the predictions that corresponds to that class in the dataset are weighted accordingly. For every u value, the sample weight, $\hat{q}_u$, is adjusted to

$$\hat{q}_u = \frac{q_u}{\sum_{r=1}^{N_d} \mathbf{1}\left(Y_r^{true} = Y_u^{true}\right) q_r}. \tag{2.18}$$

2.3.7.3 Confusion Matrix

To visualize the performance of a ML classification algorithm there is a table layout that presents the performance in a pedagogical way. The table layout is called a confusion matrix and contains the predicted class distribution. The rows in the matrix correspond to the predicted class while the columns correspond to the actual class. An illustration of the example classes {*Class 0*, *Class 1*, *Class 2*} can be observed in Figure 2.5.

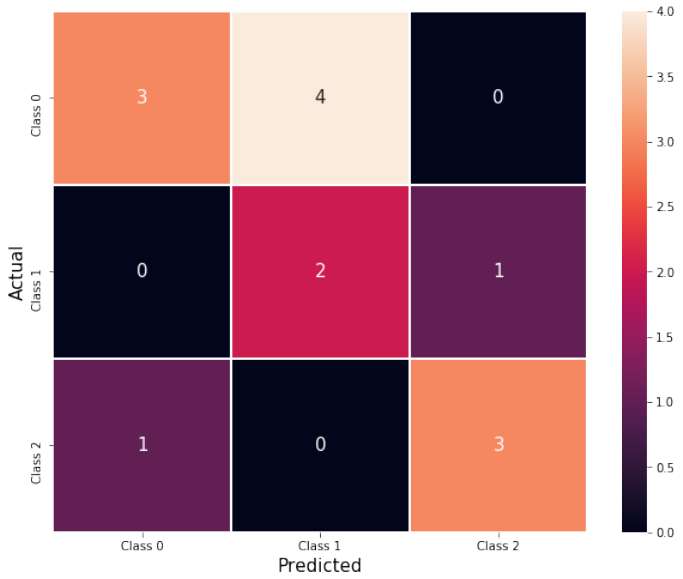


Figure 2.5: Confusion matrix illustration.

In Figure 2.5 the diagonal corresponds to the correct class predictions. The interpretation of observing, e.g., the first row, second column, is that *class 0* was predicted incorrectly as *class 1* and done so four times. An ideal confusion matrix would be a full diagonal matrix.

2.4 Genetic Algorithm

When performing time series generation throughout this thesis, a successor to *Evolutionary Algorithms* called *Genetic Algorithm* will be used. The main purpose of a genetic algorithm is to solve problems where deterministic algorithms or other traditional algorithms are too costly [17]. In other words, when faced with a complex problem, which in this case is to generate an accurate time series representing an engine drive cycle, a genetic algorithm can optimize a solution and make the computations in a relatively fast time. The algorithm is inspired by the mechanism of natural selection, that is a biological process in which stronger individuals are likely to be winners and will proceed to evolve with upcoming generations [1].

When initiating the genetic algorithm, a population that consists of G uniform distributed random vectors is generated, called individuals. Each individual lies within a minimum and maximum value, and has a predetermined length of t_s. Each iteration of the algorithm is called a generation which leads to calling this the initial generation. During each generation, an evaluation of a *fitness function* (defined by an objective function in the optimization problem) is done, comparing the new individuals with their predecessors and saving the individuals with best achieved fitness score. For the iteration process, a threshold or fitness condition is set to be fulfilled, and the algorithm will iterate until sufficient score is reached or if a generation limit is set. Then a solution, or generation in this case, is obtained. During the iteration process, the genetic algorithm performs a so called parent selection from the current population. That is, a stochastic selection of the best suited parents within the current population. Then a crossover operation is done between each parent to form the new generation of population with a certain probability of performing a mutation to these crossovers. Lastly, the algorithm calculates the fitness function once again to check if the set fitness condition is fulfilled. If the fitness score is not updated in E number of iterations, the algorithm will stop. Finally the genetic algorithm will give a solution that the iteration has achieved consisting of a vector, i.e., an individual, of length t_s, which equals to a drive cycle. Algorithm 2 presents a pseudocode on obtaining a vector that achieves the highest fitness score based on the optimization problem.

Algorithm 2: Genetic algorithm pseudocode

Input : G, $[min, max]$, t_s, *number of iterations*, E
Output: Vector with highest fitness score

```
Start
Initialization of a random population of size G of individuals ∈ [min, max] with
  size t_s
Define fitness function
Compute fitness score for individuals in the population
iteration = 0
while iteration ≠ number of iterations do
    Selection of parents from current population
    Crossover operation for new population (with certain probability)
    Mutation of new population (with certain probability)
    Compute fitness score for new individuals
    iteration = iteration + 1
    if Maximum fitness score is not improved in E iterations then
        break
    end
end
Return individual with highest fitness score
Stop
```

3
Methods & Implementation

This chapter presents an explanation of the methodology to obtain the detection, semantic segmentation and generation of engine drive cycles. Figure 3.1 presents an overview of the workflow scheme through the thesis.

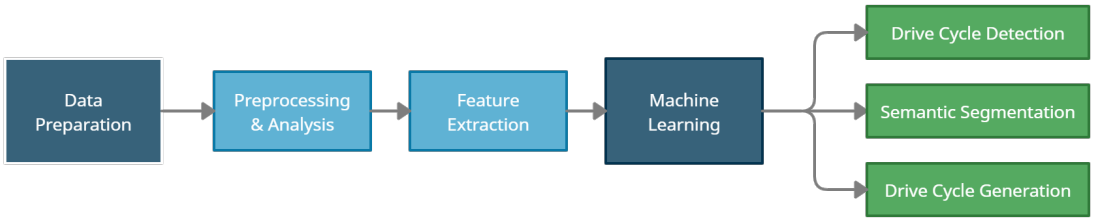


Figure 3.1: The Workflow to obtain the detection, semantic segmentation and generation of engine drive cycles.

The following sections will explain each block in Figure 3.1 in more details, starting from Section 3.1 which will cover preprocessing and feature extraction. Section 3.2 presents the method of constructing a machine learning algorithm for classification in order to perform detection, semantic segmentation and generation of engine drive cycles.

3.1 Data Preparation

Data is the most important resource, and even though the data is structured from the beginning, there is still a need to divide the data preparation process into two stages: *Preprocessing & Analysis*, and *Feature Extraction* as explained in the following subsections.

3.1.1 Preprocessing & Analysis

The data is the main source of information which will be used throughout the thesis and within the ANN. The data is raw sensor readings, and model based calculations, from industrial engines, which exists in a Microsoft Azure data lake and consists of *.mat* and *.delta* formats. The *.mat* formatted files have been used in this thesis for convenience.

The selected engines are presented in Table 3.1 together with the months of data

that was available within data lakes and used in the ANN, depending on the applied method. The months were chosen in a sequential order where the goal was to acquire a balanced dataset with equal amount of data points for each class. Another reason that months are sequentially chosen is to minimize, e.g., extreme temperature fluctuations. The assumption in this thesis is that no extreme temperatures will be present within the validation month in relation to the train and test months. The **Class Name** notation will be used for each class further on.

Table 3.1: The engines and their respective application. The train/test months are extracted to create the train/test dataset for the ANN. The third month is used for validation where the network is tested for its classification capabilities.

Class Name	Engine Size	Pseudonym	Application	Train/Test Set	Validation Set
D11 T	D11	T	Logstacker	2020-08 2020-09	2020-10
D13 T	D13	T	Logstacker	2020-08 2020-09	2020-10
D13 M	D13	M	Stone Crusher	2020-08 2020-09	2020-10
D13 K	D13	K	Reach Stacker	2020-06 2020-07	2020-08

After discussions with people from the field test department and the data collection unit at Volvo Penta, there has been a selection of relevant signals that in the clearest and most commonly used way represent an engine drive cycle. The selected signals are *Engine Speed*, *SCR Temperature*, and *Engine Torque*. These are illustrated in the thesis as in Figure 3.2.

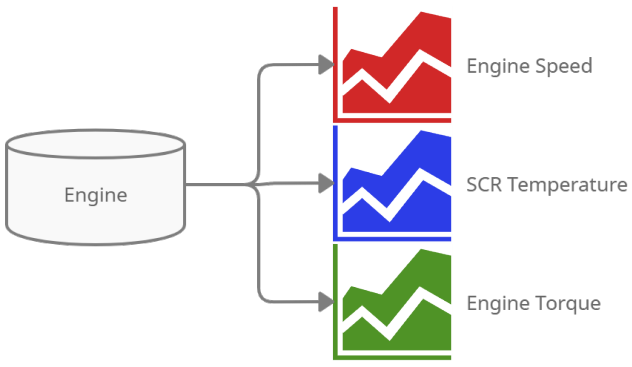


Figure 3.2: Engine signals selection workflow.

The data needs to be prepared for analytical reasons, and the purpose of the preparation is due to existing inconsistencies between different sensor readings. The sampling frequency for the sensors differs in the data collection phase, which is determined by the the *Engine Management System* (EMS) CAN-bus. E.g., the engine speed and engine torque are sampled with 10 $[Hz]$ while the SCR temperature is sampled with 40 $[Hz]$. The EMS is accessed through a field test logging device called ULTRA. The solution for this difference in sampling rate is to resample the

SCR temperature data to 10 [Hz].

The preprocessing and analysis part of the project consists of time series analysis which was based on understanding characteristics of the signals in order to distinguish patterns between the features. Initially, the process of understanding the signals began with observing daily and hourly time horizons. As the signals are logged sequentially, the initial work started with concatenating all the measurements of a month into a sequential time series for each signal. Having data with a time resolution less than one hour, may lead to inaccuracies within the training or validation data. However, this aspect is not investigated further in this thesis.

The analysis of the data was based on features included in the statistical, temporal and spectral domain within the TSFEL package that consisted of more than 300 features. The majority of the features were deemed redundant and eventually 12 features were selected for extraction. These 12 features, see Table 2.1, were chosen based on current knowledge of the features.

3.1.2 Feature Extraction

The data analysis defined the relevant features which are inputs to the ML algorithm. The definition of how relevant the features are in an ANN is based on a feature importance metric, which is evaluated with a technique called *integrated gradients*. The feature importance metric has a value, called significance score, which shows how much the input features activate the nodes in the ANN in the process of classifying the engines.

The features are not excluded to be a single feature for the ML model, but consists of the 12 features, presented in Table 2.1, that are representative of the time series. As the time series varies for each engine and the feature distribution is different depending on the drive cycle, the integrated gradients is a tool to intuitively analyze the ANN after the training. Therefore, as the features are relatively few, from a computational point of view, the need to reduce the amount of features before training is not interesting. Figure 3.3 presents a workflow of obtaining the features that represents a drive cycle and creating a dataset.

The feature extraction is built as a vector for the corresponding label (class) and the dataset is created in a sliding window manner where the features are consecutively extracted for a whole month and appended to a dataframe. The data was arranged such that every feature vector has a class correspondence, and ultimately, the dataset was shuffled randomly and divided into a train and a test set of 75 % and 25 %, respectively. Before using the feature vector as an input to the ANN, the vector is min-max normalized to values between 0-1 to suppress features that have large numerical values in relation to other features. The purpose of the normalization is a practical necessity to reduce the convergence time during the SGD optimization [18].

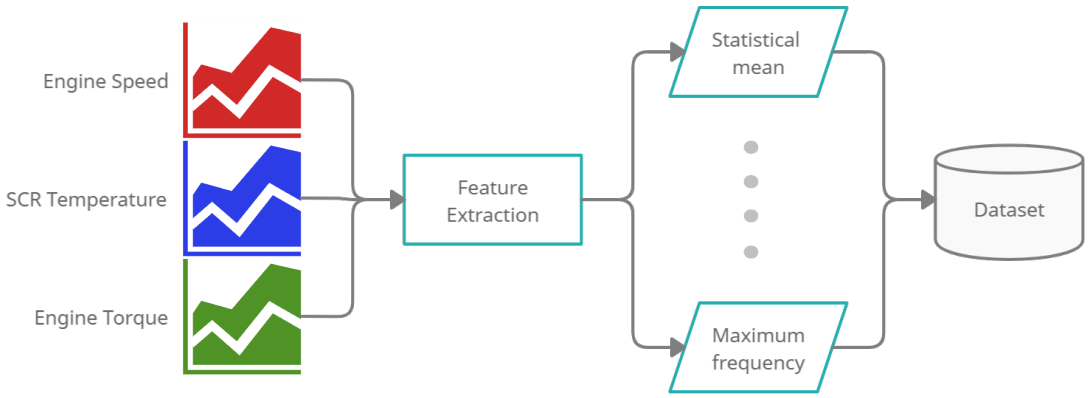


Figure 3.3: Feature extraction from the selected signals to a dataset.

3.2 Proposed Approaches Using Machine Learning

Machine learning is the concept of training a model with historical data where the intention of the ML model is to map a set of features to a class. The data that has been presented in Table 3.1 as the train/test set, will define the ML mapping behavior and is the most important aspect when training a ML model.

To be able to classify engine drive cycles, algorithms are needed to model historical data to be able to make predictions. The model must avoid to be biased towards a certain historical data type. However, it must be accurate enough to be able to separate one class from another. The proposal will be to implement an ANN with multilayer perceptron. The structure of the ANN is determined by an iterative testing process to be able to obtain high accuracy of the predicted labels.

This section presents three methods that utilize the ANN for different purposes. Section 3.2.1 presents a method of detecting drive cycles, Section 3.2.2 presents a way of annotating and semantically classifying segments of the data. Lastly, Section 3.2.3 presents a brief application of the ANN as a fitness function in the genetic algorithm for drive cycle generation.

3.2.1 Drive Cycle Detection

The method of detecting engine drive cycles is the fundamental way of evaluating if the features extracted from the signals are separable or not. As the data initially does not have detailed non-deterministic annotations, the choice of labels is based on the class names.

The labels correspond to the classes and are chosen as the class names in Table 3.1: *D11 T*, *D13 T*, *D13 M*, *D13 K*. Figure 3.4 presents the classification process and how to utilize the ANN.

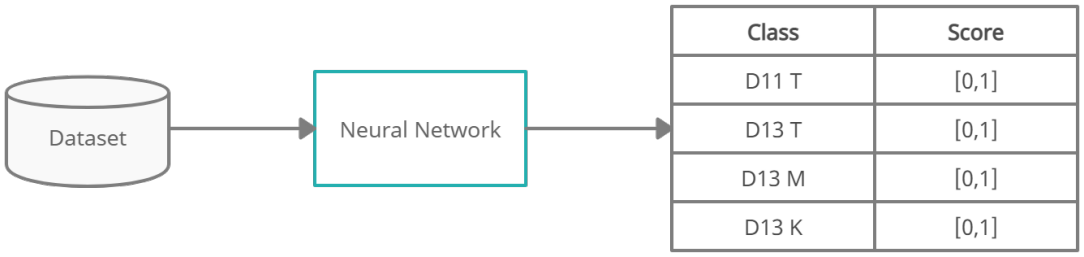


Figure 3.4: Example of drive cycle detection process.

The ANN was designed in a way that it has the ability to give the score of the predicted class as an output. A higher predicted score for a certain class implies that the features that have been extracted from an arbitrary time series, and used as input to the ANN, correspond to that class. The last output layer of the ANN is a sigmoid function which enables the score to correspond to the class that has similar features as the training data. The encoding of the engines in terms of the ML network is presented in Table 3.2, where the one-hot encoding represents a binary value of 0 or 1.

Table 3.2: One-hot encoding of the four classes for detection.

Class	Encoding
D11 T	[1 0 0 0]
D13 T	[0 1 0 0]
D13 M	[0 0 1 0]
D13 K	[0 0 0 1]

The ANN structure is illustrated in Figure 3.5 and a detailed specification is presented in Table 3.3. The network structure was obtained by an iterative testing process with the intention to minimize the loss and maximize the accuracy.

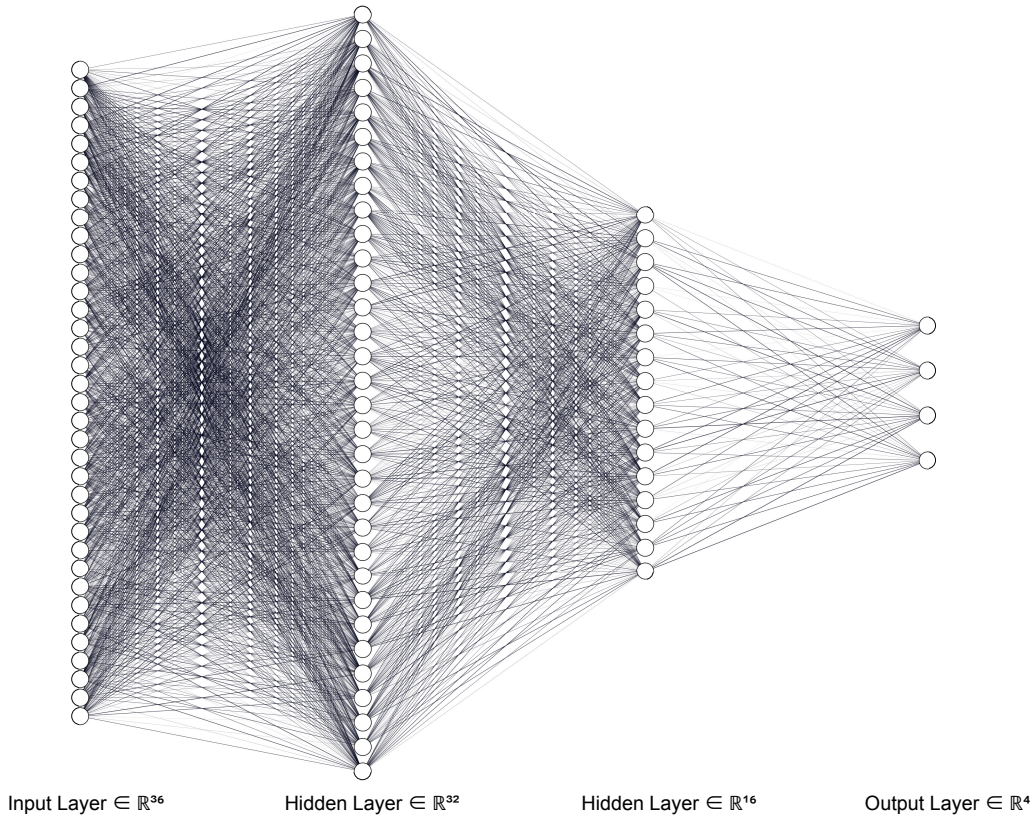


Figure 3.5: Illustration of the ANN structure for engine drive cycle detection.

Table 3.3: The detailed ANN structure for engine drive cycle detection.

Layer (type)	Input Shape	Output Shape
FCL	**[36,1]**	**[32,1]**
ReLU	[32,1]	[32,1]
FCL	**[32,1]**	**[16,1]**
ReLU	[16,1]	[16,1]
FCL	**[16,1]**	**[4,1]**
Sigmoid	[4,1]	[4,1]

The ANN is trained with a subset of historical data and tested on a different subset of this data which is an important choice in the data selection process. The method uses two months of sequential data as training and testing to ensure that the network model has the ability to distinguish the different engines. Since the last layer of the ANN is a FCL with a sigmoid activation function, the output of the ANN corresponds to a score. In this structure, the ANN is applied to a third month of data, and lastly extracts five one-hour drive cycles that the network considers to be the highest scoring cycles. These drive cycles are interpreted as the most accurately represented general engine drive cycles according to the ANN. This is done for each engine.

The choice of the historical data is a vital parameter that defines how the ANN

will evaluate the validation set. We did not focus on how the engine behaves in these two training and testing months in this method. The goal of the drive cycle detection is to screen either one or several months of data and propose a set of typical drive cycles for an engine application.

3.2.2 Semantic Segmentation of Engine States

Semantic segmentation of engine states is a method of classifying parts (segments) of the data, for univariate or multivariate time series. The semantic segmentation can be useful when classifying predefined characteristics based on historical data. For this purpose, two signals were selected for the study: *engine speed* and *engine torque*. The SCR temperature is neglected in this case because the annotation process did not include this signal for labelling the segments, and therefore, it was not included as an input feature.

The states that have been defined as the new class representations are shown as a one-hot encoding in Table 3.4. The semantic segmentation method is focused on the *D13 M* data because the states for the engine speed data are easy to annotate. Table 3.5 presents the data used for semantic segmentation.

Table 3.4: One-hot encoding of three classes for the semantic segmentation of engine states.

Class Name	**Encoding**
Running	[1 0 0]
Working	[0 1 0]
Idle	[0 0 1]

Table 3.5: The data for the semantic segmentation of *D13 M*. The train/test month is manually annotated while the validation month is not related to the annotation and ANN training process.

Name	**Engine Size**	**Pseudonym**	**Application**	**Train/Test Set**	**Validation Set**
D13 M	D13	M	Stone Crusher	2020-10	2020-07

A similar ANN to the one that is defined in Section 3.2.1 is used with the same purpose for semantic segmentation, but adapted to the correct amount of inputs and outputs as these are modified. The new ANN structure is presented in Table 3.6. Note that the reason for changing the structure of the ANN is the absence of the SCR temperature signal, and thus, reduced number of input features from 36 to 24. In addition to the reduced number of inputs, the outputs are reduced from 4 classes to 3 to correspond to the class names.

Table 3.6: The detailed ANN structure for semantic segmentation.

Layer (type)	Input Shape	Output Shape
FCL	**[24,1]**	**[32,1]**
ReLU	[32,1]	[32,1]
FCL	**[32,1]**	**[16,1]**
ReLU	[16,1]	[16,1]
FCL	**[16,1]**	**[3,1]**
Sigmoid	[3,1]	[3,1]

The annotated segments in the time series are shortened to a span of 3000 samples, which corresponds to 5 minutes, as the dynamics of the states fluctuates between running, working and idle more frequently than the detection method. The time span may even be shortened depending on the performance of the classification, but the time resources to manually annotate the data was limited. The signals from which the features have been extracted are limited to the *engine speed* and *engine torque* for the simplicity of annotating and analyzing the segmentation. In the validation phase of the segmentation process, there is a degree of freedom for decreasing the segmentation length, i.e. t_s, when classifying each segment. Three values of t_s are presented in Subsection 4.2.2.

3.2.3 Drive Cycle Generation

In this thesis, Volvo Penta has requested to be able to generate a time series representing an accurate engine drive cycle. There are multiple ways for time series generation, but this thesis will cover an implementation which originates from *evolutionary algorithms*.

Genetic Algorithm is used in this thesis to optimize and generate a time series that most accurately, based on the ANN, can describe an engine drive cycle. This type of algorithm generates a population of randomized vectors which will be altered through each iteration of the process. Mainly, each so called generation of vectors is altered, combined or mutated to achieve a better fitness score. The fitness score is obtained as a score output from the ANN for the specified class. The generation with the highest fitness score provides the best possible solution, in this case a vector, from the population, which represents an engine drive cycle. Figure 3.6 illustrates the genetic algorithm and its procedure. Table 3.7 presents the new ANN structure.

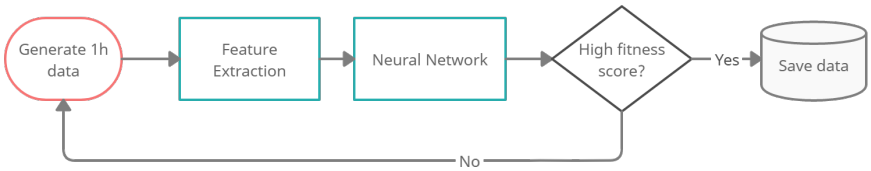


Figure 3.6: Genetic algorithm workflow to obtain a generated 1 hour drive cycle.

Table 3.7: The detailed ANN structure for time series generation.

Layer (type)	**Input Shape**	**Output Shape**
FCL	**[12,1]**	**[32,1]**
ReLU	[32,1]	[32,1]
FCL	**[32,1]**	**[16,1]**
ReLU	[16,1]	[16,1]
FCL	**[16,1]**	**[4,1]**
Sigmoid	[4,1]	[4,1]

4
Results & Discussion

This chapter is divided into three main sections. These sections are: *Drive Cycle Detection*, *Semantic Segmentation of Engine States*, and *Drive Cycle Generation*. Each section will present the data distribution and ANN performance followed by the predicted output and feature importance for the classes. Ultimately, the result and evaluation of each method is presented, followed by discussion regarding the results.

4.1 Drive Cycle Detection

The dataset distribution of data points for each class is presented in Figure 4.1 where it is clear that *D11 T* and *D13 T* have a similar amount of data points, opposed to *D13 M* and *D13 K* which have significantly fewer data points. Hence, the dataset is unbalanced.

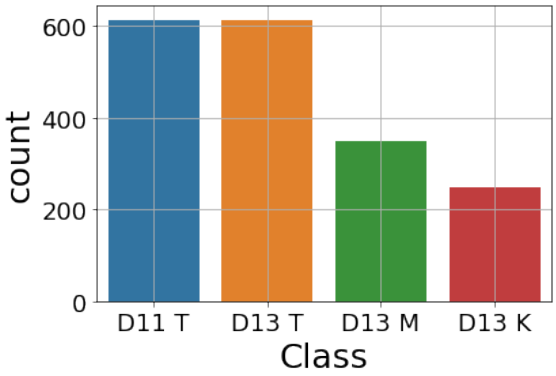


Figure 4.1: Visualization of the data distribution for each class. Each data point represents a set of features that have been extracted from segments of 1 hour from the signals.

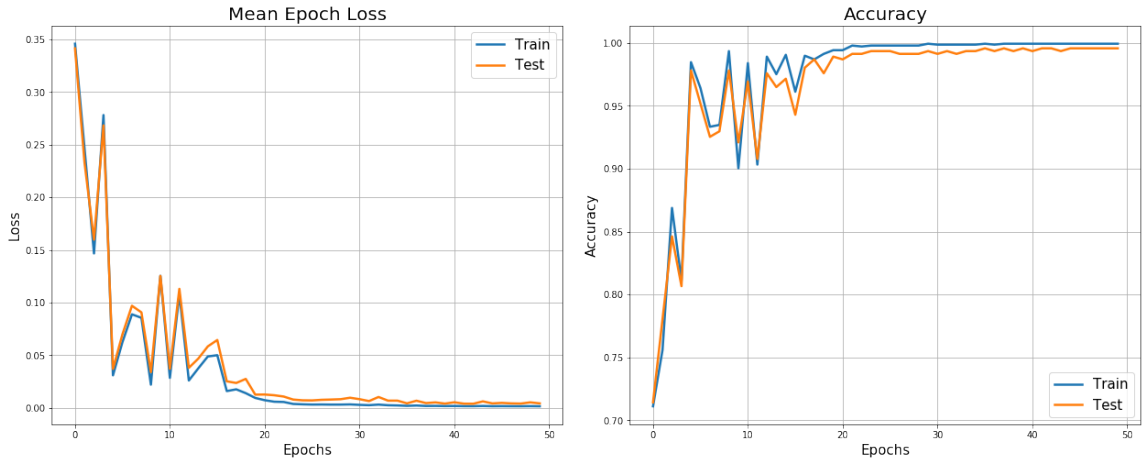
4.1.1 ANN Performance

The structure of the ANN can be seen in Table 3.3 from Chapter 3 in which we discussed the methods. In Table 4.1, the hyperparameters are listed which are the result of the ANN performance. These are tuned in an iterative process in order to maximize the accuracy and minimize the loss for both training and test datasets.

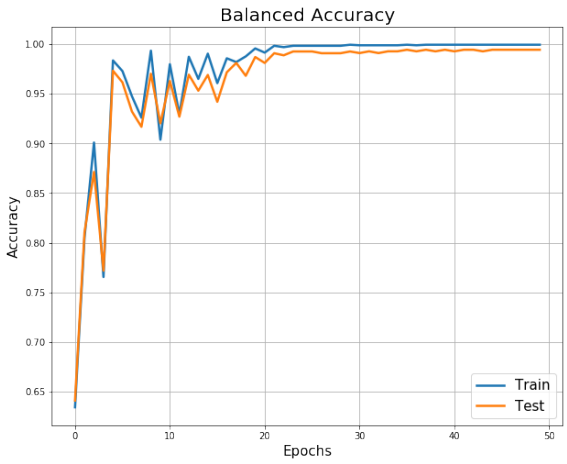
Table 4.1: Hyperparameters for the ANN – Detection method.

Hyperparameters	
Number of Epochs	50
μ - *Learning Rate*	0.1
ρ - *Momentum*	0.4

Figure 4.2 shows the resulting performance achieved from the ANN. The three plots illustrate the *Mean Epoch Loss*, *Accuracy*, and *Balanced Accuracy*, respectively. The results show that all three scores in Figure 4.2 behaves similarly. Although, during the first 20 epochs, the score fluctuates before stabilizing at each metric's optimum.



(a) Mean Epoch Loss **(b)** Accuracy



(c) Balanced Accuracy

Figure 4.2: ANN performance for the drive cycle detection training and testing.

The confusion matrix in Figure 4.3 shows the correct classifications on the diagonal, while the cases in the upper and lower triangular segments are the falsely classified

ones. These results are based on the test set that is fed into the ANN. As an example for the lower left corner in Figure 4.3, the ANN predicts the class as *D11 T* while the correct class is *D13 K*.

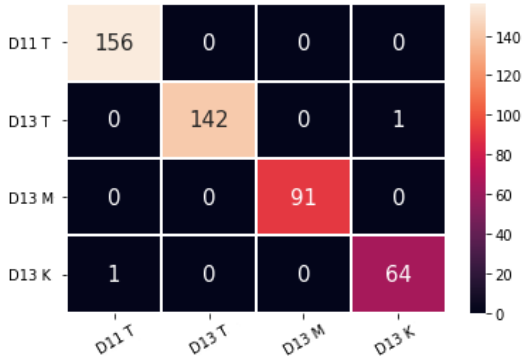


Figure 4.3: Confusion matrix of predictions for the test set.

The Figures 4.4-4.7 cover feature importance for each class, respectively, and in each figure, individual signals are presented. It shows how each feature affects the ANN decision making for each class. Figure 4.4 illustrates feature importance for the class *D11 T*, where the ANN decision making is highly dependent on both engine speed and SCR temperature, while the engine torque signal affects the decision negatively.

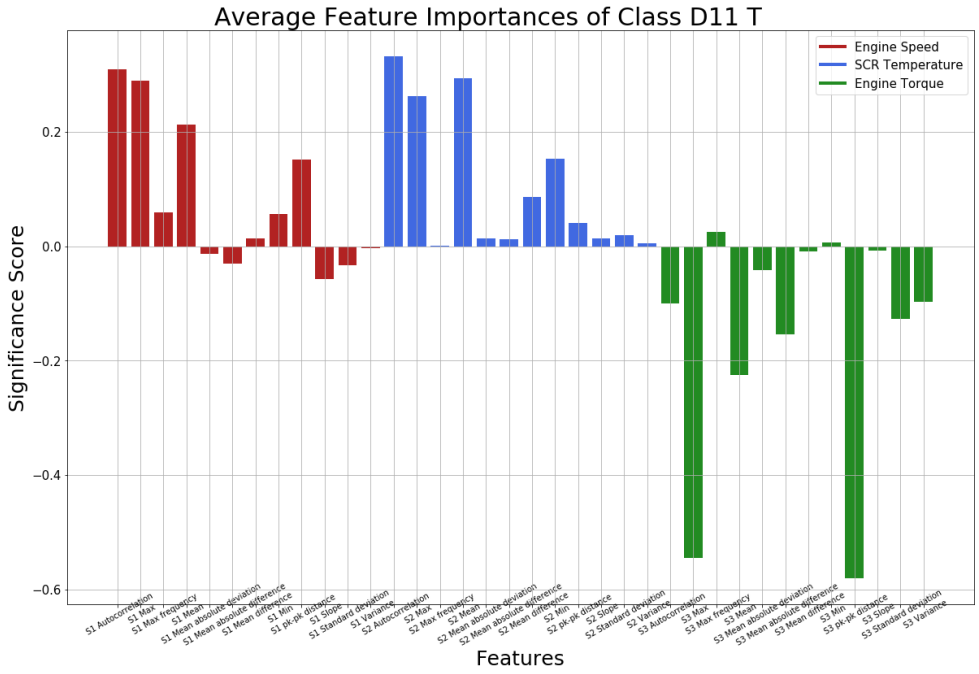


Figure 4.4: Feature importance for the class *D11 T*.

Feature importance for *D13 T* is shown in Figure 4.5 which presents almost an opposite result, compared to *D11 T*. Both engine speed and SCR temperature show mostly a negative impact on the ANN decision making, although it is relatively mild for the SCR temperature signal. Instead, these results show that feature importance depends heavily on the engine torque signal.

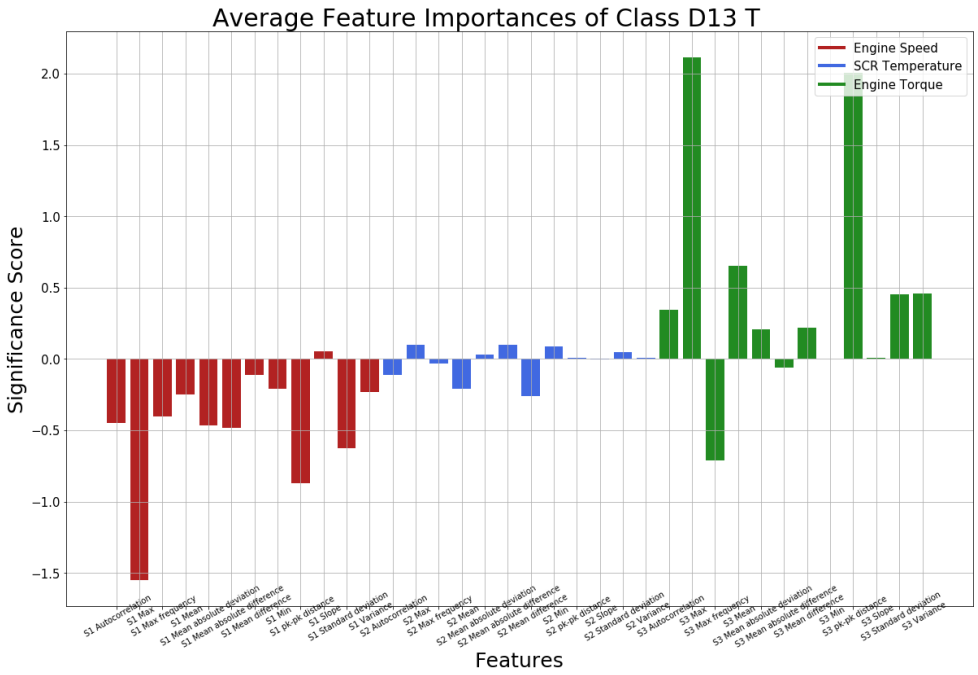


Figure 4.5: Feature importance for the class *D13 T*.

Results for the *D13 M*, seen in Figure 4.6, have an overall lower feature importance score compared to previous classes. The figure also shows that the engine speed signal has the most weighed features for the ANN to be able to make a correct classification. While engine torque also shows significant importance, the score is relatively low compared to the engine speed.

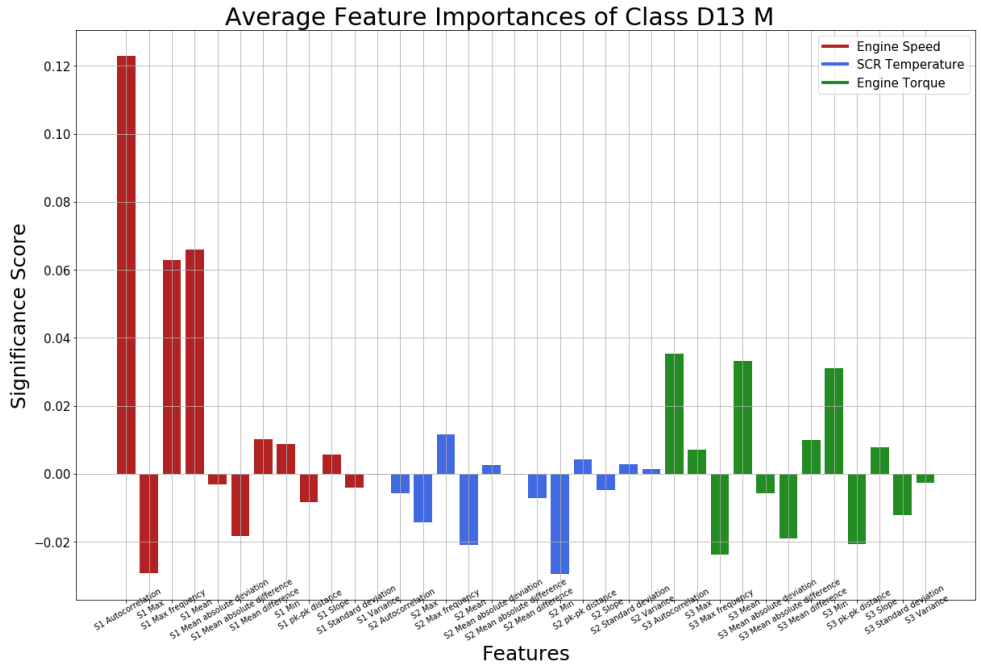


Figure 4.6: Feature importance for the class *D13 M*.

Lastly, *D13 K* relies on its features based on engine speed as seen in Figure 4.7. There are also some features that are important regarding the engine torque signal, e.g., *max frequency* and *mean absolute difference*. Although, both *max* and *peak-to-peak distance* negatively affect the ANN decision making.

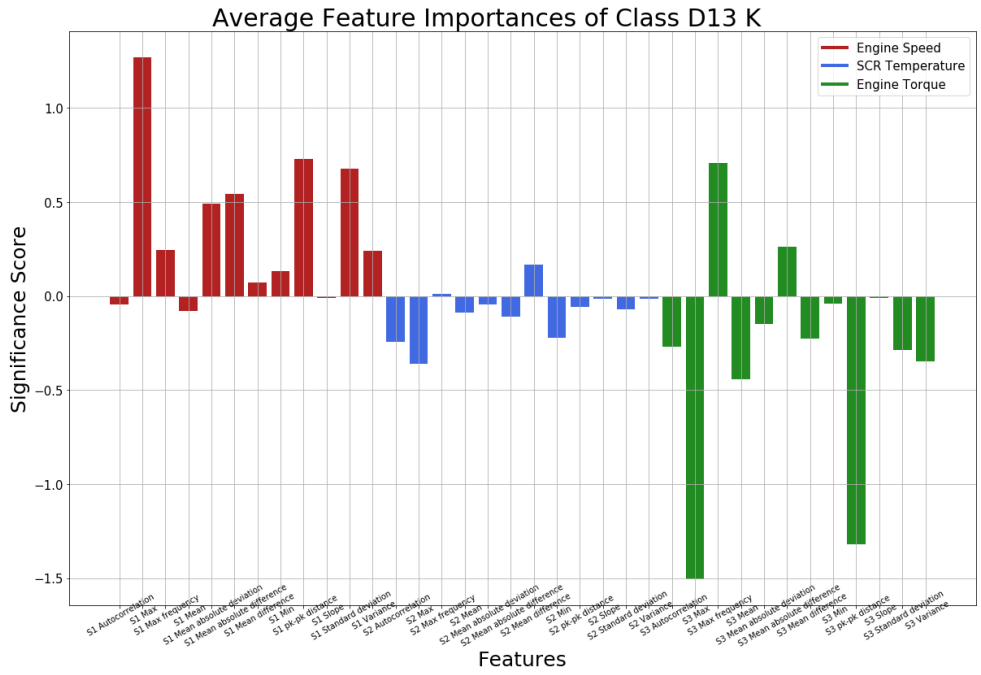


Figure 4.7: Feature importance for the class *D13 K*.

4.1.2 Ranking & Evaluation of Detected Cycles

To analyze the best type of general engine drive cycle in this thesis, five one-hour cycles with top score, determined by the ANN, are extracted. The top scoring engine drive cycles for *D11 T* can be seen in Figure 4.8.

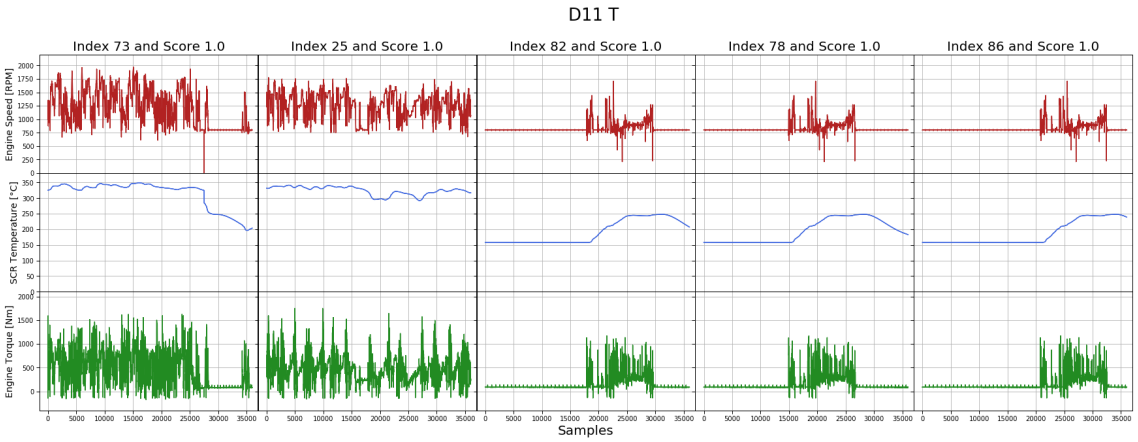


Figure 4.8: Top 5 scoring engine drive cycles for *D11 T*.

In Figure 4.8, each row describes one signal, and from top to bottom the signals are: Engine speed, SCR temperature, and Engine torque. Each column provides a one-hour drive cycle with each column title describing the index of the hour in the validation month, which can be seen in the Figures A.1-A.4 provided in appendix. The score is also provided for each column, and note that in all provided results the score is 1 which is the highest score that can be achieved by the ANN. In Figure 4.9, the top scoring cycles for *D13 T* are shown.

Figure 4.9: Top 5 scoring engine drive cycles for *D13 T*.

Note that *D11 T* and *D13 T* have somewhat different engine drive cycles, though with similar scores, scored by the ANN. Below in Figure 4.10, a completely different type of engine drive cycle and application is shown for the *D13 M*. Recall that compared with *D13 T*, which is a logstacker, *D13 M* is a stone crusher, hence it has a different type of engine drive cycle.

Figure 4.10: Top 5 scoring engine drive cycles for *D13 M*.

Lastly, Figure 4.11 is the *D13 K* which is a reach stacker, again with a completely different type of engine drive cycle.

Figure 4.11: Top 5 scoring engine drive cycles for *D13 K*.

Figure 4.12 shows, as an example, a pair plot of the features for class *D13 M* and how its most important features are scattered and distributed. The diagonal plots presents the distribution of the features in the form of histograms, while the upper and lower triangular presents the features plotted against each other. The most important features are selected based on the highest feature importance score given by the Figures 4.4-4.7. Additionally, the best and the worst scores of the classification of the validation set are presented in green and red respectively.

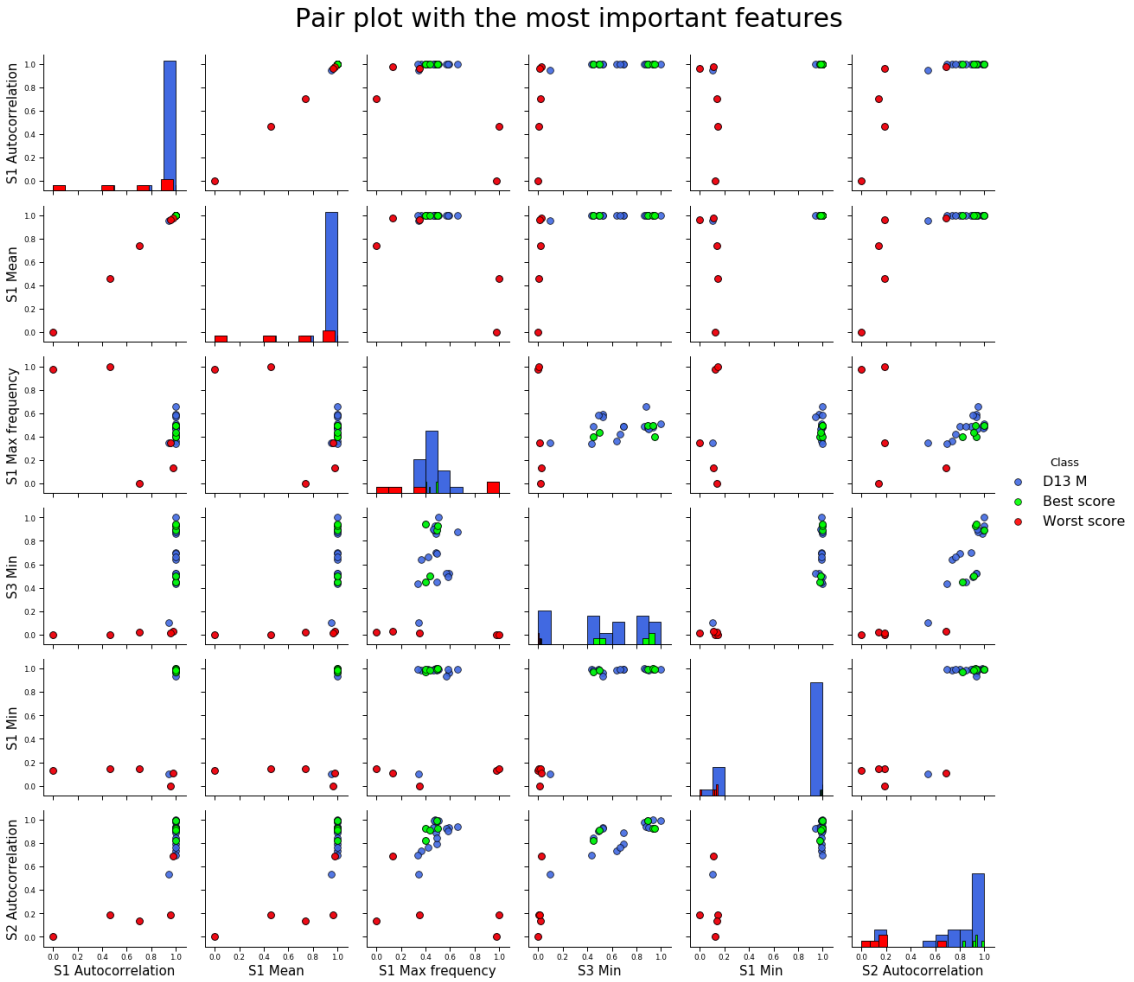


Figure 4.12: Scatter plot and the data distribution of the class *D13 M* together with the data points with best and worst scores. The features with the largest significance score is presented.

4.1.3 Discussion on Drive Cycle Detection

The dataset presented in Figure 4.1 shows that the data corresponding to each class is not equally distributed among all classes. The goal when selecting the months of the corresponding classes was to create a dataset where each class has a similar amount of data. Obtaining a balanced dataset was not possible due to the fact that the industrial machines had different distributions of uptime every month. The focus was rather to be consistent with the months that were extracted to avoid potential seasonal interference when using the SCR temperature. Therefore, we want to use months that are relatively close to each other. No studies about whether or not the season would interfere with the classification has been done since this is considered to be a part of future work for this thesis.

The parameters presented in Table 4.1 were tuned in order to minimize the loss and maximize the accuracy of the ANN based on the test dataset. Figures 4.2a-4.2c all present similar characteristics, such as fluctuations between 0-20 epochs and ac-

curacy convergence around 0.98, which indicates that the balanced accuracy metric is not needed. The fluctuations are results of a high value for the learning rate, μ. The mean epoch loss in Figure 4.2a converges towards 0 for the train and test dataset with small deviance between each other, which is preferred. The confusion matrix in Figure 4.3 presents the classifications, and what can be observed is that there are two misclassifications as *D13 K* was falsely predicted as *D11 T*, and *D13 T* was falsely predicted as *D13 K*. We cannot conclude whether the similarity of these two drive cycles are the factor for these misclassifications. However, a hypothesis for this specific case is that the data points are too scarce for the ANN to learn the distinctions. The results from the ANN performance are nevertheless promising as it concludes that the extracted features are separable enough between the classes to be able to distinguish different engine drive cycles by using a rather simple neural network.

Figures 4.4-4.7 present the feature importance by using the integrated gradients method based on the ANN. The high average significance score is noticeable for both the engine speed and the SCR temperature when classifying *D13 T*, in comparison to engine torque. This indicates that the features for engine speed and SCR temperature are good additions to the ANN when classifying *D13 T*. However, by looking at engine torque, which has negative significance scores, we see that it affects the ANN negatively and rather complicates the classification process. When we classify *D13 T*, the significance scores are high for the feature contributions of the engine torque. *D13 M* and *D13 K* share a somewhat similar feature importance distribution, but differ in absolute value of the significance score. *D13 M* has generally lower significance scores compared to the other classes, which might be explained by the score distributions depending on how certain the ANN is. The ANN creates a mapping which divides the classes by their clusters, which is illustrated in Figure 4.12. The data points that have the largest score output will not necessarily be the data points that are closest to the clusters, but rather inside the mapping done by the ANN. Further research regarding the score distribution among the classes is required to understand the mapping process.

The results from the ranking and evaluation of detected cycles Figures 4.8-4.11 present the top five highest scoring one-hour cycles considered by the ANN. These results may be a way for Volvo Penta to extract the top scoring cycles which then can be analyzed by experts to evaluate if any of the cycles are any good or representable for an application. Note that every score output from the ANN is highly dependent on how the cycle is represented in the training process. Therefore, a suggestion would be to briefly analyze the training data and remove obvious outliers. As an example of engine drive cycles that are representable, *D11 T* seen in Figure 4.8, shows that the cycles in index 73 and 25 both have cycles that look more realistic compared to the remaining three (index 82, 78 and 86). The later three hours of different drive cycles show a not so realistic scenario. Also, observing the full validation month presented in Figure A.1 the cycles are repeated for these three hours with a small displacement. This conclusion is derived by looking at the complete engine drive cycle month, shown in the Figures A.1-A.4, which show

that these scenarios are not so likely to occur during general operations for that application. This indicates that the training data included data points that do not represent a general engine drive cycle, which is expected since the training data was not screened. Therefore, we conclude that depending on what the ANN wants to detect, a selection of the training data that represents a typical engine drive cycle must be properly defined during the phase of the training dataset creation.

Lastly in Figure 4.12, a pair plot is shown as an example to compare a class, in this case *D13 M*, towards the features that the ANN classifies as both the best and worst scores. The best scores have the largest score output from the ANN, opposed to the worst scores which have the lowest score output from the ANN. We can observe that the data points with the best scores tends to be closer to the majority of the data points for the class *D13 M*, while the worst scores are further away. In this particular case, we can conclude that the ANN has been trained on features that corresponds to where the majority of the data points are located. The worst scoring data points is a result of the lack of feature representations in the training dataset.

4.2 Semantic Segmentation of Engine States

The dataset distribution of data points for each class is presented in Figure 4.13. This dataset is imbalanced where the *Working* class has the majority of data points while *Running* and *Idle* have significantly fewer.

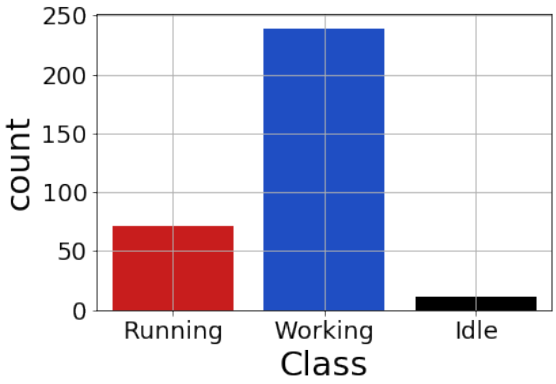


Figure 4.13: Visualization of the class distribution for the data included in the dataset. Each data point represents a set of features that has been extracted from segments of 5 minutes from the signals.

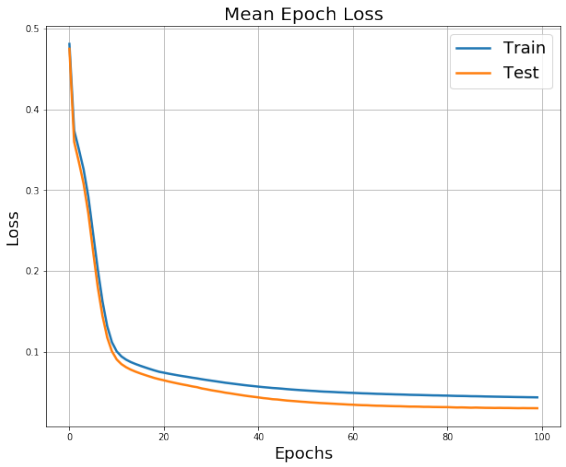
4.2.1 ANN Performance

The structure of the ANN can be seen in Table 3.6 from Chapter 3 in which we discuss the method. In Table 4.2, the hyperparameters are listed which is the result of the ANN's performance during training. These are tuned in an iterative process in order to maximize the balanced accuracy and minimize the loss for both the training and test dataset.

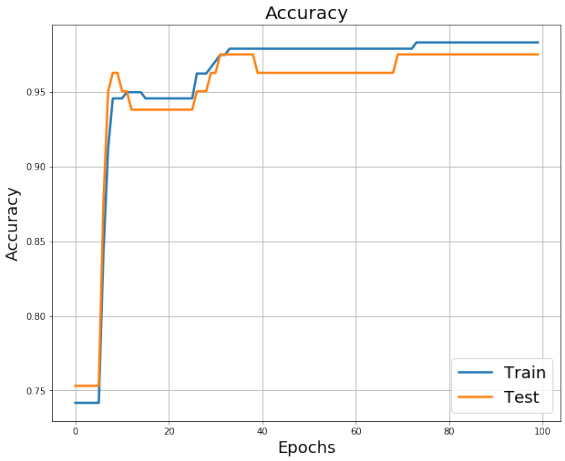
Table 4.2: Hyperparameters for the ANN – Semantic Segmentation.

Hyperparameters	
Number of Epochs	100
μ - *Learning Rate*	0.01
ρ - *Momentum*	0.4

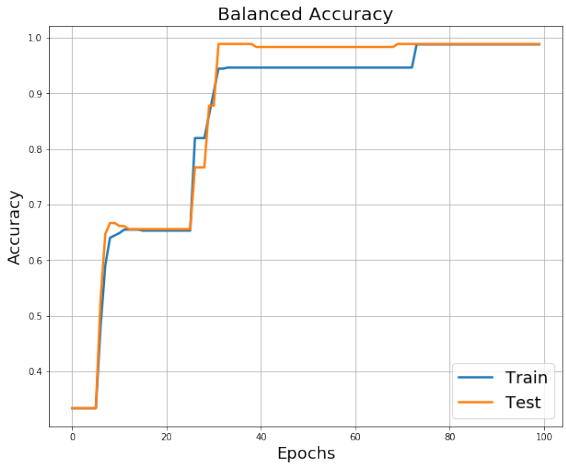
The mean epoch loss, accuracy and the balanced accuracy are presented in Figure 4.14. The most relevant metrics to observe, as the dataset is imbalanced, are the mean epoch loss, and the balanced accuracy. The mean epoch loss converges for both the train and the test set with minor differences. The balanced accuracy for the train and test maintain a close relationship through the epochs. Figure 4.14b in comparison to Figure 4.14c shows a higher accuracy around epoch 10-30 due to the imbalance of the dataset.



(a) Mean Epoch Loss



(b) Accuracy



(c) Balanced Accuracy

Figure 4.14: ANN performance for the semantic segmentation training and testing.

The result of the classifications can be observed in Figure 4.15, where two instances of the class *Working* was mispredicted as *Running*.

Figure 4.15: Confusion matrix of predictions for the test set.

The following figures present the average feature importance during the prediction stage of the ANN, for each feature. The feature importance is evaluated using

the test set, and with higher significance score, the features will contribute more to the classification of the evaluated class. Observing Figure 4.16, the features *S1 Autocorrelation* and *S1 Mean* have the largest significance score on the classification of class *Running*. Figure 4.17 on the other hand shows that the features *S3 Mean* and *S3 Autocorrelation* have the largest significance score on the classification of class *Working*. Lastly, observing Figure 4.18, the features *S3 Max Frequency*, *S3 Slope* and *S3 Mean Difference* have the largest significance score on the classification of class *Idle*.

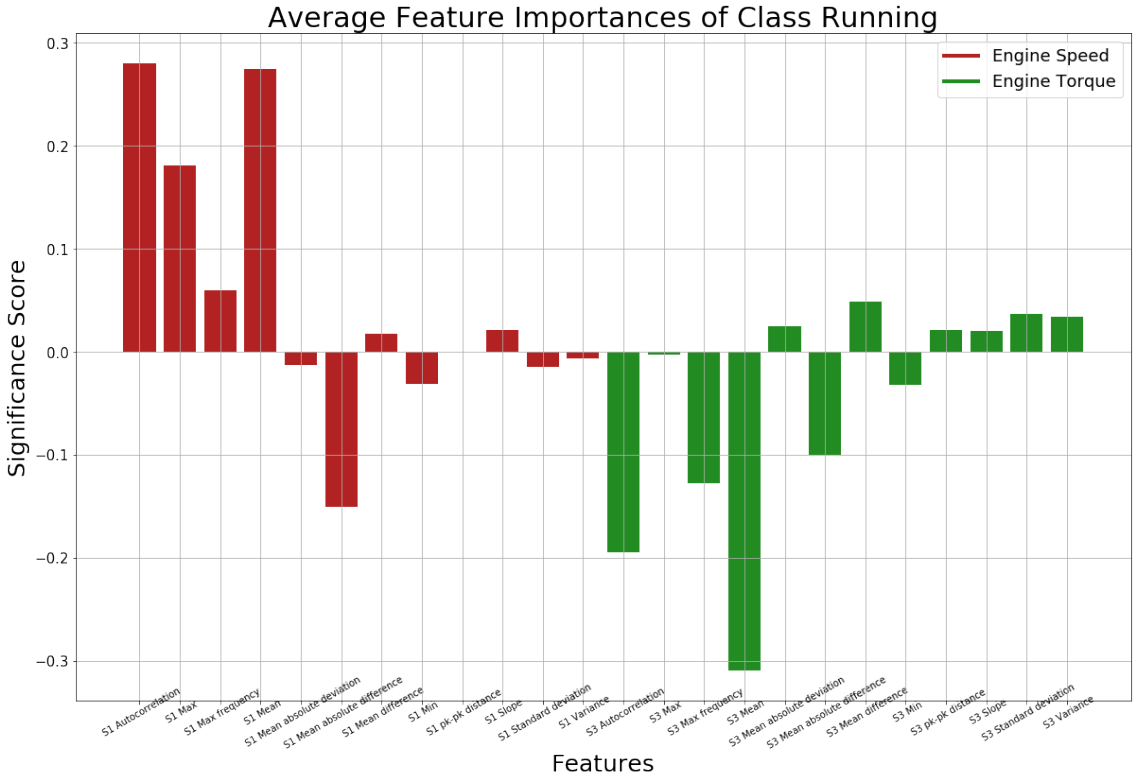


Figure 4.16: Feature importance for class: Running

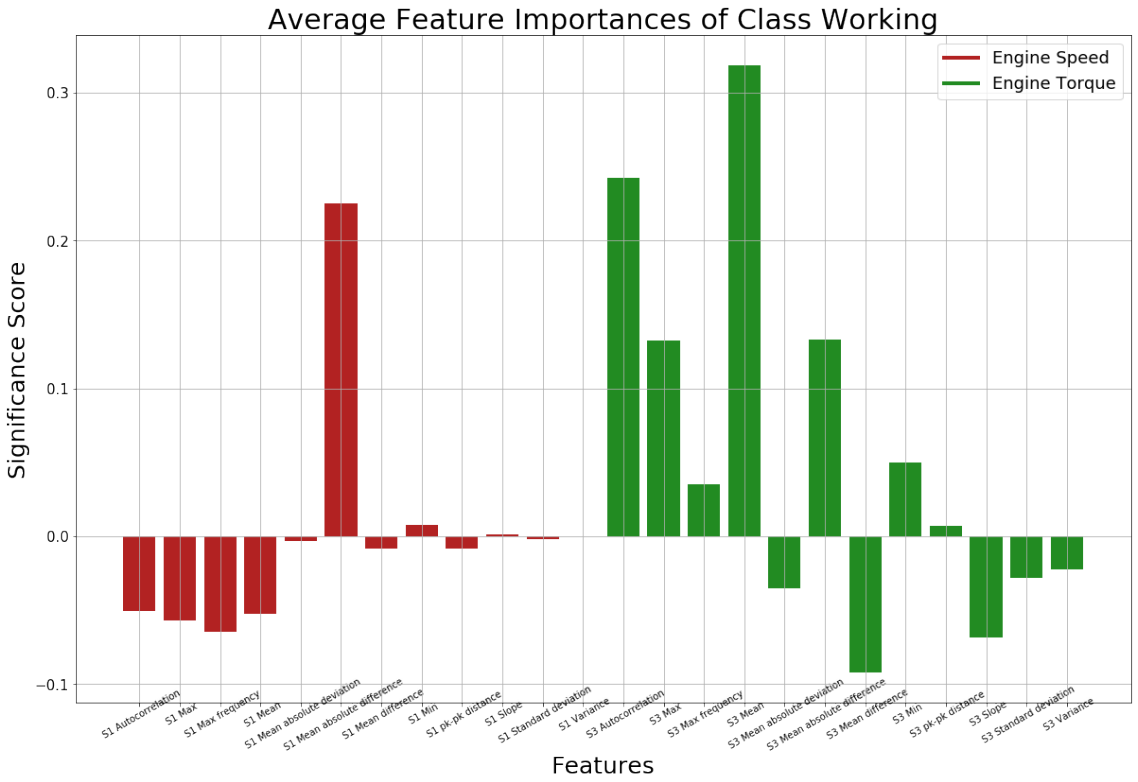


Figure 4.17: Feature importance for class: Working

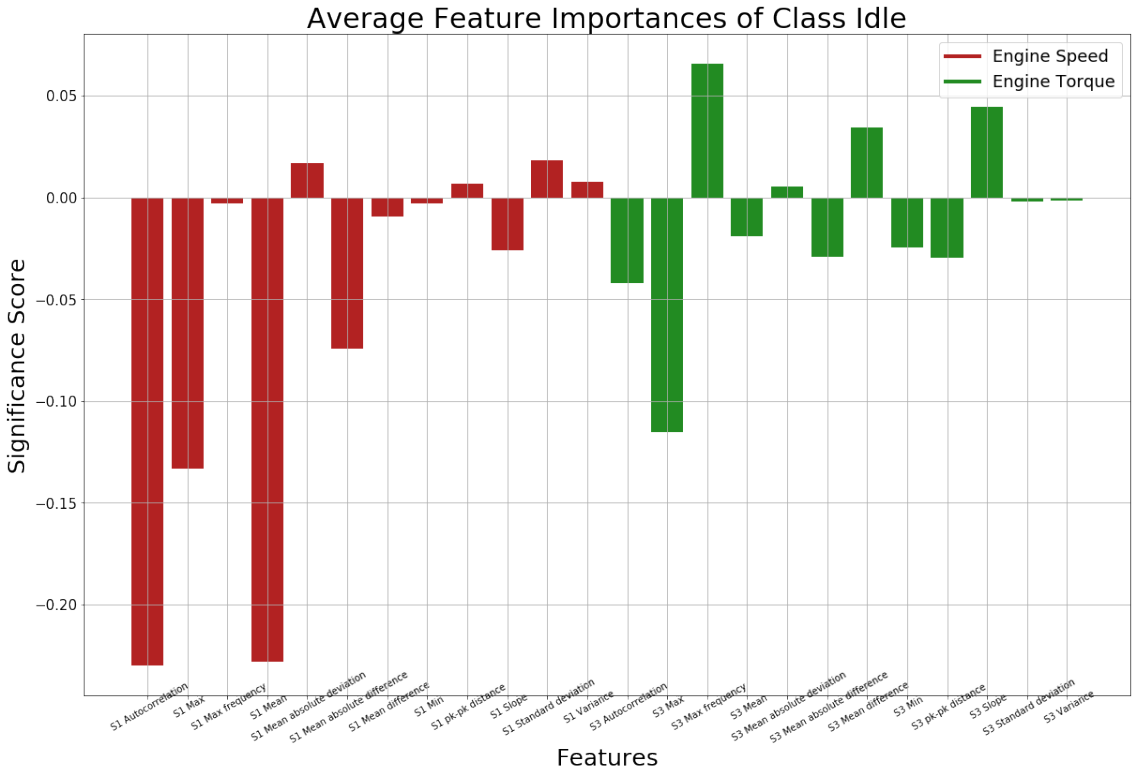
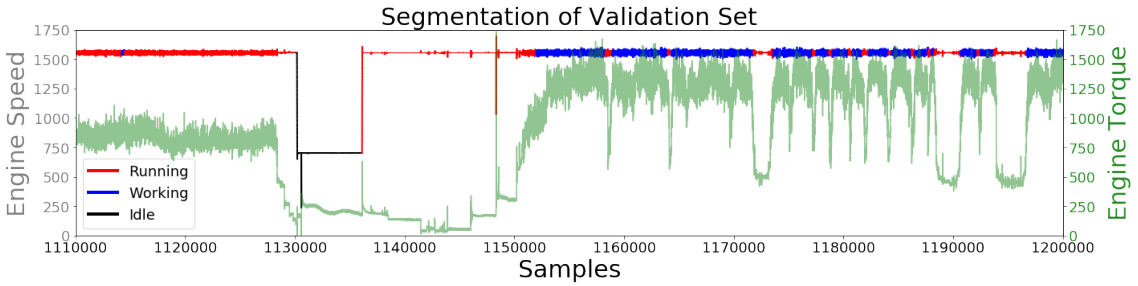


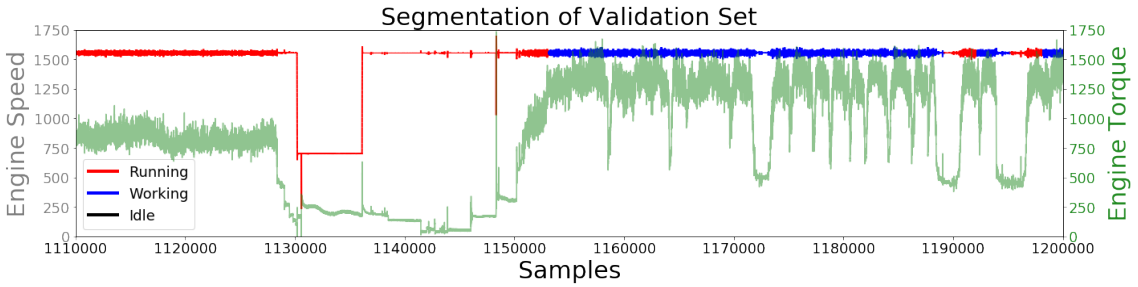
Figure 4.18: Feature importance for class: Idle

4.2.2 Semantic Segmentation Validation

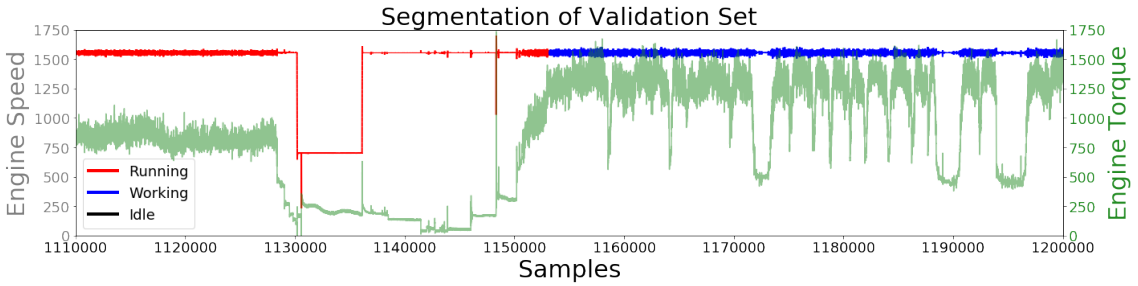
The Figures 4.19a, 4.19b and 4.19c present the contiguous application of the ANN with different segmentation lengths, i.e., t_s of 300, 3000 and 5000 samples corresponding to 0.5, 5 and 8.33 minutes. The plots have two axes that present the *engine speed* (left y-axis) and *engine torque* (right y-axis). A zoomed in part of the validation month of 90000 samples, which corresponds to 150 minutes, is presented to be able gain a clear view of the segmentation. The segments of the engine speed that are red, blue and black correspond to the classes *running*, *working*, and *idle*, respectively.



(a) The segments are 300 samples each, which corresponds to 0.5 minutes.



(b) The segments are 3000 samples each, which corresponds to 5 minutes. These segments has the same segmentation length as the train/test set.



(c) The segments are 5000 samples each, which corresponds to 8.33 minutes

Figure 4.19: Semantic segmentation of the validation month with three values of the segmentation length t_s.

From the feature importances presented in Subsection 4.2.1, two features have been chosen based on the largest significance score of each of the classes and can be observed in Figure 4.20. The scatter plots show the clustering of the features against each single feature, while the histogram presents the distribution of the individual features.

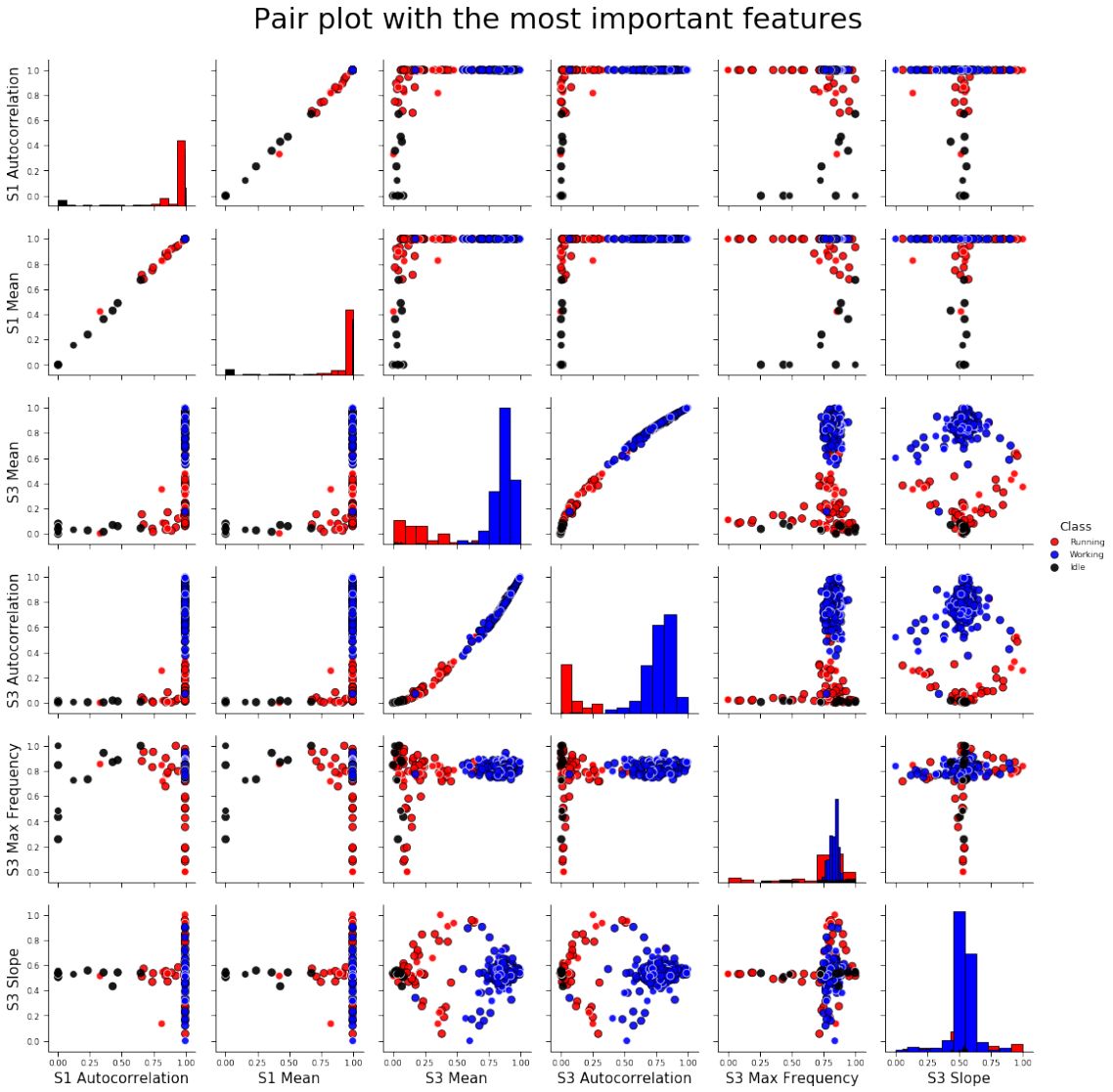


Figure 4.20: The data distribution scatter plot for each class. Two features from each class with the largest significance score is presented.

4.2.3 Discussion on Semantic Segmentation of Engine States

The dataset presented in Figure 4.13 shows that the data corresponding to each class is not equally distributed among the classes and has a large class imbalance. The reason for this is that the annotated month did not have an equal distribution of engine states for each class. This results in a skewed, or imbalanced, dataset that may affect the classification negatively on those states that has the lowest amount of data representation. The reason is that the ANN does not have the ability to learn as many feature representations for the class with the lowest count of data points.

When comparing the results presented in Figure 4.14b and 4.14c, there is a noticeable deviation around epochs 6-25 that shows a larger accuracy, but a lower balanced accuracy. The explanation is related to the imbalanced dataset, as this skewness of training is reported by the balanced accuracy, opposed to the standard accuracy. The limited amount of datapoints for the class *idle* is also an indication for this result. This motivates imbalanced datasets to include imbalanced accuracy metrics to be able to correctly evaluate an ANN. Ultimately, presented by the confusion matrix in Figure 4.15, there are a low percentage of misclassifications of the test set where only two labels were misclassified. This concludes that the ANN is able to separate the classes successfully based on the test dataset.

The Figures 4.16-4.18 show that a large amount of the 12 features for each signal have a significance score close to 0 which implies that these features would be safe to remove from the feature extraction, while retaining the high metric reports. These results only imply the feature importance, and need to be investigated further to be able to draw a definite conclusion regarding the removal of features.

The Figure 4.19 presents the semantic segmentation of a zoomed in part of the validation month. The validation month is unrelated to the train and test set and its purpose is to present how the ANN classifies each segment. The segmentation lengths are chosen as three different time spans to show that the method is not excluded to a single segmentation length. As the annotations were labeled with segmentation lengths of $t_s = 3000$, the best representation is to also validate with this segmentation length. Meanwhile, lowering t_s to 300 samples, there is a noticeable difference at the classification between running and working from around 1160000 samples and forward. The difference is that shorter segments where the state is running are being classified correctly by the ANN as running, compared to larger segmentation lengths where these shorter segments of running states are neglected. The *Idle* segments at around 1130000 samples were not classified correctly for the larger values of t_s. The explanation for this lies within the historical data and how the data is annotated, as the training set did not include enough *Idle* data points to represent the *Idle* states at that segmentation length. More data representations of the mispredicted class is also important when fitting a model to a neural network.

Figure 4.20 presents a set of pair plots along with the distribution for six features. Two features with the highest significance score for each class is chosen with the intention to illustrate the clustering of the classes. Generally, there are feature com-

binations that create clusters of the classes, such as the *S3 Mean* in combination with *S1 Max Frequency*. This concludes the importance of annotating segments of the data that are easily separable by the ANN in the training process. When annotating the segments, a necessity to create an accurate ANN that models the features to the classes, is to more accurately annotate the segments. By more accurately annotate the segments, we refer to the importance of understanding the dynamics of an engine and its application and annotate the segments accordingly.

This semantic segmentation method has shown that engine states can successfully be semantically classified based on the test class *D13 M*. The segment length was proved to have a large degree of freedom when segmenting the states correctly. This proves that depending on the engine state and how the dynamics behave, a shorter segment helps the classification. Further research regarding how the complexity of the engine states affects the classification must be conducted in order to validate the robustness of the ANN.

4.3 Drive Cycle Generation

The dataset distribution of data points for each class is presented in Figure 4.21, which is the same dataset as the engine drive cycle detection. The difference between the methods is that only the engine speed features are extracted and used as an input to the ANN for the generation.

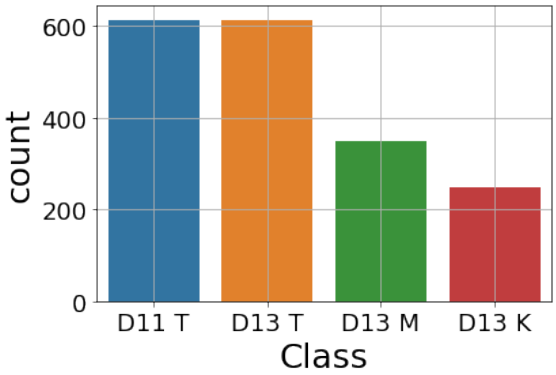


Figure 4.21: Visualization of the data distribution for each class. Each data point represents a set of features that has been extracted from segments of 1 hour from the signals.

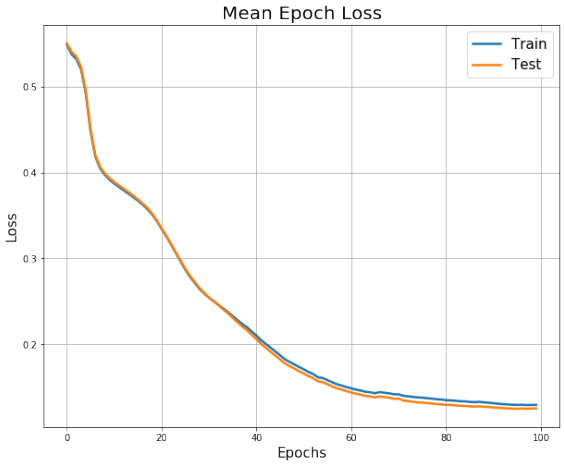
4.3.1 ANN Performance

The structure of the ANN can be seen in Table 3.7. In Table 4.3 the hyperparameters are listed which is the result based on the ANN's performance during training. These are tuned in an iterative process in order to maximize the accuracy and minimize the loss for both the training and test dataset. The learning rate, μ, is lower than the previous methods.

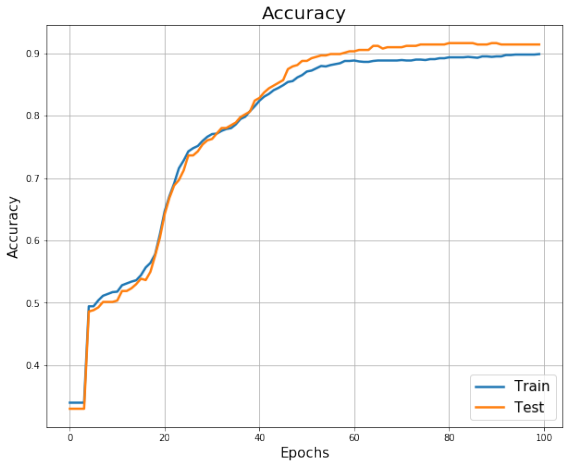
Table 4.3: Hyperparameters for the ANN – Drive cycle generation.

Hyperparameters	
Number of Epochs	100
μ - *Learning Rate*	0.005
ρ - *Momentum*	0.4

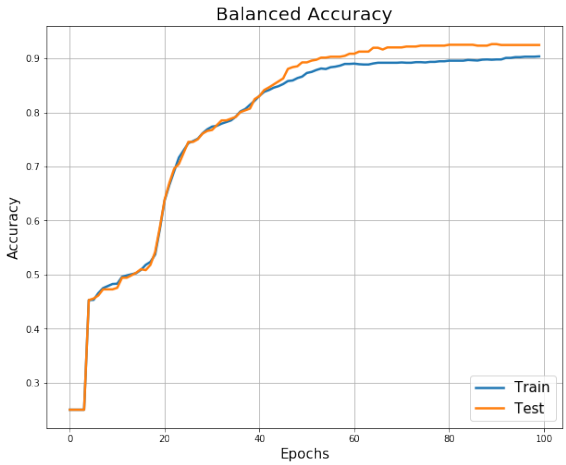
The mean epoch loss, accuracy, and balanced accuracy are presented in Figure 4.22. The mean epoch loss converges for both the train and the test sets with minor differences. The accuracy and balanced accuracy maintain a close relationship throughout all epochs. The general accuracy is lower than the previous methods, where the ANN utilized more features in the form of more signals, and ends up at around 0.9 accuracy.



(a) Mean Epoch Loss



(b) Accuracy



(c) Balanced Accuracy

Figure 4.22: ANN performance for the generation training and testing.

The result of the classifications can be observed in Figure 4.23, where the ANN is generally bad at separating class *D11 T* and *D13 T*. Note that these two classes have different engine sizes but have the same type of application, i.e, logstacker.

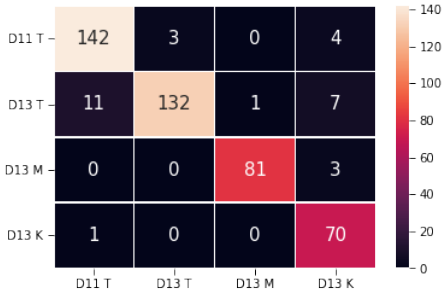


Figure 4.23: Confusion matrix of predictions for the test set.

The feature importance is presented in Figure 4.24-4.27 where the engine speed

features can be observed, as the generations goal is to generate an engine speed drive cycle of one hour. Three features with the highest significance score for class *D11 T* can be observed in Figure 4.24 as *S1 Max*, *S1 Pk-Pk Distance* and *S1 Autocorrelation*. For the class *D13 T* presented in Figure 4.24, the best are represented as *S1 Max*, *S1 Mean Absolute Deviation*, and *S1 Standard Deviation*. *D13 K* in Figure 4.26 has three remarkably higher scores compared to the other features significance score which are *S1 Autocorrelation*, *S1 Max frequency* and *S1 Slope*. Lastly, the class *D13 K* feature importance is presented in Figure 4.27 where the three features with the highest scores are the *S1 Max*, *S1 Max frequency* and *S1 Pk-Pk Distance*.

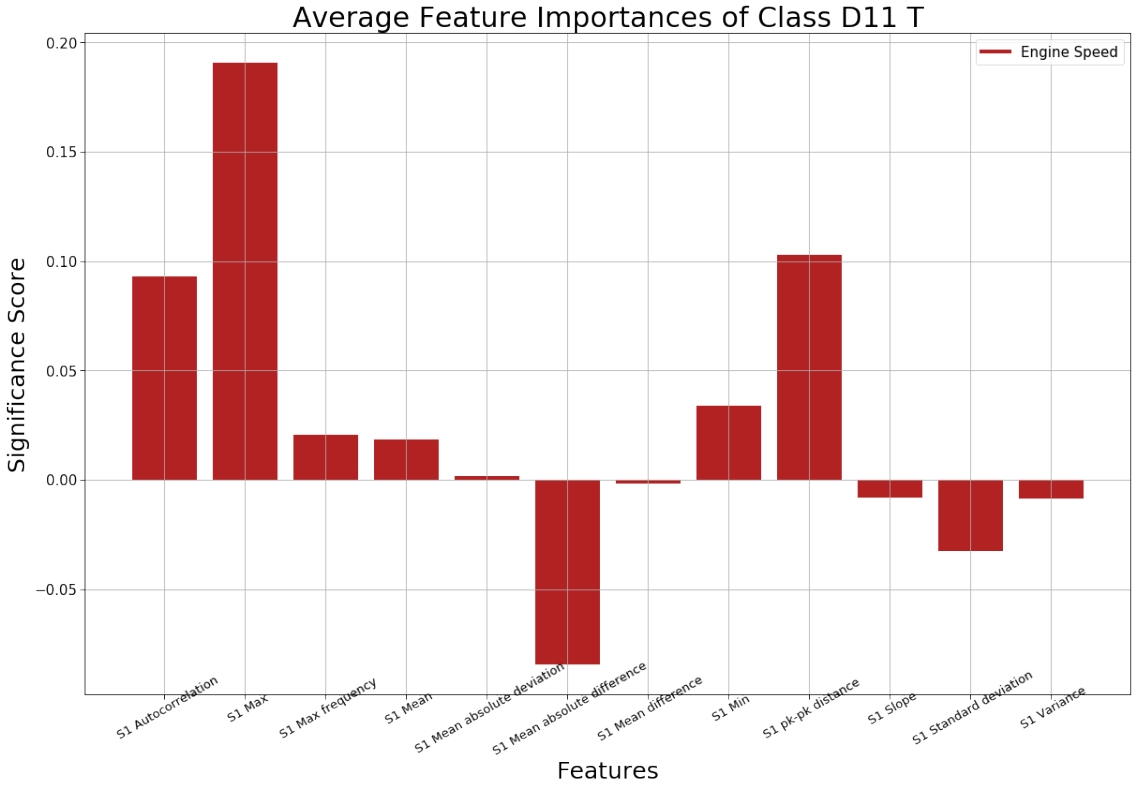


Figure 4.24: Feature importance for class: *D11 T*

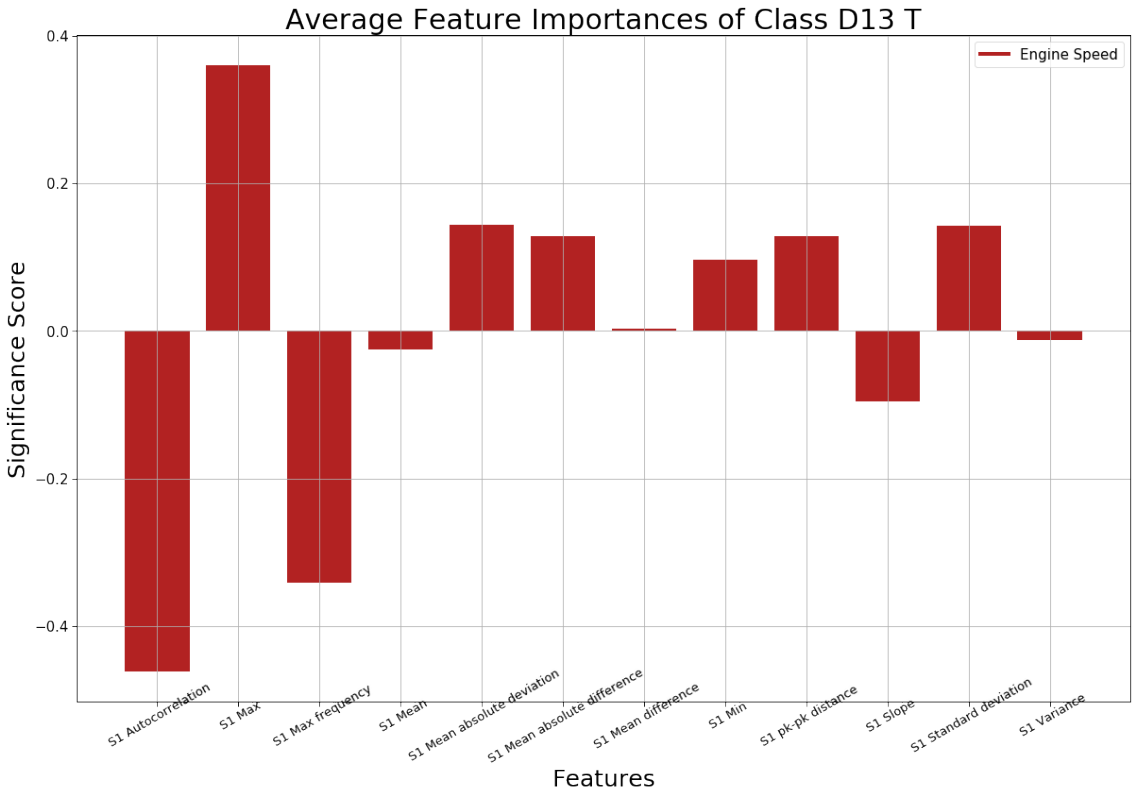


Figure 4.25: Feature importance for class: *D13 T*

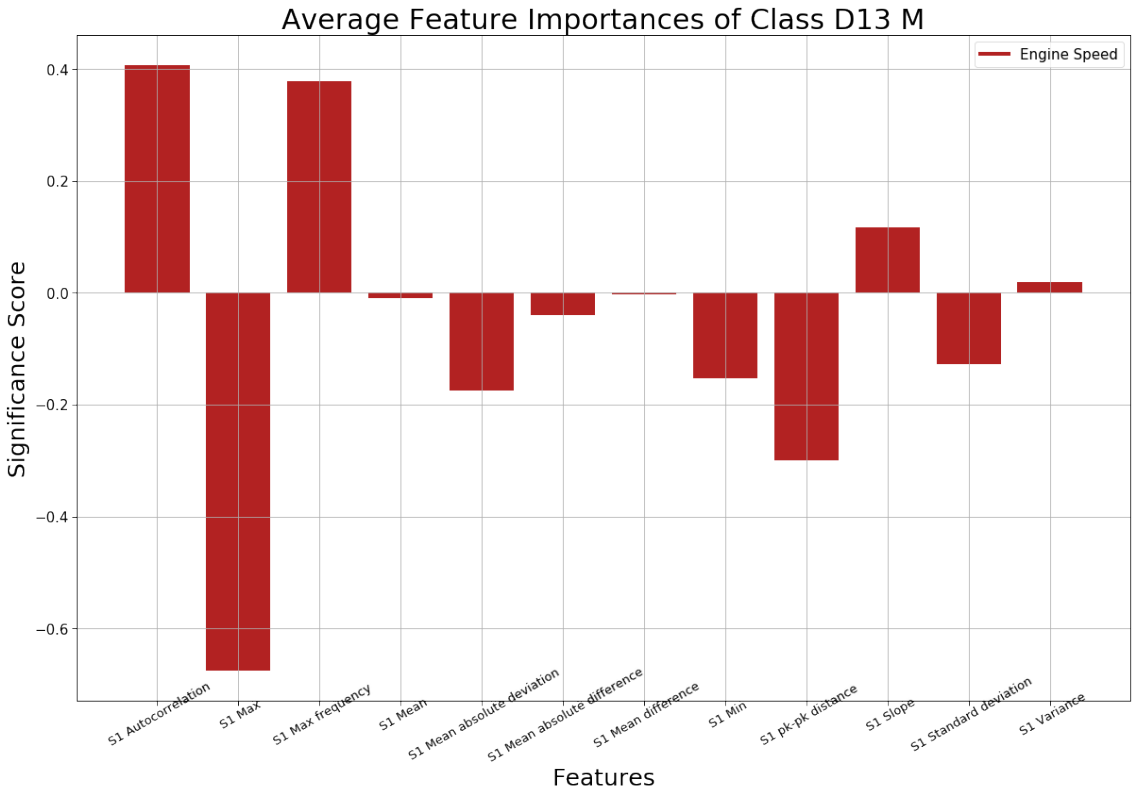


Figure 4.26: Feature importance for class: *D13 M*

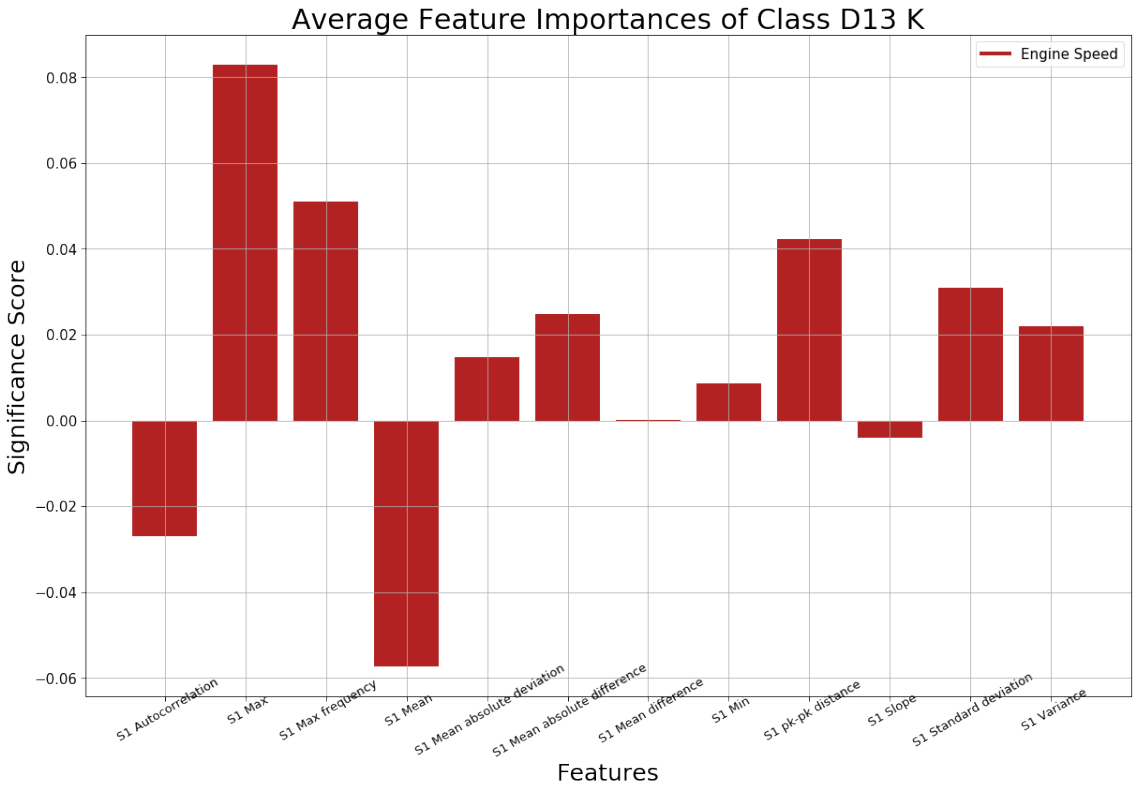
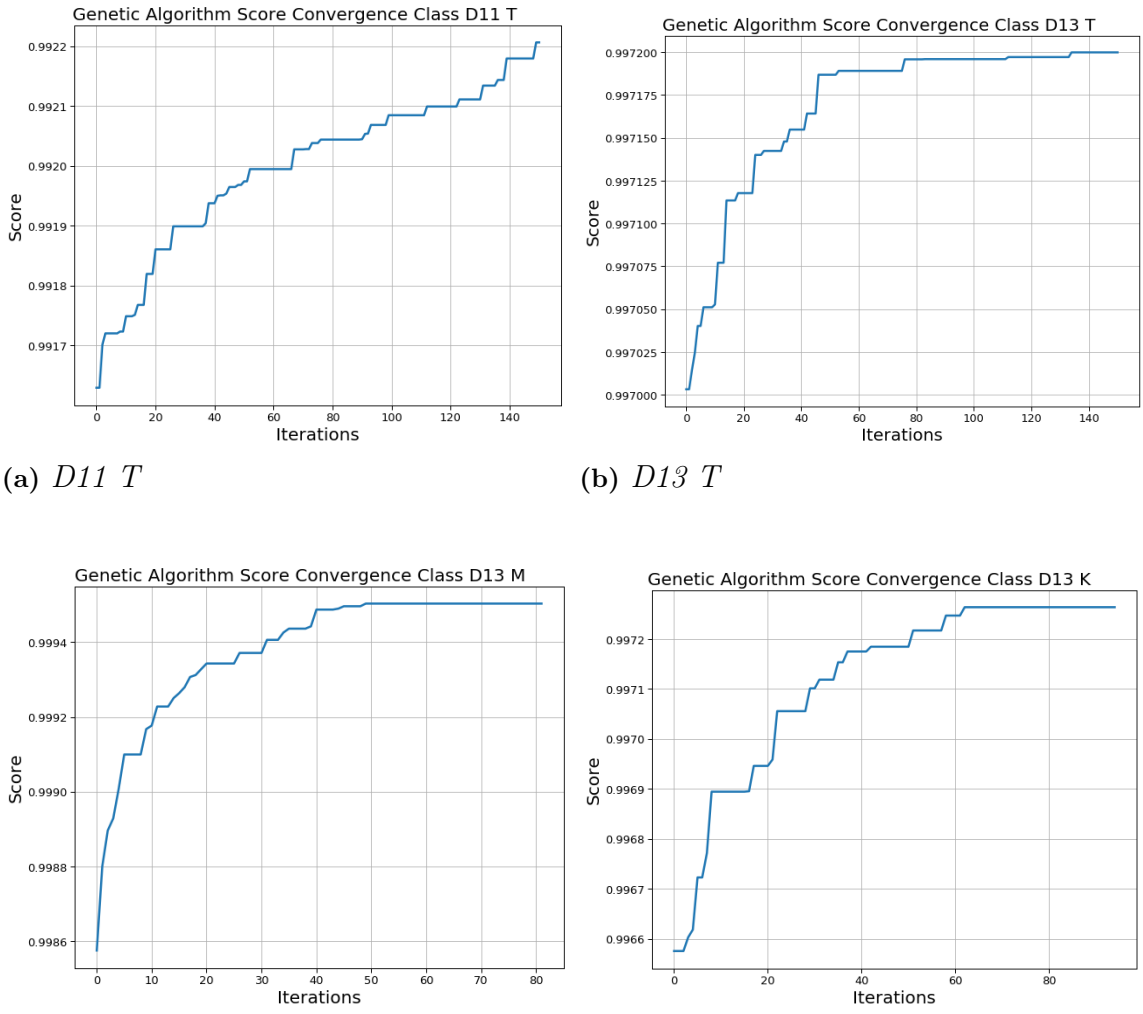


Figure 4.27: Feature importance for class: *D13 K*

4.3.2 Genetic Algorithm Performance and Generation of Engine Drive Cycles

This section presents the result of the generation of the engine speed drive cycle for each class. Figure 4.28 presents the genetic algorithm score convergence for each class. Note that this is the fitness score which is the ANN score output for each class. In all generations, the genetic algorithm manages to converge at around 0.99 or more. Note that a different number of iterations are needed for each class.

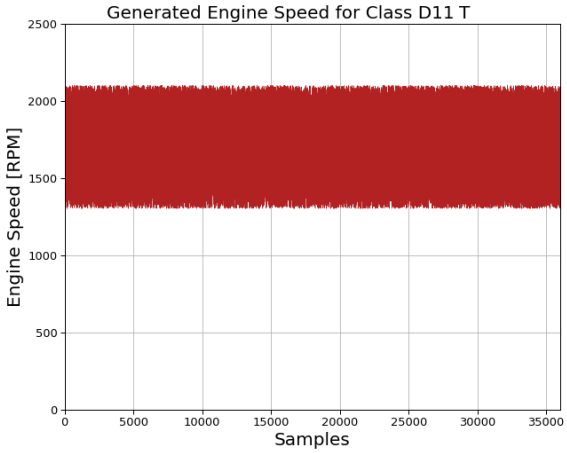


(a) *D11 T* **(b)** *D13 T*

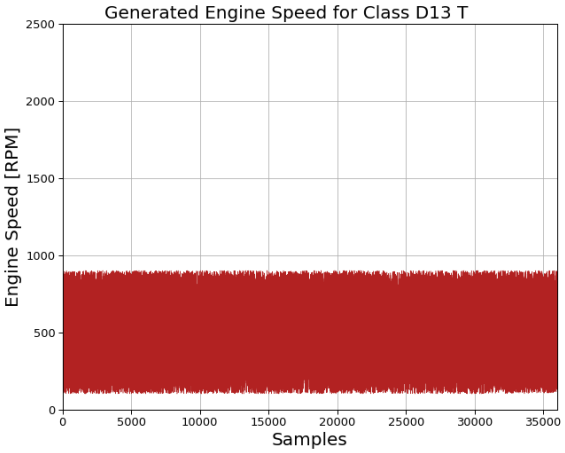
(c) *D13 M* **(d)** *D13 K*

Figure 4.28: Score convergence for the genetic algorithm.

Figure 4.29 presents the final result of the generated engine speed drive cycles. Note that the cycles are generated with main constraint of a minimum and a maximum value of the generated sequence. The features corresponding to the generations are presented in Figure 4.30, for four different classes, where the three most important features are shown in a pair plot. The most important features are selected from the feature importance presented in the Figures 4.24-4.27 with the highest significance score corresponding to each class.

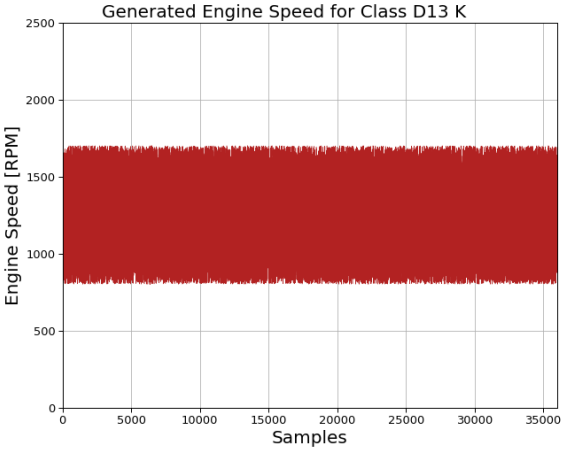


(a) *D11 T*



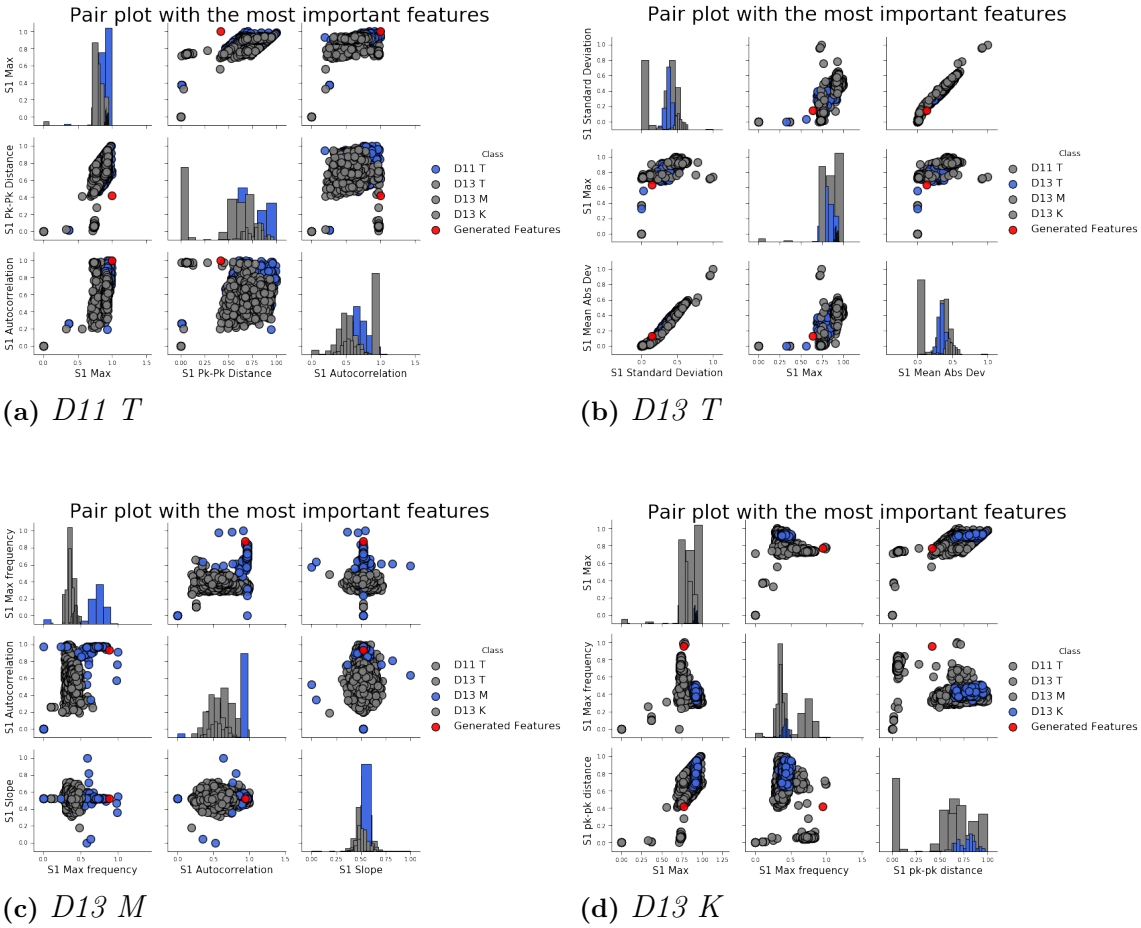
(b) *D13 T*

(c) *D13 M*



(d) *D13 K*

Figure 4.29: Engine speed generation provided by the genetic algorithm.



(a) *D11 T* **(b)** *D13 T*

(c) *D13 M* **(d)** *D13 K*

Figure 4.30: Scatter plot and data distribution

4.3.3 Discussion on Drive Cycle Generation

The dataset presented in Figure 4.21 shows that the data corresponding to each class is not equally distributed among all classes and follows the same discussion made for Figure 4.1, as the methods share the same dataset. The exception is that for this method, only the engine speed features are extracted as the genetic algorithm is bounded to generate only one signal. The consequence of this choice is that the ANN receives fewer features as input, thus making it harder for the ANN to model the mapping. Figure 4.22 proves this statement as the performance of the ANN presents a lower general accuracy convergence to around 0.9, which equals to 90% correct classifications. Figure 4.23 shows the confusion matrix, which indicates a larger misclassification on *D13 T* as the model thinks that the cycle is a *D11 T*. The confusion matrix generally shows, compared to previous ANN evaluations, that the ANN has a harder time to distinguish the classes. This concludes that more signals included in the feature extraction as input to the ANN will increase the general performance of the classifications based on the structure of this ANN. Further study if a more complex ANN structure could model the mapping even better needs to be conducted.

The score convergences in Figure 4.28 presents the genetic algorithm fitness score for each iteration in the algorithm. Note that the fitness score is the score output from the ANN. In Figure 4.28, a high score convergence can be observed, at different points in the iterations. The iterations vary due to the early stopping mechanism that breaks the loop when a higher score has not been found in 50 iterations. This is used as the genetic algorithm is time consuming, and the result of the scores are satisfying enough for evaluation purposes.

The generated engine speed drive cycles are presented in Figure 4.29. What can be observed is a one-hour cycle for each class, evaluated with the fitness score, which has been manually constrained by a minimum and maximum value for the amplitudes for each class. This constraint was necessary since the genetic algorithm did not converge in a reasonable time period, and therefore, an iterative tuning process of finding the minimum and maximum value was conducted. The values that were selected to maximize the fitness score of the algorithm did not reflect the real minimum and maximum of each drive cycle. However, there is an exception for *D13 M* observed in Figure 4.29c compared to the extracted drive cycle in Figure 4.10. The explanation for this is most likely to be a combination of the lower general accuracy of the ANN performance, together with the combination of features presented by the feature importance. The features that most typically represent the characteristics of the engine speed drive cycle is either: not sufficient enough in terms of feature importance of the ANN, or not included in the feature extraction process. This concludes that further studies regarding what characterizes the engine speed drive cycle in terms of feature representations must be conducted. The fitness function, which is the ANN with no constraints, will most likely be able to perform better if constraints are introduced. From a robustness point of view it is interesting to see that an unrealistic time series has the ability to set a relatively high score output from the ANN, which implies that the ANN is not robust enough.

By observing the scatter plot in Figure 4.30, the features tend to be near, or at least in the vicinity of its corresponding cluster. An exception is seen in Figure 4.30d for class *D13 K*, where the generated features do not appear to be close to the corresponding cluster. No conclusions can be drawn from this deviance as it is hard to visually observe the mapping of the ANN.

5
Conclusion

The main goal of the study, stated by the problem formulations in Section 1.3, was to determine how the engine data can be used for classification purposes. As concluded in the discussions in Section 4.1.3 and based on the results in Chapter 4, the structured data acquired by Volvo Penta is shown to be useful when it comes to classification because the results present enough separability between engine drive cycle features.

The experiments confirmed that by using an ANN, it is possible to identify and detect engine drive cycles for different engine sizes and applications. In addition, our study shows that using more signals as input to the network can provide a higher general accuracy of the predictions. The annotated training data defines the mapping of the ANN features to classes and with the correct competence of annotating training data, our study proved that an ANN can successfully be utilized for semantic segmentation of engine states.

Utilizing the ANN as a fitness function for the genetic algorithm has been proved to provide an understanding of the robustness of the mapping behavior for a given ANN. Although, the generation of engine drive cycles approach cannot generate a cycle which fulfills the characteristics of a realistic engine drive cycle. As the ANN is utilized for different methods seen in Sections 4.1-4.3, with small modification on the ANN, it would be able to perform detection, semantic segmentation and generation of engine drive cycles.

The findings will be of interest to Volvo Penta as a proof-of-concept study, which lays the foundation for further research in advanced analytics towards applied machine learning. Although this study focuses on practical implementations of machine learning, the findings may have a bearing on the understanding of the engines and the corresponding application that they are installed in. In this study, four engines were used for classification, and it would be of interest to further study the robustness of the ANN by introducing more engines. Intuitively the performance of the ANN would suffer when introducing more engines since the feature input might be less separable.

6
Future Work

The results from drive cycle detection, Section 4.1, show that the network performs well and extracts the top five engine drive cycles that the network considers to be well represented. However, from analysis and discussion of the results there was a clear flaw where, e.g., *D11 T*'s top picks had several cycles not representing the general case for that application. Future work to achieve more satisfying results would be to clear "bad data" as well as possible. In other words, we should make sure that the data used for training the network is "good data" which can represent the general behavior of an application. Hence, a more comprehensive study of the data and removing unnecessary data points could potentially lead to better and more accurate engine drive cycle detections.

Interesting to see in future work for semantic segmentation would be to increase the number of labeled data, and see if this provides better accuracy. Also as discussed in 4.2.3, with smaller segments where data is labeled, better accuracy on the actual labeling can be achieved. A more complex set of data points with more classes and less intuitive differences in the states is needed for a future study to conclude the robustness of this method. Therefore, a valid study would be to annotate a known application and its states that have complex dynamical behavior. This would be interesting from an analytical point of view as it enables the approach to find behaviors in the time series that are hard to distinguish for a person.

Concluded in the discussions regarding drive cycle generation, Section 4.3.3, a synthetic engine drive cycle is generated with the help of the *genetic algorithm*. However, the result did not match with how Volvo Penta usually perceive these engine drive cycles. Therefore, future work would be to tweak parameters within the genetic algorithm to allow it to restrict what kind of cycles the algorithm can generate. An example of this could be to increase the interval of when the algorithm generates a data point allowing more spacing in between data points and hopefully achieving more realistic cycles. Adding constraints to the fitness function is also a way to constrain the generated cycles to behave more realistically.

The main focus of this thesis has been to see if the structured data collected by Volvo Penta can be used to solve some of their challenges. So far, detection, semantic segmentation and cycle generation have been possible using the data with an ANN. Future work on this project would be to see how the ANN works on a larger scale, since throughout this thesis the amount of used data is just a portion of the data that Volvo Penta has acquired. The question to answer here should be: Is the

ANN good enough to handle big data or does it need re-evaluation to obtain better stability and accuracy?

Further, another interesting area that would provide Volvo Penta with great competitive advantage for their own product line is to be able to perform advanced analytics on their engine data. As an example, predictive maintenance is an important topic when it comes to products, such as engines, which has components that might tear or wear over time. Being able to predict if a potential engine breakdown is imminent can save service hours, but most importantly, it can reduce service costs and provide better reliability for Volvo Penta's products. By using the solution provided in this thesis, and the fact that an ANN can detect an engine drive cycle, it would be interesting to implement predictive maintenance to study these engine cycles. An example is to alert when the measured data is deviating from general operational behavior for a specific application.

References

[1] K. F. Man, K. S. Tang, and S. Kwong. "Genetic algorithms: concepts and applications [in engineering design]". In: *IEEE Transactions on Industrial Electronics* 43.5 (1996), pp. 519–534. DOI: 10.1109/41.538609.

[2] R. Sathya and Annamma Abraham. "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification". In: *International Journal of Advanced Research in Artificial Intelligence* 2 (Feb. 2013). DOI: 10.14569/IJARAI.2013.020206.

[3] Marília Barandas et al. "TSFEL: Time Series Feature Extraction Library". In: *SoftwareX* 11 (2020), p. 100456. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2020.100456. URL: https://www.sciencedirect.com/science/article/pii/S2352711020300017.

[4] Nonso Nnamoko et al. "Evaluation of Filter and Wrapper Methods for Feature Selection in Supervised Machine Learning". In: *The 15th Annual Postgraduate Symposium on the convergence of Telecommunication, Networking and Broadcasting*. June 2014, pp. 63–67.

[5] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG].

[6] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. "Visualizing the Impact of Feature Attribution Baselines". In: *Distill* (2020). https://distill.pub/2020/attribution-baselines. DOI: 10.23915/distill.00022.

[7] Alberto Landi et al. "Artificial Neural Networks for nonlinear regression and classification". In: *2010 10th International Conference on Intelligent Systems Design and Applications*. 2010, pp. 115–120. DOI: 10.1109/ISDA.2010.5687280.

[8] Warren S. McCulloch & Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics*. Bulletin of Mathematical Biophysics 5, 1943, pp. 115–133. DOI: 10.1007/BF02478259.

[9] Howard B. Demuth et al. *Neural Network Design*. 2nd. Stillwater, OK, USA: Martin Hagan, 2014. ISBN: 0971732116.

[10] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG].

[11] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[12] Zhang Jiawei. *Gradient Descent based Optimization Algorithms for Deep Learning Models Training*. Mar. 2019. URL: https://arxiv.org/pdf/1903.03614.pdf.

[13] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[14] Paul Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78 (Nov. 1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[15] scikit-learn. *Metrics and scoring: quantifying the quality of predictions*. 2021. URL: https://scikit-learn.org/stable/modules/model_evaluation.html.

[16] Lawrence Mosley. "A balanced approach to the multi-class imbalance problem". In: *Graduate Theses and Dissertations, IOWA State University*. 2013. DOI: https://doi.org/10.31274/etd-180810-3375.

[17] S. D. Immanuel and U. K. Chakraborty. "Genetic Algorithm: An Approach on Optimization". In: *2019 International Conference on Communication and Electronics Systems (ICCES)*. 2019, pp. 701–708. DOI: 10.1109/ICCES45898.2019.9002372.

[18] Peshawa Muhammad Ali and Rezhna Faraj. "Data Normalization and Standardization: A Technical Report". In: *Machine Learning Technical Reports*. Jan. 2014, pp. 1–6. DOI: 10.13140/RG.2.2.28948.04489.

A

Appendix 1: Validation set

The Figures A.1-A.4 included in this appendix presents the validation set which is mentioned in Section 3.1.1. The figures presents three signals: Engine speed, SCR temperature and Engine torque, along with the sequential ANN score distribution.

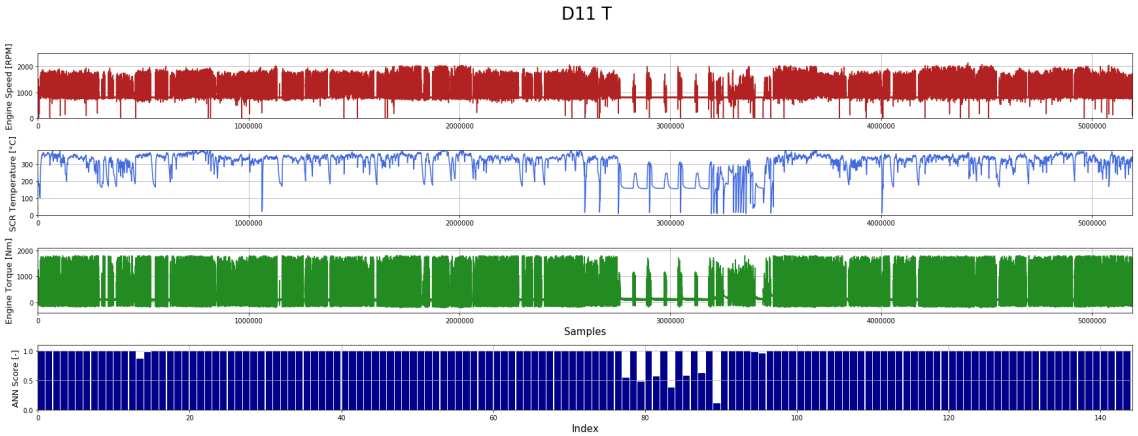


Figure A.1: Overview of the validation set covering a single month, 2020-10, for the class *D11 T*.

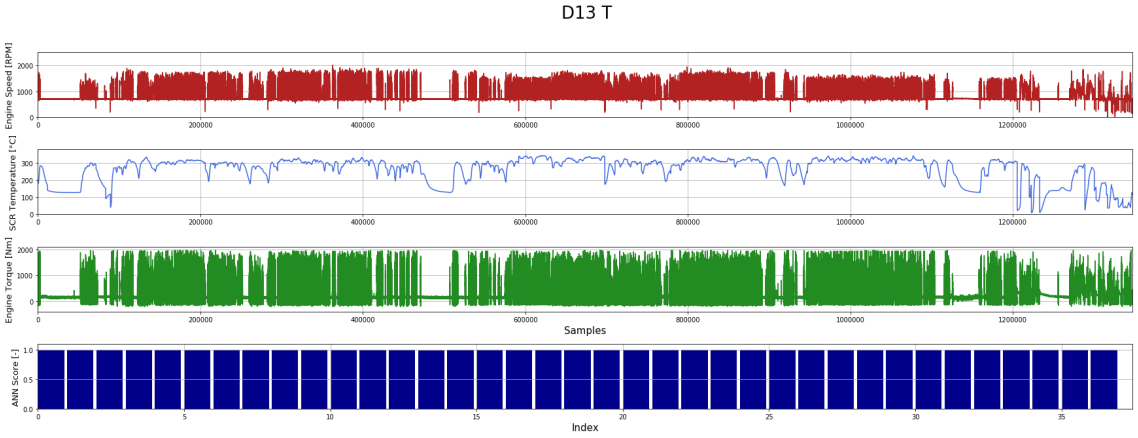


Figure A.2: Overview of the validation set covering a single month, 2020-10, for the class *D13 T*.

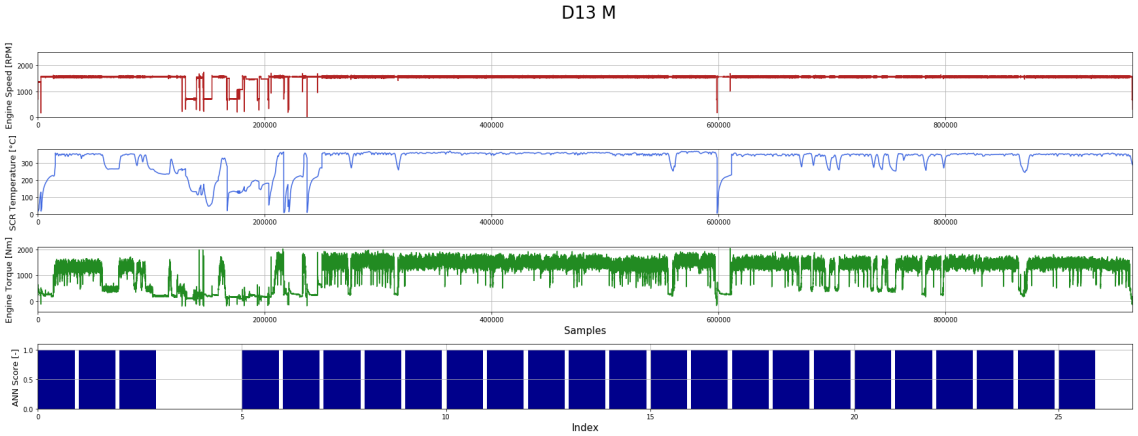


Figure A.3: Overview of the validation set covering a single month, 2020-10, for the class *D13 M*.

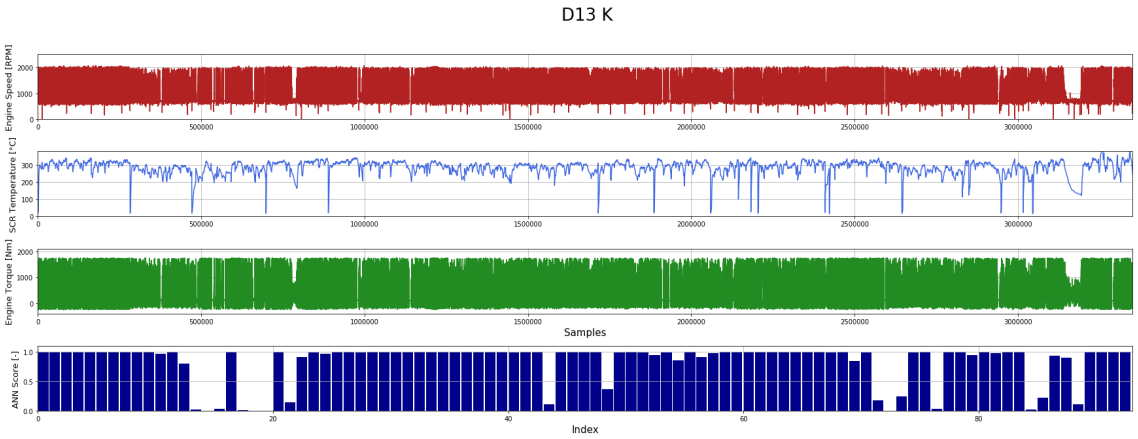


Figure A.4: Overview of the validation set covering a single month, 2020-08, for the class *D13 K*.