



Automated terrain classification using vibrations and a feed-forward neural network

**ITRI 671
HFR Potgieter
29971349
B.Sc. Hons. 2021**

School of Computer Science and Information Systems
Potchefstroom Campus NWU

Supervisor: Prof. JV du Toit

Date of submission: 8 November 2021

ABSTRACT

Automated terrain classification can be described as the ability to identify different terrains based on feedback from the environment on a specific object using a technique such as deep learning. Automated terrain classification using vibration readings has a number of practical applications for society. One of these applications is autonomous self-driving vehicles. With the growing popularity of these vehicles, the ability to classify the terrain the vehicle is currently traversing hold several key benefits for these vehicles. The primary benefit being the fact that these vehicles could traverse various terrains more effectively by adjusting torque settings to accommodate each terrain. Automated terrain classification can be achieved in a number of ways. This study intends to classify terrains using a feed-forward neural network model built in the Keras Application Programming Interface (API) using vibration readings. This study found that a feed-forward neural network can be constructed in the Keras API to accurately classify six different terrains namely bricks, dirt, epoxy, grass, stone and tarmac using vibration readings.

Keywords: Autonomous vehicles, deep learning, feed-forward neural networks, Keras API, terrain classification, vibrations

TABLE OF CONTENTS

ABSTRACT	1
CHAPTER 1 – INTRODUCTION AND PROBLEM STATEMENT	1
1.1 Introduction.....	1
1.2 Problem description and background.....	1
1.3 Aims and objectives.....	2
1.4 Procedures and methods that will be used	3
1.5 Approach to project management and project plan	4
1.5.1 Project plan	4
1.5.2 Project Scope.....	7
1.5.2.1 Requirements	7
1.5.2.2 Constraints	7
1.5.2.3 Risks	8
1.5.3 SWOT analysis	8
1.6 Development platform, resources, and environments that will be used	9
1.6.1 Development platforms	9
1.6.2 Resources	10
1.6.3 Environments	10
1.7 Ethical and legal implications and dealing with these.....	10
1.8 Provisional chapter division	10
1.9 Chapter summary	11
CHAPTER 2 – LITERATURE REVIEW.....	12

2.1	Introduction to review.....	12
2.2	Terrain classification.....	12
2.3	Feed-forward neural networks	14
2.4	Keras API	16
2.5	Uber Ludwig	18
2.6	LEGO Mindstorms robots and Raspberry Pi Sense HAT modules.....	21
2.7	Conclusion to review.....	23
2.8	Chapter summary	24
CHAPTER 3 – DEVELOPMENT OF ARTEFACT		25
3.1	Description of artefact.....	25
3.1.1	The artefact's backend	25
3.1.2	The artefact's front-end.....	27
3.2	Life cycle and different phases	29
3.2.1	Design science research	29
3.2.2	Positivism research paradigm.....	30
3.2.3	The artefact life cycle	30
3.3	Artefact development description.....	33
3.3.1	Create the product backlog.....	33
3.3.2	Planning the sprints	34
3.3.3	Executing the sprints	34
3.3.3.1	Sprint 1	34
3.3.3.2	Sprint 2.....	34
3.3.3.3	Sprint 3.....	35

3.3.4	Demonstrate the artefact	37
3.3.5	Sprint retrospective	37
3.4	Chapter summary	38
CHAPTER 4 – DATA COLLECTION AND PRE-PROCESSING.....		39
4.1	Data collection	39
4.2	Data pre-processing	39
4.2.1	Combining the data sets	39
4.2.2	Pre-processing the combined data set	41
4.2.3	Creating the three smaller data sets.....	43
4.3	Chapter summary	43
CHAPTER 5 – RESULTS		44
5.1	Introduction.....	44
5.2	Comparative study between artefact and Uber Ludwig model	44
5.3	Chapter summary	47
CHAPTER 6 - REFLECTION		48
6.1	Experience gained from the study.....	48
6.2	Evaluation of aims and objectives.....	48
6.3	Management of the study	50
6.4	Concluding remarks about the study and future work	50
6.5	Chapter summary	50
BIBLIOGRAPHY		51
APPENDIXES		54

7	APPENDIX A: RESEARCH ETHICS FORM	54
8	APPENDIX B: RESEARCH PROPOSAL FORM HONOURS PROJECT	58
1.	STUDENT INITIALS, SURNAME AND STUDENT NUMBER	58
2.	DEGREE FOR WHICH STUDENT IS REGISTERED	58
3.	NAME OF SUPERVISOR.....	58
4.	PROPOSED TITLE	58
5.	PROBLEM STATEMENT AND SUBSTANTIATION	59
6.	RESEARCH AIMS AND OBJECTIVES.....	59
7.	BASIC HYPOTHESIS (WHERE APPLICABLE)	60
8.	METHOD OF INVESTIGATION.....	60
8.1	Literature study.....	60
8.2	Methods of investigation	61
9.	PROVISIONAL CHAPTER DIVISION	61
10.	LITERATURE REFERENCES.....	63

LIST OF TABLES

Table 1: SWOT analysis.....	8
Table 2: Summary of the steps of the scrum software development methodology.....	32
Table 3: Product backlog for the artefact	33
Table 4: Evaluation of the aims and objectives of the study.....	49

LIST OF FIGURES

Figure 1: Project plan	6
Figure 2: The different network elements adopted from Fine (2016).....	15
Figure 3: The two different implementations of Keras (Géron, 2019).	17
Figure 4: A graphical representation of the Encoder-Combiner-Decoder architecture (Molino <i>et al.</i> , 2019).	19
Figure 5: An example of a LEGO Mindstorms NXT brick (Valk, 2014).	21
Figure 6: A Raspberry Pi with a Sense HAT module installed (Nusairat, 2020).	23
Figure 7: The resulting model of KerasTuner.....	27
Figure 8: The Graphical User Interface (GUI) of the artefact.....	28
Figure 9: The result of the prediction	28
Figure 10: A graphical representation of the design science research cycles (Hevner & Chatterjee, 2010).	29
Figure 11: An overview of the scrum software development methodology (Rubin, 2012).	31
Figure 12: A summary of the feed-forward neural network built in the Keras API.....	35
Figure 13: The Sequence of steps of the user interface.....	36
Figure 14: Descriptive statistics about the combined data set.....	40
Figure 15: Descriptive statistics about the pre-processed data set.	42
Figure 16: Training and validation accuracy over the epochs for the artefact.....	45
Figure 17: Training and validation accuracy over the epochs for the Uber Ludwig model.	46

CHAPTER 1 – INTRODUCTION AND PROBLEM STATEMENT

In this chapter, the necessary background information for this study will be presented. This includes the problem statement as well as the aims and objectives of the study. The different procedures and methods that will be used will also be presented. This also includes the project plan as well as the resources that will be used during the duration of this study.

1.1 Introduction

Wang (2019) states that self-driving autonomous vehicles have been hailed as a significant achievement to improve transport safety for the masses for the future of traffic systems. These autonomous vehicles must rely on systems that enable them to navigate any given road surface by using powerful computers. Autonomous vehicles use various sensors such as cameras, radars, lidars, the Global Positioning System (GPS), and accelerometers. Information surrounding the road surface these vehicles traverse is essential information these autonomous vehicles must acquire.

Therefore, the terrain classification of autonomous vehicles will play an important role in how effective these vehicles traverse the different terrains they experience. Developing a system that can accurately classify different terrains will significantly benefit these vehicles. This will directly affect the safety of autonomous vehicles and how they traverse their environment to choose the most effective and safest path.

1.2 Problem description and background

Autonomous ground vehicles are employed in many different operational fields like supply and logistics, surveillance, search and rescue missions, and agricultural applications (Weiss *et al.*, 2006). These operations may be necessary to traverse some indoor or off-road terrain, affecting the vehicle's performance. The efficiency of these vehicles can be improved by detecting their environment. In any outdoor setting, the surface is inherently diverse, some territories may be flat and smooth, and some may be rugged and bumpy. Thus, the surfaces outdoor can be described as hazardous environments and therefore, any autonomous vehicle must know what kind of surface it is currently traversing. Autonomous vehicles will play an essential role in the transport industry in the future. Therefore, developing a system that can classify terrains autonomously will be very beneficial for these autonomous vehicles.

This act of identifying the type of terrain being crossed is called terrain classification. There are different methods to collect the necessary data to determine the current terrain the robot is

traversing. Vision-based terrain classification uses colour or texture information, whilst vibration-based methods measure the vibration induced in the autonomous vehicle. At the same time, for the latter approach, the car traverses the terrain it is currently on. Vibration-based classification has the advantage over vision-based classification in that the former is independent of lighting conditions. Therefore, the necessary data to train the deep learning algorithm were obtained by traversing different terrains in this project. According to Brooks and Lagnemma (2005), terrain classification would allow an autonomous vehicle to adapt its planning and motion control strategies to cross various types of terrain safely.

A feed-forward neural network will be trained to classify the various terrains that will be traversed. This type of neural network can accurately estimate almost any function of interest (Bebis & Georgiopoulos, 1994). The Keras application programming interface (API) will be used to build and train the feed-forward neural network. Keras is implemented in the programming language Python and is a highly modular neural network library (Jiang & Shen, 2019). In the remainder of this report, it will be referred to as the Keras API.

To determine the effectiveness of the neural network, it must be compared to an existing framework. Therefore, the Uber Ludwig system will be used (Hutson, 2019). It is open-source software that can train itself when given two files: training data in a spreadsheet file and a configuration file indicating which columns are samples and labels. Once the Uber Ludwig system starts to recognise associations, the software can answer questions or make numerical estimates.

A LEGO Mindstorms robot combined with a Raspberry Pi Sense HAT has been used to collect the data to train the feed-forward neural network in a previous study. The supervisor of this study has supplied this data, and the participants of the previous research have granted the necessary permission. The Raspberry Pi Sense HAT module was used to take multiple measurements whilst the robot traversed different terrains. This study aims to create and train a feed-forward neural network to classify different terrains based on vibration readings.

1.3 Aims and objectives

The primary aim of this study is to construct a feed-forward neural network that can accurately classify terrains based on the input data provided by the supervisor. To achieve the previously mentioned aim, the following objectives and sub-objectives must be satisfied:

1. A thorough literature study must be conducted on the primary concepts in regards to the following:

- a. A terrain classification system will be defined, and its purpose and implementation will be identified and discussed. Literature about neural networks and terrain classification will be studied in general.
 - b. The overview of feed-forward deep learning frameworks will be studied as well as no-code deep learning frameworks.
 - c. The Keras API will be studied in-depth as it is the primary objective of the study is to implement a feed-forward neural network.
 - d. The Uber Ludwig deep learning framework will be studied to build a model to compare the previous framework's implementation effectiveness.
 - e. Previous attempts of building terrain classification systems using feed-forward neural networks will be studied to identify their findings and potential shortcomings.
2. Obtain the dataset provided by the supervisor that will be used for this study:
 - a. Modify the dataset to meet the requirements of the supervisor.
 - b. Split the dataset into a training set, a validation set, and a testing set.
3. Use these data sets to construct the model definition for Uber Ludwig.
4. Train the Uber Ludwig deep learning framework.
5. Use these data sets to construct a feed-forward neural network using the Keras API.
6. Train the feed-forward neural network.
7. Compare the accuracy of the Keras API model with the Uber Ludwig framework model.
8. Conclude on the artefact's accuracy and reflect on the insights obtained and any potential shortcomings identified.

1.4 Procedures and methods that will be used

Design science is the study and design of artefacts built to interact with a problem to improve something in the problem's context (Wieringa, 2014). According to Hevner and Chatterjee (2010), design science is a research paradigm wherein answers to questions are provided utilizing an artefact in context to human problems. Therefore, this study will use the design science research methodology as the primary goal of this study is to build an artefact that provides a solution to a problem in a context. The design science research methodology has six steps: problem identification and motivation, identifying the objectives for the solution, the design and development of the artefact, and demonstration, evaluation, and communication. Therefore, these steps will ensure that a satisfactory artefact is designed and developed.

Since the accuracy and effectiveness of the artefact can be measured, an experimental research method will be used. Therefore, the positivism research paradigm will be used. The core function

of the positivism research paradigm is to be able to make a prediction based on the phenomena in question (Park *et al.*, 2020).

The supervisor provides the dataset that will be used for this study. This dataset will be divided into three smaller datasets: *the training set*, *the validation set* and *the test set*. These three datasets will be used to train and test the feed-forward neural network model.

To build the artefact, the Agile software development life cycle will be utilised. According to Ahmed *et al.* (2010), agile means active, swift, and responsive, and this is the core definition of this methodology. This software development life cycle has several incremental and iterative phases (Bhalerao *et al.*, 2009). Therefore, if changes are made during the artefact's development, they can be made swiftly. This approach also allows stakeholders to react quickly to changes that might be required and provide the necessary flexibility to accommodate quick changes (Thesing *et al.*, 2021). Thus, the supervisor will always be able to provide regular feedback on the artefact's development progress.

1.5 Approach to project management and project plan

1.5.1 Project plan

- (i) Planning and research proposal phase – in this section, the project proposal and project plan will be written and brought forth. In this part, the problem will be identified, and the background on that problem will be discussed. A literature study on essential concepts will be undertaken. Furthermore, potential problems that may arise during the study will be identified. The aim and objectives of the study will be stipulated to ensure that there is a structure that is to be followed. Any resources that the product requires will also be identified and taken into account. This document will be sent to the supervisor of this study for appropriate feedback.
- (ii) Conduct literature study – A thorough investigation into important concepts tied to this study will be conducted. There are several necessary research fields such as Lego Mindstorm robots, Raspberry PI Sense HATs, terrain classifications by using vibrations, feed-forward neural networks, the Keras library, and the Uber Ludwig no-code deep learning framework. These previously mentioned topics will be discussed and sent to the supervisor of this study for appropriate feedback.
- (iii) Build artefact and design poster – In this section, the artefact will be built, and the poster for this study will be designed. This section is characterised by developing the feed-forward

neural network in the Keras library and the evaluation thereof by using the Uber Ludwig deep learning framework. Any findings will be put on a poster summarising the entire study. After that, the researcher will prepare for the physical demonstration of this study.

- (iv) Demonstrate artefact – A video demonstration of the study's findings will be demonstrated to the interested parties.
- (v) Artefact documentation – In this section, all the documentation of the study will be combined. This document will contain all the necessary information and previous documentation such as the study proposal and research proposal, the literature study, the technical detail, and the study results. The researcher will send this document to the supervisor for final feedback.

A project plan has been drawn up as indicated in Figure 1. In this project plan, the different phases, as well as milestones, have been identified in order to successfully complete this study. The dates, phases and millstones identified in the project plan are subject to change.

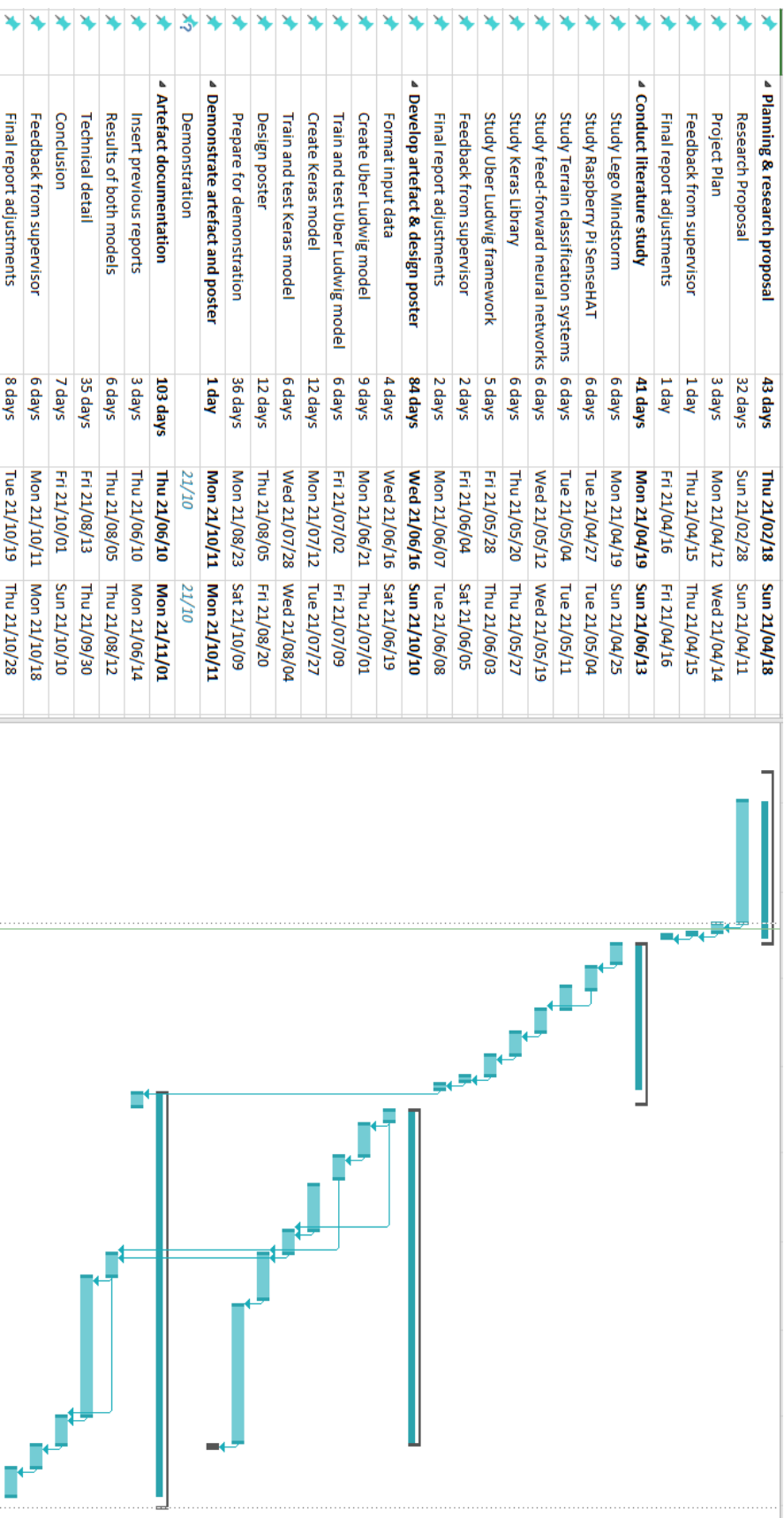


Figure 1: Project plan

1.5.2 Project Scope

The scope of the study involves the documentation and determination of specific requirements, constraints, risks, and analysis of the study. The project has only two parties involved, namely, the researcher and the supervisor of this study. In this section, the requirements, constraints, and risks of the project will be discussed.

1.5.2.1 Requirements

The following criteria determine the successful completion of the study:

- The artefact should be able to accurately classify different terrains based on the dataset provided by the supervisor. In other words, based on vibration.
- The different deliverables should be completed on time.
- The Keras library feed-forward neural network and the Uber Ludwig model must be trained by the dataset provided by the supervisor.

1.5.2.2 Constraints

There are a few constraints that have been placed on the study by the supervisor.

Time constraints:

- The research proposal and the project plan must be handed in by 18 April 2021.
- The literature study must be handed in by 13 June 2021.
- A video demonstration must be submitted on the 1st of November 2021.
- The complete documentation must be handed in on 8 November 2021.

Note: these dates are subject to change.

Monetary constraints:

There is no monetary constraint for the successful completion of this study. All of the required resources are open source at no additional cost.

Resource constraints:

The terrain-classifier that will be built will only determine the terrain that is being traversed with the specific format of the dataset in mind. The artefact must be able to train itself without any human intervention other than changing the feed-forward neural network structure.

1.5.2.3 Risks

Essential risks that may influence the successful completion of the study will now be discussed.

Added workload – New features or a new direction of the study may influence the initial objective of the study, thereby damaging the quality of the final artefact. Therefore, an in-depth discussion on the necessity of the added features or change in direction will be needed to ensure that the study is completed successfully.

Unplanned work that must be accommodated – during the development of the study artefact, there might be unforeseen circumstances that may lead to more work than initially intended. Therefore, time must be set aside to quickly and effectively deal with issues such as this.

Scope creep – additional requirements or features that were not included in the study scope may arise as the study progresses. This could have significant implications for the successful completion of the study. Therefore, the researcher will have to evaluate whether any new additions to the study's scope is necessary and what value these additions will bring to the study.

Schedule risks – The study may not be completed in the time frame specified. This, however, is not a concern as the tasks have been scheduled with a generous amount of lead time between each task.

1.5.3 SWOT analysis

Table 1: SWOT analysis

Strengths <ul style="list-style-type: none">• The feed-forward neural network may be able to classify terrains based on vibrations accurately.• This artefact could be included in further research by other researchers suiting their needs.	Weaknesses <ul style="list-style-type: none">• The feed-forward neural network's classification of terrains will be only as good as the model that the researcher implements.• The data in the dataset is in a particular format; therefore, if there are any changes to the dataset structure, the entire model will need to be changed to accommodate this change.
Opportunities	Threats

<ul style="list-style-type: none"> • This artefact could be implemented in autonomous vehicles to provide additional information to the vehicle's systems, therefore helping the autonomous vehicle be more effective on any terrain. • This artefact could be a building block for many other implementations of terrain classification using different types of neural networks. 	<ul style="list-style-type: none"> • Researchers and other interested parties may find other artefacts that are more flexible or suit their specific needs.
--	--

1.6 Development platform, resources, and environments that will be used

In this section, the different platforms that will be used will be outlined. The various resources that will be used to complete this study will be identified. The environments that will be used to develop the artefact will be determined.

1.6.1 Development platforms

Keras is built on top of TensorFlow written in the programming language Python. This library will be used to create the feed-forward neural network for this study. It is a highly productive interface for the solving of machine learning problems. Keras provides the necessary building blocks and abstractions for building machine learning systems. Keras will be used to create a model for the feed-forward neural network of this study.

Uber Ludwig is a toolbox that enables the training and testing of deep learning models without writing a single line of code. It is also built on top of TensorFlow and can be used in the Python programming language. To train a model with a dataset, a list of columns for input and a list of columns for output, and Uber Ludwig will perform the rest of the functions. It is used to be able to train and test deep learning models relatively quickly. Therefore, Uber Ludwig will be used as a baseline to determine the accuracy and effectiveness of the model that will be built in Keras.

Pandas is flexible, powerful, and a fast data manipulation and analysis tool that is also open source. This tool is also built on top of the Python programming language. The DataFrame object in Pandas will be used to manipulate the dataset provided by the supervisor. The reason is to remove the two columns that would not be needed and split the dataset into a training set, validation set and a testing set for the two previously mentioned deep learning models. The data will also be processed before it will be used in the two deep learning models.

1.6.2 Resources

This study will only require one resource, and the supervisor provides it. It is the dataset that will be needed to train the two deep learning models. The dataset is a comma-separated value (CSV) file already in the correct format to train the models.

1.6.3 Environments

Visual Studio Code is an open-source code editor made by Microsoft. It supports several functions such as support for debugging, snippets and syntax highlighting. This environment will be used to manipulate the dataset provided by the supervisor for the two models.

Google Colab is a web-based development environment that allows any individual to write and execute any form of code written in the Python programming language. This environment is well-suited for machine learning and will be used to develop the feed-forward neural network.

1.7 Ethical and legal implications and dealing with these

This study did not gather any information or data from any individuals, therefore there are no participants in this study. Thus, there are no legal or ethical implications to this study and this has been indicated in the completed form in Appendix A.

1.8 Provisional chapter division

The provisional chapter division is as follows:

Chapter 1. Introduction

A brief description of the problem being researched, building an automated terrain classification neural network using vibration readings, is discussed in this chapter. The necessary information regarding the background of the study is highlighted, the aims of the study are brought forth, the procedures, as well as the methods that will be utilised, is discussed, the project plan is presented, and the ethical and legal implications are explained.

Chapter 2. Literature review

This chapter is a discussion on related literature to the study. These include key terms such as terrain classification, feed-forward neural networks and any other relevant technology that will be used to build the artefact.

Chapter 3. Development of artefact

Chapter 3 focuses on the creation of the artefact. This discussion includes the development process of the artefact as well.

Chapter 4. Data collection and pre-processing

The various methods utilised to gather, process and analyse the data that will be used to train the feed-forward neural network is discussed in this chapter.

Chapter 5. Results

The results obtained from the study are discussed and reflection on the results will also be discussed.

Chapter 6. Reflection

A brief reflection on what has been learned in this study will be brought forth.

1.9 Chapter summary

In this chapter, a number of key topics were identified that is crucial to this study. Introductory literature concerning the study was introduced such as terrain classification using vibration readings and feed-forward neural networks amongst others. Alongside this, the aims and objectives of this study are central to the success of the study as well as the project plan that will guide the study to a successful conclusion.

A brief description of the research topic was provided as well as the motivation behind the study. The various different platforms and technologies that will be used during this study were also introduced and discussed.

CHAPTER 2 – LITERATURE REVIEW

In this chapter, relevant sources regarding the topics applicable to this study will be discussed. Their findings and results, as well as shortcomings, will also be identified.

2.1 Introduction to review

This literature study aims to shed light on several topics connected to terrain classification and autonomous vehicles. The following topics have been identified: Terrain classification and systems that have implemented it, an overview of feed-forward neural networks, research on the Keras API that will be used to build a feed-forward neural network, the no-code Uber Ludwig framework, and LEGO Mindstorms robots as well as Raspberry Pi Sense HATs modules.

This study will consist of seven distinct sections. In Section 2.2, a discussion on terrain classification will be provided, and different research findings will be discussed. Section 2.3 provides an overview of feed-forward neural networks to understand how these neural networks are built and how they work. In Section 2.4, a discussion on the Keras API will follow as this API will be used to create the previously mentioned feed-forward neural network. Section 2.5 consists of a discussion on the no-code deep learning framework, Uber Ludwig, to understand how this framework is implemented. In Section 2.6, a discussion involving LEGO Mindstorms robots and the Raspberry Pi Sense HAT modules will be given. This was the method used to collect the data in a previous study. Finally, in Section 2.7, a conclusion will be provided in which the main agreements and disagreements in the literature will be identified. After that, any shortcomings or areas for further research from the sources provided in this study will be identified, and an overall perspective of the topic will be brought forth.

2.2 Terrain classification

In this section, research regarding terrain classification and systems that implement them will be discussed. Different approaches to terrain classification will be discussed, and their potential shortcomings will be identified if any.

According to Dupont *et al.* (2008), several economic sectors are expected to be impacted by autonomous vehicles. These autonomous vehicles will be expected to traverse various terrains, which could affect the performance of each vehicle. A terrain classification system aims to identify the terrain before travelling to avoid hazardous objects and landscapes. Recent terrain classification algorithms were based on vision; however, these methods can be unreliable as the

vision of the sensor can be hypothetically obstructed. Vision-based algorithms can also have difficulty distinguishing between terrains that look very similar. Their study attempts to categorize terrains by measuring the frequency response of the vibrations that the vehicle experiences. Hence, this research complements the approach used to gather the data for the terrain classification system this study aims to implement and build. To classify the different terrains, the study made use of a multi-layered feed-forward neural network classification system. The researchers made use of a probabilistic neural network classification scheme. This type of neural network will be discussed further in the following section. Their algorithm was able to classify terrains 85 per cent of the time accurately. Therefore, this is a good starting point to compare the algorithm built for this study. The reason being that the algorithm presented does not produce false positives when classifying terrains. Therefore, the algorithm for this study will be built to handle these situations.

In a study by Weiss *et al.* (2006), different data types could be potentially used for terrain classification. Vision or lidar sensors are used in most approaches. Methods that use lidar aim to fragment the ground surface from vegetation and other obstacles such as rocks. Techniques that use vision utilises visual features to classify terrains based on colour or texture. Vibration-based terrain classification focus on the soil itself as a hazard. The primary aim of this type of terrain classification is to observe the different vibration signals that are induced in the body or wheels of the vehicle. This means that each signal that a terrain type generates can be used for classifying that specific terrain. The research task was to estimate the kind of terrain traversed by a vehicle based on the vibrations induced by the landscape. The vehicle used in the experiment utilized an accelerometer to measure the vibrations the vehicle was experiencing. This is very similar to the method used to gather the data for this study. Their approach used Support Vector Machines (SVMs) to train the model to classify the different terrains accurately. This method produced an accuracy of 95 per cent.

In a study by Bai *et al.* (2019), they propose an improved terrain classification method that utilizes three-dimensional vibration information that is captured by the wheel of the rover making contact with the terrain. This method complements the way the data has been gathered for this study. The data collected also measured the vibrations the vehicle experienced in three dimensions. This information was used to create a neural network model trained using backpropagation to classify the different terrains. Their classification process transformed the initial raw vibration data to form a vector with a duration of one second. After that, the Fast Fourier Transformation (FFT) transforms the frequency domain for training the algorithm. Then a backpropagation neural network was developed to train the model. Their findings suggest that some terrains have higher recognition rates than others. However, their algorithm maintained an accuracy of 89 per cent. To

classify the terrains even more accurately, a suggestion has been made to use a combination of sensors such as vision or lidar.

Komma *et al.* (2009) suggest two different approaches in which ground surface characteristics can be inferred. The first approach estimates terrain parameters directly, such as cohesion, with no prior knowledge of the terrain that is currently being traversed. The second approach groups various terrains into classes representing each ground surface to a certain degree of hazardousness. Then a model is generated that can predict the current class of terrain the vehicle is traversing from a set of available classes using sensor measurements. The approach used is based on the interaction between the different types of terrain and an automated vehicle. Thus, their approach also relied on vibrations to distinguish different terrains by using accelerometers placed at the wheels of a vehicle. The approach in this study was to classify different terrains by using Support Vector Machines in combination with a Bayes filter to combine several successive observations into one final prediction. Their use of a Bayes filter increased the performance of their classifying algorithm by about four per cent.

Interestingly, the different sources used similar methods to gather the data to train the models to classify different terrains. However, each source used other models to train their algorithms, and each of these approaches has generated different results in terms of accuracy.

2.3 Feed-forward neural networks

In this section, neural networks will be discussed. The discussion will include the overall structure of these networks and the different types of neural networks and their applications. The focus will be placed on feed-forward neural networks as this type of neural network will be used to implement a terrain classifier, as discussed in the previous section.

Schmidhuber (2015) asserts that a neural network comprises many interconnected neurons, also called processors, each delivering a series of decimal activations. Sensors that perceive the environment activate the input neurons, and these input neurons activate other neurons through weighted connections. How these networks learn is built on finding weights that ensure that the neural network demonstrates desirable behaviour. Obtaining desirable behaviour may require long chains of computational stages, wherein each of the stages transforms the aggregate activation of the neural network. The idea behind deep learning is to assign credits across a variety of such stages. Schmidhuber also mentions several neural networks that have been created. Models that have few stages have been around for decades and are called shallow

neural networks. Backpropagation neural networks are discrete, distinguishable networks with varying depth and have been praised for implementing supervised learning using the gradient descent method. However, this type of neural network, with many layers, have been found as challenging to implement. Feed-forward neural networks are acyclic, meaning the connections between the nodes do not form a cycle. Whilst recurrent neural networks, such as the name suggests forms cycles between the connections of the different nodes in the model.

Fine (2006) explains the overall architecture of neural networks by making use of mathematics. An architecture is a group of related networks with the same node functions and directed graphs but could have potentially different weights on the various links. A neural network is represented by making use of a directed, weighted graph. In Figure 2, the different elements that make up a neural network are shown.

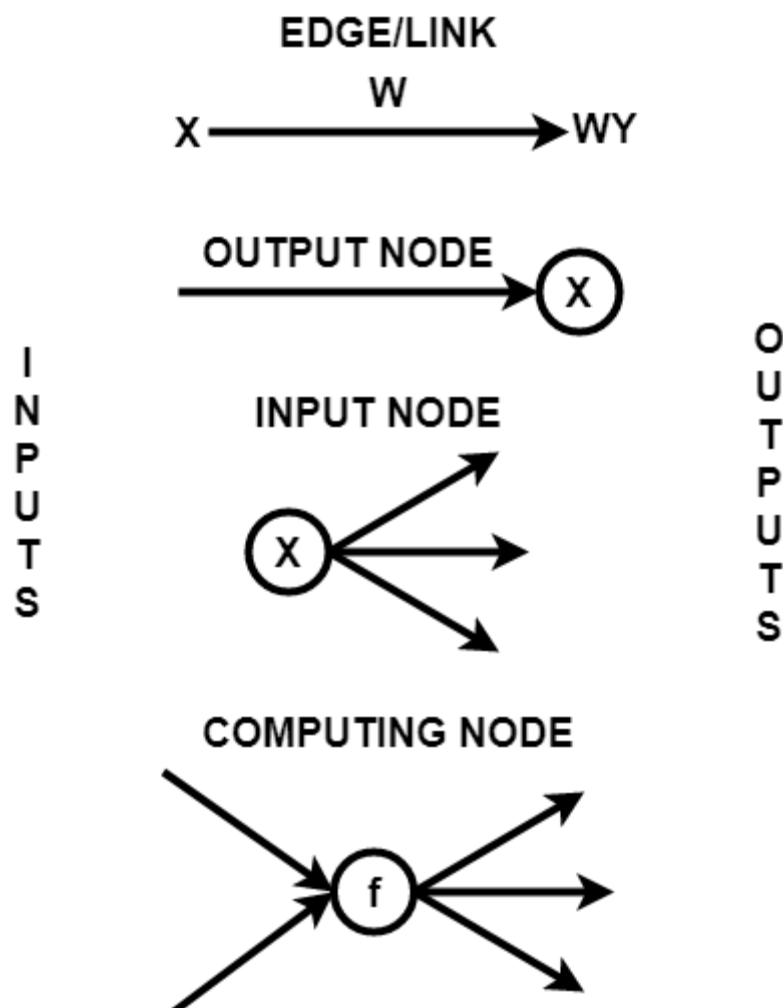


Figure 2: The different network elements adopted from Fine (2016).

As depicted in Figure 2, a graph is a collection of vertices connected by edges that carry a weight, denoted by W , associated with each edge. These two components are used to establish a

network's topology. The information flows from input to output unidirectionally, and the arrows indicate the way the information flows. A network's input variable is determined independently from the network. However, an output node is connected to a variable of interest designated as a single incoming link. Output variables are usually inputted as distinguished input variables or the responses of certain computational elements. The function of the variables that are connected to an incoming link is called a computational element. Feed-forward neural networks are based on the idea of acyclic graphs, meaning there are no closed cycles as previously stated. A directed, acyclic graph with a choice of node functions defines a feed-forward neural network architecture. These types of graphs can be divided into multiple layers. The input nodes are contained in the initial layer, and the final layer consists of the output layers.

Russell and Norvig (2020) give a straightforward explanation of what a feed-forward neural network is and how it functions. A feed-forward neural network has chosen input and output nodes and has connections only in one direction. Each node in the network calculates a function of the inputs it receives and sends the calculation result to its successors in the network. Another term for a node in a network is a unit. A unit is responsible for two operations: compute the weighted sum of the inputs acquired by its predecessor nodes and apply an activation function to create its output. Nodes are responsible for creating continuous outputs by receiving continuous inputs. These inputs can also be parameters of the network, and the network can learn by adjusting these parameters to make the whole network fit the training data.

Feed-forward neural networks are acyclic, meaning that this type of neural network only has connections between the nodes in one direction, and they do not form a cycle. These networks are usually built in layers where the input nodes are contained in the initial layer, and the output nodes are included in the final layer. They are relatively simple to implement and is the chosen network model for this study to build a terrain classifier in the Keras deep learning API.

2.4 Keras API

In this section, the Keras API, which will be used to build the feed-forward neural network to train the terrain classifier, will be discussed. This section will provide an overview of the Keras API and its essential functions.

According to Géron (2019), Keras is an open-source deep-learning application programming interface used to build, train, evaluate different types of neural networks. Keras is relatively easy to use, flexible and was designed by François Chollet as part of a research project and was

inspired by Scikit-Learn and Chainer. Keras relies on a computational backend when heavy computations need to be performed by specific neural networks. Keras can be implemented with three open-source deep learning libraries: *TensorFlow*, *Theano*, and *the Microsoft Cognitive Toolkit (CNTK)*. Code written in Keras can also be implemented in a web browser, therefore Keras code can also be implemented in popular scripting languages such as JavaScript and TypeScript. Keras is also integrated with its TensorFlow implementation called *tf.Keras*. Keras supports TensorFlow as its backend. This integration offers several advantages, such as shown in Figure 3. This figure shows two diagrams for the different implementations of Keras. The left-hand side depicts the implementation of Keras with the three different backends, whereas the right-hand side depicts the integration of Keras with its TensorFlow implementation.

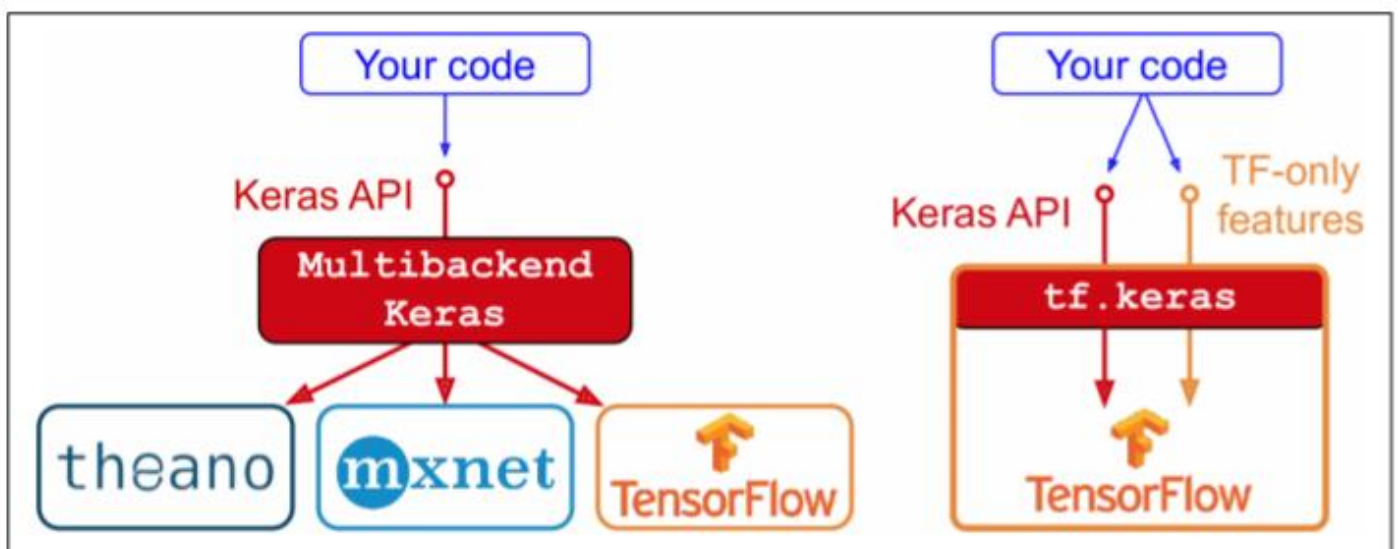


Figure 3: The two different implementations of Keras (Géron, 2019).

The left-hand side has the advantage of supporting TensorFlow's Data application programming interface that makes it possible to load and preprocess data efficiently. TensorFlow 2 has been adopted by Keras as its official application programming backend, and the documentation of this package has been rewritten, making it much easier to use TensorFlow with Keras. The study will use the latter implementation, meaning that a feed-forward neural network will be implemented using Keras with its TensorFlow implementation. The reason being for the advantage mentioned earlier as well as for simplicity.

Brownlee (2016), explains that Keras is a tightly packaged deep learning library that is built for the Python programming language. This library runs on top of Theano or TensorFlow and was developed to build deep learning models relatively quickly and with ease for research and development. Keras runs on top of Python and can execute both on central processing units (CPUs) as well as Graphical Processing Units (GPUs). Keras was developed with four guiding

principles: modularity, minimalism, extensibility, and natively Python. The concept behind modularity is that a model should be understood as a graph alone. Minimalism implies that the library should only provide enough to achieve a certain outcome and no more. Extensibility ensures that new components are easy to include in the underlying architecture of Keras. Every component of the Keras library should be able to run within Python natively with no new file formats or custom model files.

Keras is built around the idea of a model and the primary type of model is called a Sequential model that is a linear stack of layers. Once the model is created, it is compiled by the underlying framework that aims to optimise all the computations to be done by the model. This allows an individual to select the loss function and the type of optimiser to be used in this library. Once the model has been compiled, it must be able to fit the data. This can be achieved by either using the entire dataset or using one batch of data at a time. Once the model has been trained it can be used to make predictions on a new set of data. The construction of deep learning models in Keras can be summarised into four steps. The first step is to define the model. In this step, a sequential model is created, and layers are added to it. Thereafter, the model is compiled by selecting the loss function and the optimiser for the model. The next step, called fit the model, wherein the model is trained by a sample of the input dataset. Then, the model can be used to make predictions on new data that the model has not seen yet.

The Keras API is a highly modular and flexible interface that can be used to build several different types of neural networks. This flexibility of the Keras API, such as being able to select the backend, makes this API very easy to adapt to the specific needs of a user. The fact that the Keras API is native in Python makes this API very easy to set up on any computer with any operating system. This study will make use of the Keras API with TensorFlow as its backend as this allows for the efficient pre-processing of data that will only help to ensure that the model is trained as effective as possible.

2.5 Uber Ludwig

This section will focus on the Uber Ludwig no-code deep learning framework. The underpinnings of how this no-code framework is implemented and how this framework functions will be discussed.

According to Molino *et al.* (2019), Uber Ludwig is a simple, flexible framework that allows individuals to train deep learning models and use these models to make predictions without writing

a single line of code. This framework's primary reason was to enable any individual to construct deep learning models easily. This is achieved by allowing individuals to build a deep learning model by declaring the data and the task of the model. Data types and declarative configuration files are the two primary abstractions upon which Uber Ludwig models are built. The data type abstractions provide the benefit of sub-model reuse and encapsulation due to the standardized interfaces introduced. Declarative configuration files allow novice users to create effective deep learning models whilst simultaneously increasing an expert user's productivity.

In addition to the previous two abstractions, Uber Ludwig is based on a modular deep learning architecture known as Encoder-Combiner-Decoder and can be used for various machine learning applications. Figure 4 provides a graphical representation of this Encoder-Combiner-Decoder architecture.

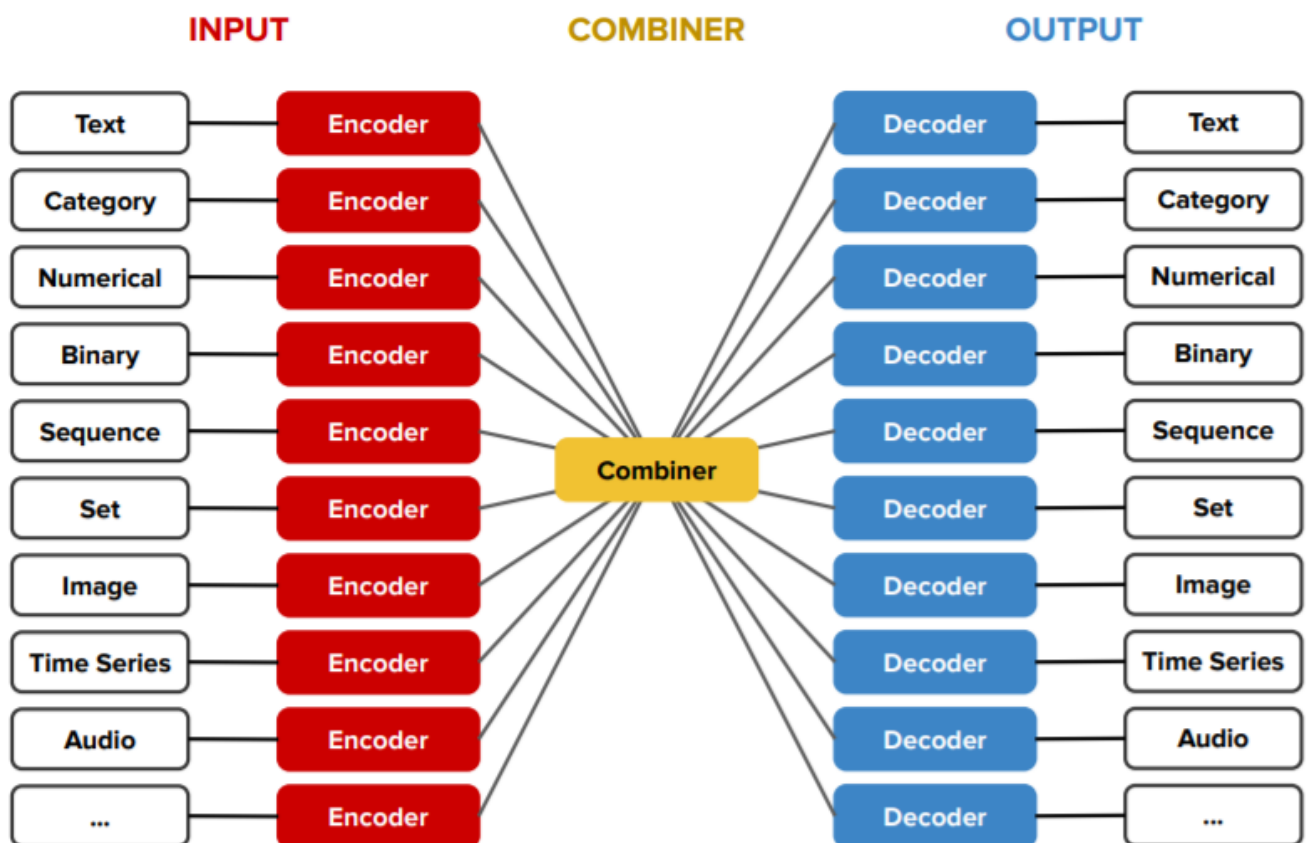


Figure 4: A graphical representation of the Encoder-Combiner-Decoder architecture (Molino et al., 2019).

Each model in Uber Ludwig is defined by encoders responsible for encoding various features of a particular input data point. The combiner's responsibility is to combine the information

originating from the multiple encoders whilst the decoders are responsible for decoding the information arising from the combiner into several output features.

The architecture of the Uber Ludwig framework is built on the concept of type-based abstraction. This framework supports several types: binary, category, numerical, text, date, audio, image, and vector. However, it is easy to add other types because of the type-based architecture. This type-based abstraction is implemented in Uber Ludwig because of the repetitive patterns in machine learning projects. These patterns are that pre-processing code is mostly similar for specific input types and the code used to implement the models. Therefore, by using modularization, it is possible to improve and reuse many code segments. Multi-task learning is another advantage of the architecture used to build Uber Ludwig. This has been very effective in computer vision and natural language processing, where more than one output feature is usually specified in the model.

Uber Ludwig uses a declarative model definition based on the Encoder-Combiner-Encoder architecture to train this model using the input data. This model definition comprises five components: input features, combiner, output features, pre-processing, and training. The input features are the component that holds a list of input features. The Combiner is the component where the type of combiner is specified. If one is not set, the default condition is *concat*. Output features are the component that is responsible for a list of output features. Any input or output feature that requires pre-processing or post-processing can be specified in the pre-processing component. The training component has several parameters that can be modified, such as the number of epochs to train the model, the batch size, the learning rate, and its scheduling.

The Uber Ludwig framework, unfortunately, also has several limitations. An example of a limitation is that this framework cannot currently implement Generative Adversarial Networks (GANs) as their architecture requires two models. In contrast, Uber Ludwig can only support one at this moment in time. Another limitation of Uber Ludwig is the declarative model definition itself. Hyper-parameter optimisation will be required as the number of encoders and their hyper-parameters increase. Fortunately, none of these limitations will have an impact on the study.

The strengths and advantages of this deep learning framework, such as its ease of use and its flexibility, as discussed previously, have made deep learning more accessible to many individuals. Uber Ludwig will be used as a tool to conduct a comparative study to ascertain the accuracy of the terrain classifier that will be built in the before-mentioned Keras API.

2.6 LEGO Mindstorms robots and Raspberry Pi Sense HAT modules

In this section, the LEGO Mindstorms robots and the Raspberry Pi Sense HAT modules used to collect the data for this study will be discussed. How these two components function, as well as their background, will be addressed.

Valk (2014) explains that LEGO Mindstorms is a set that is comprised of various building and electronic pieces that can be used to build robots. These robots make use of motors for grabbing objects or driving whilst using sensors to perceive their environment. Electronic cables are used to connect the motors to a component called an NXT brick. This NXT brick is a small computer responsible for controlling the various electronic components, such as the motors and sensors, that enable the robot to perform actions in its environment. An example of an NXT is shown in Figure 5.

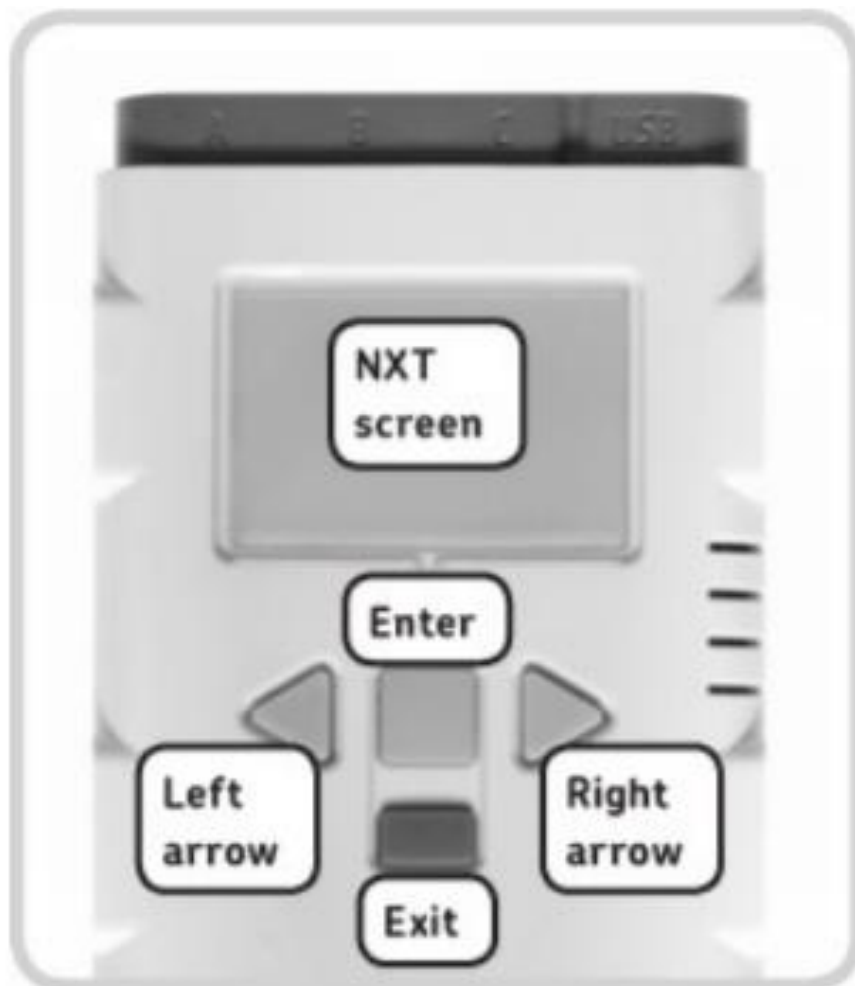


Figure 5: An example of a LEGO Mindstorms NXT brick (Valk, 2014).

This NXT brick can be turned on by pressing the Enter button, as indicated in Figure 5. After the enter button has been pressed, a start-up sound will play, and the NXT brick's main menu will appear on the liquid-crystal NXT screen. The left and right arrow buttons can be used to navigate the menu. The NXT brick can be switched off by pressing the Exit button. This small computer can be programmed by using the NXT-G programming software included with the LEGO Mindstorms package. This software allows an individual to write a program using a computer that contains all the necessary instructions that a robot can perform in its environment. This program can be transferred to a Lego Mindstorms robot via a USB cable. A program can then be run by selecting it on the NXT brick.

The data collected for this study was made possible by 'n Raspberry Pi with a Sense HAT module. The LEGO Mindstorms robot was used to traverse the different terrains, and the Sense HAT module took various recordings and logged these recordings. According to Nusairat (2020), a Sense HAT board can be placed on top of a Raspberry Pi by connecting the Sense HAT via the 40 GPIO pins. This allows the Sense HAT to interface with the Raspberry Pi and draw power from the Raspberry Pi. The Sense HAT board has many different features that make it possible for a Raspberry Pi to perceive the world around it with various sensors. These sensors include a barometer, a hygrometer, a gyroscope, a magnetometer, and an accelerometer. This Sense HAT module can be seen in Figure 6 and has already been connected to a Raspberry Pi.

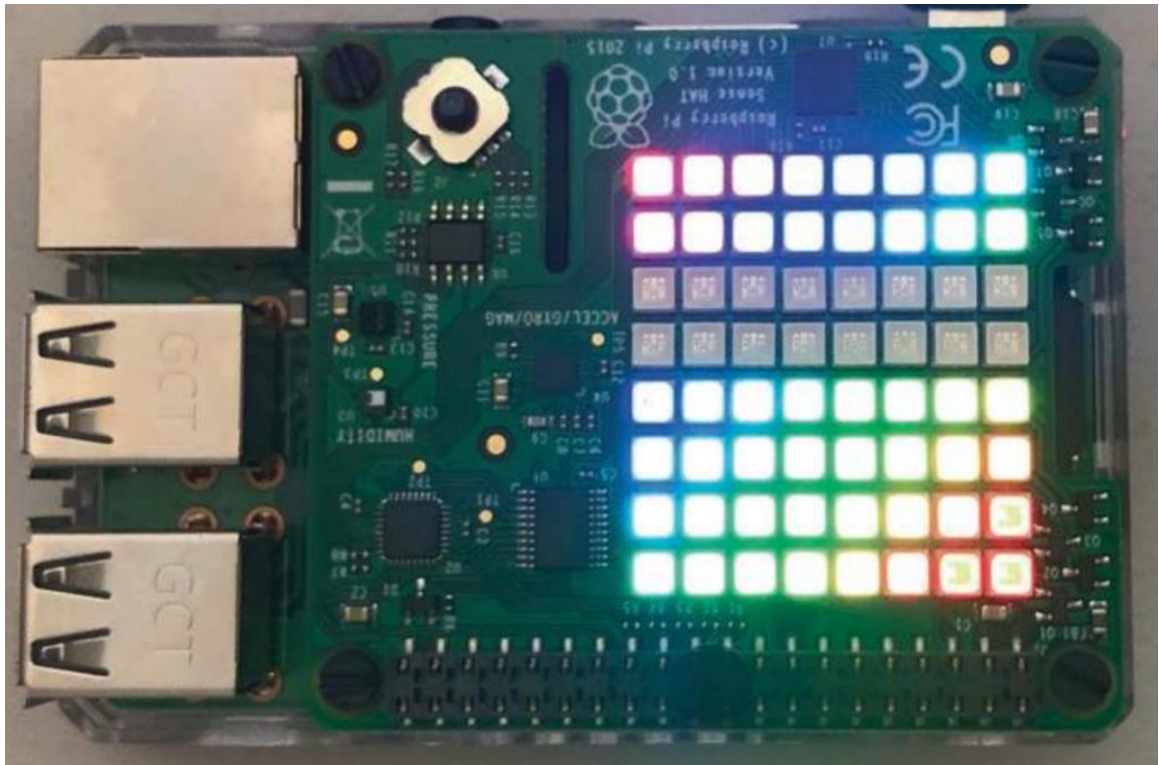


Figure 6: A Raspberry Pi with a Sense HAT module installed (Nusairat, 2020).

As can be seen from Figure 6, this Sense HAT module also has an 8x8 matrix display and a five-button joystick that can be seen in the top left corner. These two components allow interaction between it and the user.

LEGO Mindstorms allows an individual to build a robot that will satisfy a particular need. It will enable an individual to build a robot with LEGO bricks and therefore is a very versatile product that can be used in any environment. A Raspberry Pi with a Sense HAT module with these different sensors allows individuals to take exact scientific measurements of a domain. One sensor, in particular, the gyroscope, was very important for the data collection aspect of this study as the measurements made will be used to train the feed-forward neural network in the Keras API.

2.7 Conclusion to review

In this study, the importance of terrain classification has been highlighted because autonomous vehicles have become increasingly popular in the past decade and will only grow in popularity. As autonomous vehicles become more popular the safety, as well as the efficiency of these vehicles, will become more important. One method of increasing the efficiency as well as the

safety of these vehicles is called terrain classification. Therefore, developing a system that can accurately classify different terrains will significantly benefit these vehicles.

The purpose of this study is to build a feed-forward neural network capable of accurately classifying different terrains based on vibration readings. This study will make use of vibration readings induced on the vehicle through the different terrains. Terrain classification through vibration readings has shown to be the most accurate and effective method. The research that made use of vibration readings scored relatively high accuracy in the classification of different terrains. The terrain classifier will be built by making use of the Keras API. This API is flexible, well documented and open-source. The Keras API will be used to implement a feed-forward neural network model to classify different terrains based on vibration readings.

Uber Ludwig, a no-code deep learning framework, will be used to conduct a comparative study on the effectiveness of the model implemented in the Keras framework. Research has also been conducted on how the data has been collected in a previous study by a master's student. LEGO Mindstorms is a versatile product that can be used for many applications that suit an individual's needs. A robot built, using LEGO Mindstorms, combined with a Raspberry Pi with a Sense HAT module was used to collect the data that will be used to train the feed-forward neural network model in the Keras API.

2.8 Chapter summary

In this chapter, valuable literature that ties directly to this study was researched and presented. These topics range from different terrain classification methods previously used to feed-forward neural networks. The Keras API that will be used to build the feed-forward neural network was also discussed as well as the Uber Ludwig no-code deep learning framework. How the data was collected for this study was also discussed in terms of LEGO Mindstorms robots and the Raspberry Pi SenseHAT module.

CHAPTER 3 – DEVELOPMENT OF ARTEFACT

In this chapter, the development of the artefact will be discussed. The artefact will be discussed in detail as well as the life cycle that was used to develop this artefact.

3.1 Description of artefact

The artefact developed for this study is an automated terrain classification system using vibrations of a mobile robot measured while traversing six terrains and a feed-forward neural network. This was achieved by utilising the Keras Application Programmable Interface (API) for the neural network. A feed-forward neural network was trained to utilise this Application Programmable Interface (API). The neural network was trained by making use of the data that has been supplied by the supervisor of the study.

This data includes a number of input and output features. The input features are *temp_h*, *temp_p*, *pressure*, *humidity*, *yaw*, *pitch*, *roll*, *mag_x*, *mag_y*, *mag_z*, *acc_x*, *acc_y*, *acc_z*, *gyro_x*, *gyro_y*, *gyro_z*. These input features are used to train the feed-forward neural network. The output features are bricks, dirt, epoxy, grass, stone and tarmac. These output features are used to classify the different terrains based on the input features. This study is made up of two distinct sections namely the backend and the front-end. The backend will be discussed first and thereafter the front-end.

3.1.1 The artefact's backend

In order to develop an accurate model for the artefact, the KerasTuner hyperparameter optimisation framework was used. O'Malley (2019) states that KerasTuner provides a user-friendly, scalable framework that provides a solution to the painstaking task of searching for hyperparameters. KerasTuner provides several optimisation algorithms namely Bayesian Optimisation, Random Search, and Hyperband. This artefact utilised the Hyperband algorithm. According to Li *et al.* (2017), the Hyperband hyperparameter optimisation technique models the conditional probability $p(y|\lambda)$, of a model's performance on an evaluation metric y , such as validation accuracy, given a specific set of hyperparameters λ .

A custom KerasTuner class was written in order to also find the optimal batch size as well as the optimal number of epochs to train the neural network. The motivation behind creating a custom KerasTuner class is to be able to tune additional hyperparameters such as the batch size and the number of training epochs. A preferable batch size hyperparameter is found by specifying the boundaries. In this case, the boundaries are set at a minimum of 32 and a maximum of 256. The algorithm changes the batch size by 64 for each iteration until an optimal batch size is found. The

same is done for the optimal number of epochs. This custom KerasTuner class is used in combination with the Hyperband model.

In the Hyperband model, additional key hyperparameters are also specified that needs to be tuned. These include the learning rate of the model, the number of nodes for both the input layer of the model and the hidden layers of the model. A preferable learning rate is found by the model in the boundaries of $1e-2$, $1e-3$, and $1e-4$. Thereafter, the number of nodes for the input layer is optimised as well as the hidden layers of the Keras model. For the input layer as well as the hidden layers the boundaries are set at a minimum of 32 and a maximum of 128. The algorithm changes the number of nodes for these layers by a value of 16 for each iteration. The custom KerasTuner class in combination with the Hyperband model is then used to initialise a tuner.

The tuner is initialised with a Tuner class. In our case, this is the custom KerasTuner class that has been created in the first step. This class, in turn, takes several key parameters, each of which will now be discussed. The first parameter to be specified is the Hyperband model as discussed earlier. Thereafter the objective of the tuner is specified. In the case of the artefact, the objective was to increase the validation accuracy of the model. The next parameter to specify is the maximum number of epochs that the tuner will run, the tuner was instructed to run a total number of 20 epochs. A parameter, factor, is also specified which means that the maximum number of epochs is run three times per iteration. The final parameter is the directory in which the results of the tuner is stored.

The final step is to run the tuner's search function. This function takes a number of key parameters, such as the training samples, the training labels and the validation set. Once the search is finished the best model found can then be saved. This model can then be used to predict the terrain that is currently being traversed. A graphical representation of the best model found for the artefact can be seen in Figure 7.

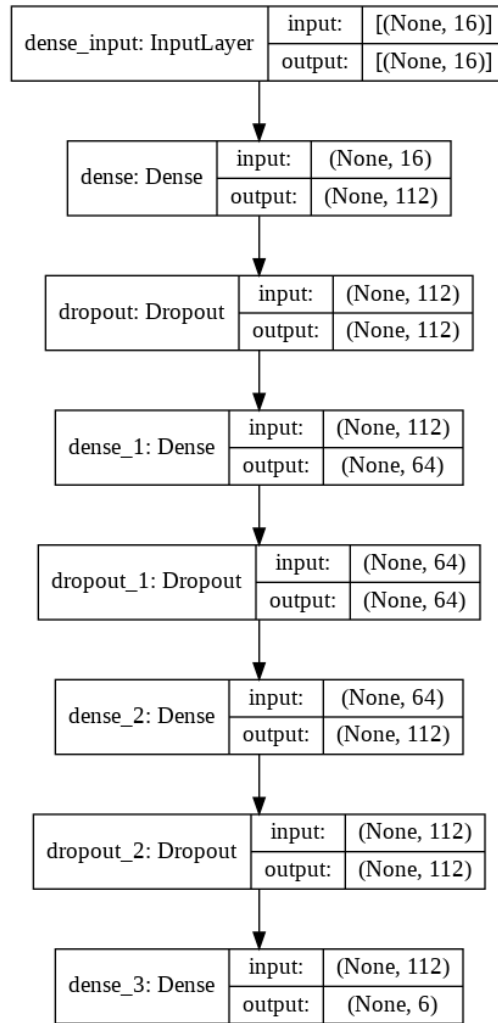


Figure 7: The resulting model of KerasTuner.

This model serves as the backend of the artefact. The input layer has a size of 16. This indicates that sixteen input features are required in order to make a prediction. However, the output layer has a size of 6. This represents the different terrains that can be classified. The terrains include bricks, epoxy, grass, dirt, stone and tarmac.

3.1.2 The artefact's front-end

A request was made by the supervisor of the study that the artefact is made more appealing and user-friendly. This was achieved by developing a front-end graphical user interface (GUI). Django is a high-level Python web development framework that can be used to serve python code on a web server. The reason this framework was chosen is the fact that once the web application is hosted, any individual can make use of this artefact. Another reason is the fact that it is based on the Python programming language, the same language that was used to create the Keras feed-

forward neural network model. The artefact's front-end graphical user interface can be seen in Figure 8.

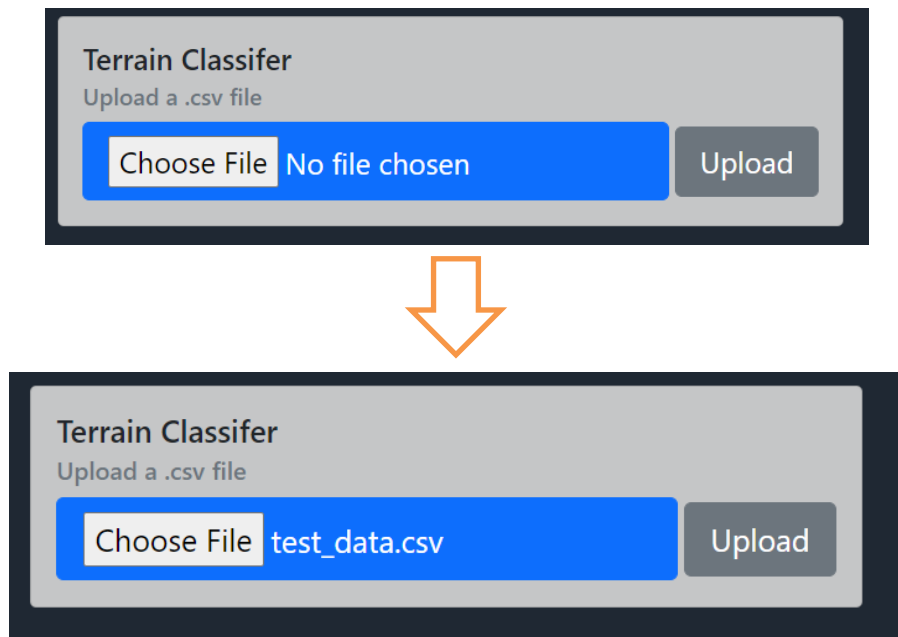


Figure 8: The Graphical User Interface (GUI) of the artefact.

As indicated in Figure 8, the user clicks the “Choose File” button, in which the user specifies a comma-separated value (.csv) file. The file contains all the necessary data in order for the Keras model to make a prediction of the type of terrain. The output of the prediction can be seen in Figure 9.

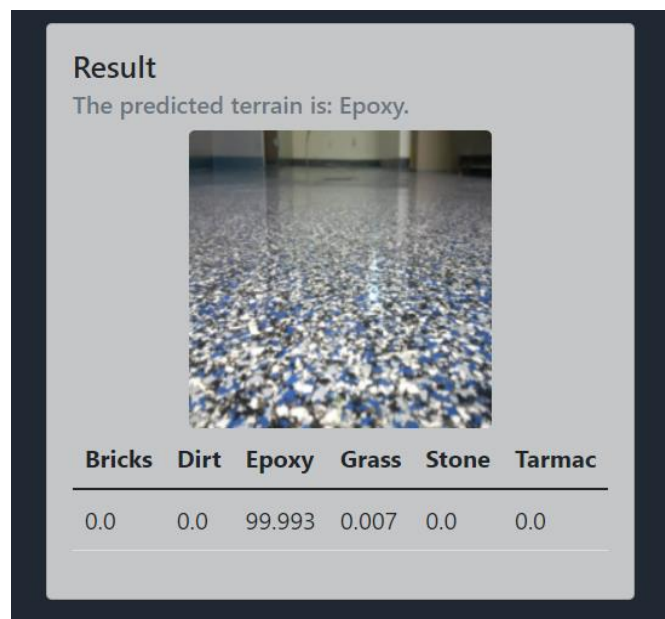


Figure 9: The result of the prediction

As indicated in Figure 9, the terrain has been predicted based on the data in the file that has been uploaded in the previous figure. In this case, the predicted terrain is Epoxy. An image is also shown to visualise the terrain as well as a table containing all the predictions the model made. As indicated from the table, epoxy has the largest probability of being the terrain that has been predicted by the supplied data.

3.2 Life cycle and different phases

3.2.1 Design science research

According to Hevner and Chatterjee (2010), the design science research paradigm is a paradigm in which the designer of a system answers questions relating to human problems through the creation of innovative artefacts. Through this, a designer can contribute new, innovative knowledge to the body of scientific evidence. A key characteristic of the artefacts is that they are useful and can be designed to resolve the problem. There are three design science research cycles as well as three environments in all design research projects as shown in Figure 10.

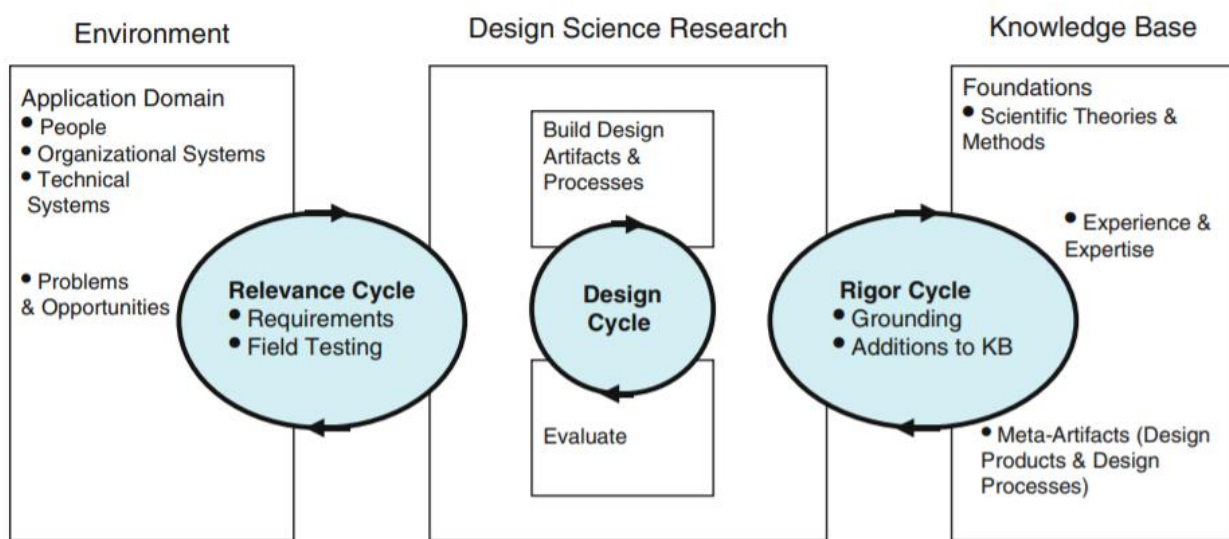


Figure 10: A graphical representation of the design science research cycles (Hevner & Chatterjee, 2010).

The relevance cycle as indicated in the above figure forms a bridge between the contextual environments and the design science research activities. Whereas the rigor cycle is a connection between the design science research activities and the knowledge base of scientific foundations. The design cycle moves between the primary activities of building and evaluating the artefacts and processes of the research that is being conducted. Each of these cycles will now be discussed in more detail.

The relevance cycle is characterised as being the motivation to improve a specific environment by introducing a brand new and innovative artefact and is concerned with the processes involved in building these artefacts. The rigor cycle, on the other hand, is responsible for providing past knowledge to the research project in question to support its innovation. It is important that researchers continue to reference and research the knowledge base in order to ensure that the designs that are produced contribute to the research in question. The primary focus of design science research is the design cycle. This cycle is responsible for the creation of the artefact, its evaluation as well as relevant feedback to improve the design even further.

3.2.2 Positivism research paradigm

Rehman and Alharthi (2016), states that the positivism research paradigm is reliant on experimentation. In this paradigm the hypothesis is presented in a question form, the empirical evidence is gathered; the total volume of the evidence is then analysed and formulated into a theory. The positivism research paradigm is characterised by deductive reasoning. The hypothesis is formed, then the hypothesis is either accepted or rejected depending on the statistical result of the analysis.

This paradigm often generated numerical data. This data is then used to both answer important research questions and formulate theories. This data is collected through experiments, standardised tests, surveys or questionnaires. The positivism research paradigm has four criteria that define whether the research conducted is of good quality. These criteria are whether the research has internal validity, reliability, external validity, and objectivity.

The reason this paradigm was selected is the fact that the artefact that has been built for this study will generate important empirical data such as the training accuracy, validation accuracy and prediction accuracy of the artefact, or more precisely the Keras feed-forward neural network. These results will be used to determine whether the artefact can be deemed as a success.

3.2.3 The artefact life cycle

As mentioned in Chapter 1, this study will make use of the agile software development lifecycle to develop the artefact. A subset of agile, scrum will be used to develop this artefact. Rubin (2021) states that Scrum is an approach that is capable of developing innovative deliverables and services. Scum is a simplistic, generic agile software development approach as indicated in Figure 11.

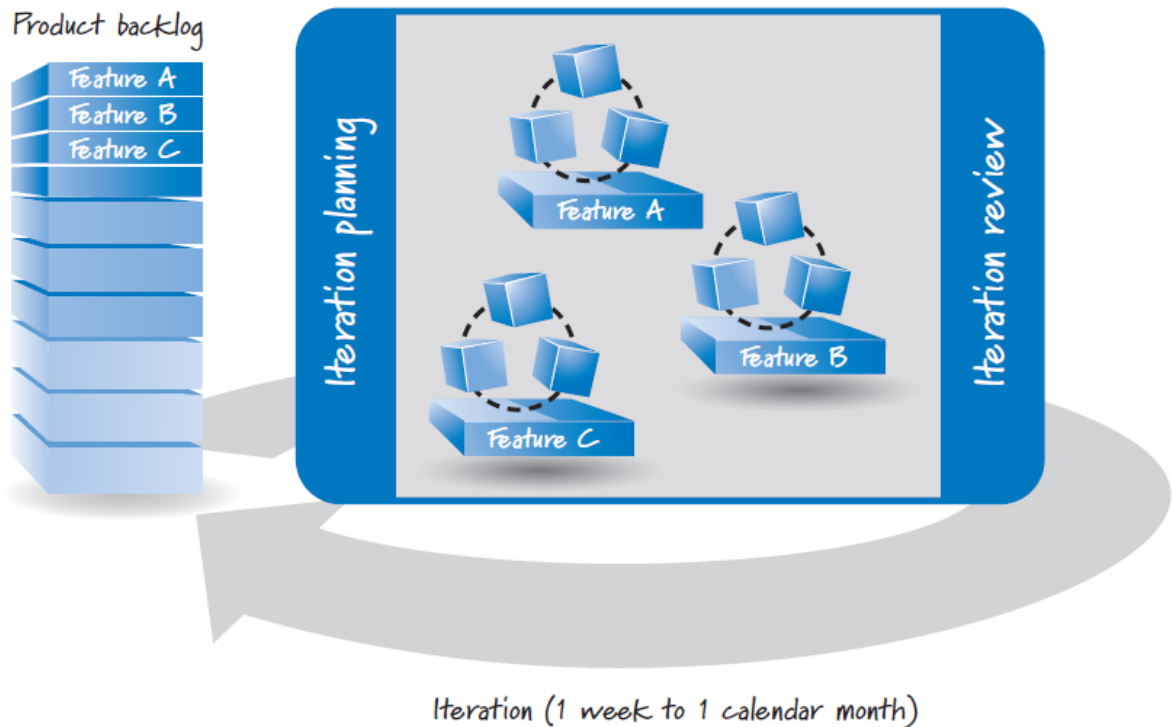


Figure 11: An overview of the scrum software development methodology (Rubin, 2012).

With this methodology, the process is initiated by developing a product backlog. A product backlog is a list of features that is a priority to develop a successful deliverable. The product backlog helps ensure that the most important items are worked on first. The development of a deliverable is conducted in short iterations, referred to as a sprint, that is usually conducted in a period of a week or even a month. At the end of each iteration, a review is conducted. In this review, the stakeholders of the project review the features that have been completed in the sprint to obtain the necessary feedback. Depending on this feedback, the planning for the next sprint is formalised. At the end of every sprint, there should be a prototype, or in other words an almost completed deliverable.

There are essentially five steps to be followed in the scrum methodology. These steps are creating a product backlog, planning the sprint, executing the sprint, a sprint review, and a sprint retrospective. Table 2 provides a summary of these steps.

Table 2: Summary of the steps of the scrum software development methodology.

Phase	Description of phase
1. Creation of product backlog:	The stakeholders of the project are responsible for determining, managing and prioritizing the sequence of tasks that needs to be completed.
2. Planning the sprint:	Determine what backlog items need to be built in the next sprint. A sprint goal is agreed upon by all the stakeholders of the project that determines what is to be achieved.
3. Executing the sprint:	All the necessary tasks as specified in the planning phase of the sprint is executed.
4. Sprint review:	The function of this step is to perform an inspection on the deliverable and adapt the deliverable based on the results of the inspection.
5. Sprint Retrospective:	This step occurs after the sprint review has been concluded and before the next sprint is planned. The primary goal of this step is to inspect and adapt the process. This is an opportunity for all the stakeholders to discuss what is working and what is not working with the Scrum. Once this step is completed the whole cycle is repeated starting with the next sprint planning session.

3.3 Artefact development description

In this section, the previously mentioned Scrum methodology will be implemented to develop the artefact. This includes the creation of the product backlog, planning the various sprints, executing these sprints, reviewing the sprints and also conducting a sprint retrospective.

3.3.1 Create the product backlog

The product backlog is a prioritized list that determines and manages the sequence of work (Rubin, 2012). For this artefact, five user stories were identified and were included in the product backlog. These user stories can be seen in Table 3.

Table 3: Product backlog for the artefact

User Story ID	User Story	Estimate (Size)	Sprint
US001	As a researcher, the data that has been supplied for this study needs to be combined, pre-processed, and divided into three sets: a training set, a test set, and a validation set for the model.	High	1
US002	As a researcher, I want to learn how to use the Keras API in order to build a feed-forward neural network model in order to classify terrains accurately using the data sets in US001.	High	2
US003	As a researcher, I want to save and export the trained model so that it can be used for other external applications by other researchers	Medium	2
US004	As a user, I would like an interface that is easy to use and can upload a file that contains the data necessary to classify a terrain based on vibrations.	Medium	3
US005	As a user, I would like to see a picture that indicates what terrain has been predicted by US004 as well as a table containing information on the accuracy of the prediction for each terrain.	Low	3

3.3.2 Planning the sprints

During this phase, the stakeholders of the project agree on a sprint goal for each sprint that is used to define what the next sprint is to accomplish (Rubin, 2012). In this project, sprints were divided into one-to-two-week durations to develop the artefact. Weekly meetings were held to discuss the progress of the project and also to discuss what should be finished next. This project made use of three sprints to develop the artefact. The first sprint's responsibility concerned the pre-processing of the data supplied for this study. The second sprint focused on building the feed-forward neural network, in the Keras API, for the artefact. The third sprint was responsible for building the front-end of the artefact so that users can interact with the feed-forward neural network built in the Keras API.

3.3.3 Executing the sprints

3.3.3.1 Sprint 1

Data pre-processing was identified as being crucial to this study as the data supplied for this study will affect the accuracy of the feed-forward neural network model. Therefore, the first sprint focuses on this as the artefact is directly dependent on this sprint. This sprint comprised three phases namely, combining the various data sets, performing the data pre-processing techniques, and dividing the data set into three smaller sets. The three smaller sets are the training set, the testing set, and the validation set. Chapter 4 places more detail on exactly how these three phases were conducted. Refer to this chapter for further information concerning these three phases.

3.3.3.2 Sprint 2

Developing the feed-forward neural network model in the Keras API was the focus of the second sprint. This was achieved by making use of Google's web-based development environment Google Colab. A few important Python libraries as well as Keras libraries were required to build the model. These include the TensorFlow library, the Keras library, the KerasTuner library, the pandas library, the NumPy library, and the sklearn library.

The TensorFlow library was required as it was used as the backend as it has been adopted by Keras as its official programming backend. Therefore, without this library, the model would not have been created. The Keras library was used to create the sequential model that was used to develop the feed-forward neural network. The KerasTuner library was used to tune the hyperparameters of the model. The pandas library was used to read in the pre-processed data set that was used to train the Keras sequential model. The sklearn library was used to split the

dataset into a training set and a testing set. The NumPy library was used to convert the pandas data frame's into NumPy arrays which the Keras model could use to train the sequential model.

All of these libraries together resulted in the feed-forward neural network model built in the Keras API. A summary of this model can be seen in Figure 12.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 112)	1904
dropout (Dropout)	(None, 112)	0
dense_1 (Dense)	(None, 64)	7232
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 112)	7280
dropout_2 (Dropout)	(None, 112)	0
dense_3 (Dense)	(None, 6)	678

```
Total params: 17,094  
Trainable params: 17,094  
Non-trainable params: 0
```

Figure 12: A summary of the feed-forward neural network built in the Keras API.

As indicated in Figure 12, the model is made up of seven layers. The first layer is the input layer that receives the input shape of the data. In this case, the input shape is 16, as 16 input features are used to train the model. There are two hidden layers as well as one output layer that has an output shape of six. The reason is that six terrains can be classified by the model. Between each layer, there is a dropout layer. This layer is used to ensure that the model does not overtrain.

Once the supervisor was satisfied with the model's performance it was saved so that it can be exported and used in the next sprint as the backend in order to classify terrains based on vibrations.

3.3.3.3 Sprint 3

In this sprint, the user interface, or front-end, of the artefact was developed. As indicated in the backlog, the requirements of this sprint are to provide the functionality to upload a file containing data that is used to predict one of the six terrains. Another requirement is to be able to present

the results of the prediction. This entails having a picture that represents the predicted terrain as well as a table containing the predictions made by the feed-forward neural network model built in Keras.

The user interface was built by making use of two technologies, Django and Bootstrap. As mentioned earlier Django is a high-level Python web development framework and was used to load the Keras model that has been saved in the previous sprint. Bootstrap was used to create the web pages that functioned as the interface with which the users can interact with the artefact. The sequence in which users interact with the user interface can be seen in Figure 13.

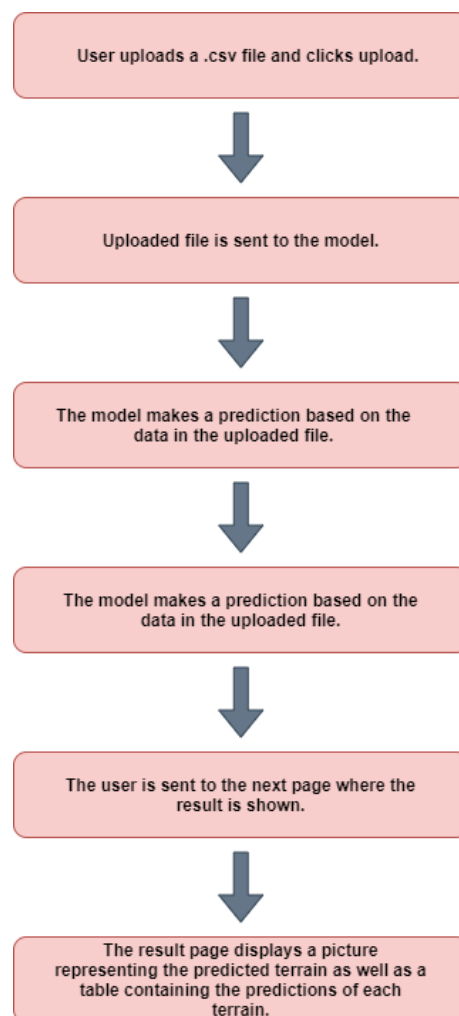


Figure 13: The Sequence of steps of the user interface.

The sequence of steps for the user interface is made up of six steps. First the user uploads a .csv file containing the necessary data in order for the model to make a prediction. This uploaded file is then sent to the model that has been exported in sprint 2. This model then makes a prediction

based on the file that has been uploaded in the first step. After this step, the user is sent to the next page that displays a picture representing the predicted terrain as well as a table containing the various predictions.

3.3.4 Demonstrate the artefact

At the end of each sprint, the artefact was demonstrated to the supervisor of the project. Depending on the feedback the next sprint was initiated, or the necessary changes indicated by the supervisor was implemented. In order to demonstrate the first sprint, descriptive statistics regarding the data before as well as after the data pre-processing was presented to the supervisor of the study. Once the supervisor was satisfied the next sprint was initiated.

The next sprint had continuous input from the supervisor regarding the structure of the model. The supervisor gave guidance to ensure that the construction of the model was heading in the correct direction. During each feedback meeting, the progress regarding the accuracy of the model was discussed as well as what can be done to further improve. Once the supervisor was satisfied with the result the next sprint was initiated.

The final sprint essentially combined all three sprints into the final artefact. Therefore, integration testing was crucial in this sprint in order to make sure that all the different modules fit together. Fortunately, since the front-end and the backend both are based on the Python programming language integration was not a problem. However, the supervisor requested minor changes to the user interface.

3.3.5 Sprint retrospective

At the end of every sprint, the progress of the artefact was evaluated. Based on the feedback either the next sprint was initiated, or changes were made to accommodate the requests made by the by supervisor. During the first sprint, different data pre-processing techniques were tested until the best technique for the artefact was identified. At the end of the first sprint, it was decided to use the StandardScaler of the sklearn library.

At the end of the second sprint, the model was exported to be used for the third sprint. No issues were identified during this sprint. However, various changes were required during the last sprint of the study. The user interface was first designed with multiple input fields, but the supervisor enquired that a file should be uploaded in order to make the interface more user friendly. The supervisor also requested that the table containing the various predictions should be changed from scientific notation to percentages.

3.4 Chapter summary

In this chapter, a brief overview concerning the development of the artefact was discussed. This artefact is comprised out of two components, a backend and a front-end. The backend is the feed-forward neural network constructed in the Keras API. This backend is responsible for classifying the different terrains based on vibration readings. The front-end is used to interact with this backend and was constructed in the Python web development framework referred to as Django. This artefact was designed and developed using the positivism research paradigm in combination with the design science methodology and the scrum methodology.

CHAPTER 4 – DATA COLLECTION AND PRE-PROCESSING

In this chapter, the techniques used during the data collection as well as pre-processing phases of the study will be discussed. An in-depth analysis of the data will also be conducted.

4.1 Data collection

As mentioned previously the data for this study was supplied by the supervisor of the study. The data had been generated in a previous study. The data was generated by using a LEGO Mindstorms robot in combination with a Raspberry Pi SenseHAT module. The Raspberry Pi SenseHAT module took readings with various sensors while the LEGO Mindstorms robot traversed the six different terrains.

4.2 Data pre-processing

The process of pre-processing the data for the feed-forward neural network model can be divided into three sections namely, combining all the data, pre-processing the data, and creating three smaller data sets. The three smaller data sets are the training set, the testing set, and the validation set for the model. Each of these sections will now be discussed.

4.2.1 Combining the data sets

In this section, a discussion on how the various data sets were combined into one large data set will be conducted. The data was received from the supervisor in separate data sets for each terrain. The various data sets were combined into one large dataset containing all the data for every terrain. Two fields were removed from the combined data set, count and time, as these two fields were not required to train the feed-forward neural network. In order to obtain more information about the fields in the data set, descriptive statistics about each field were calculated using the pandas library in the Python programming language. These statistics can be seen in Figure 14.

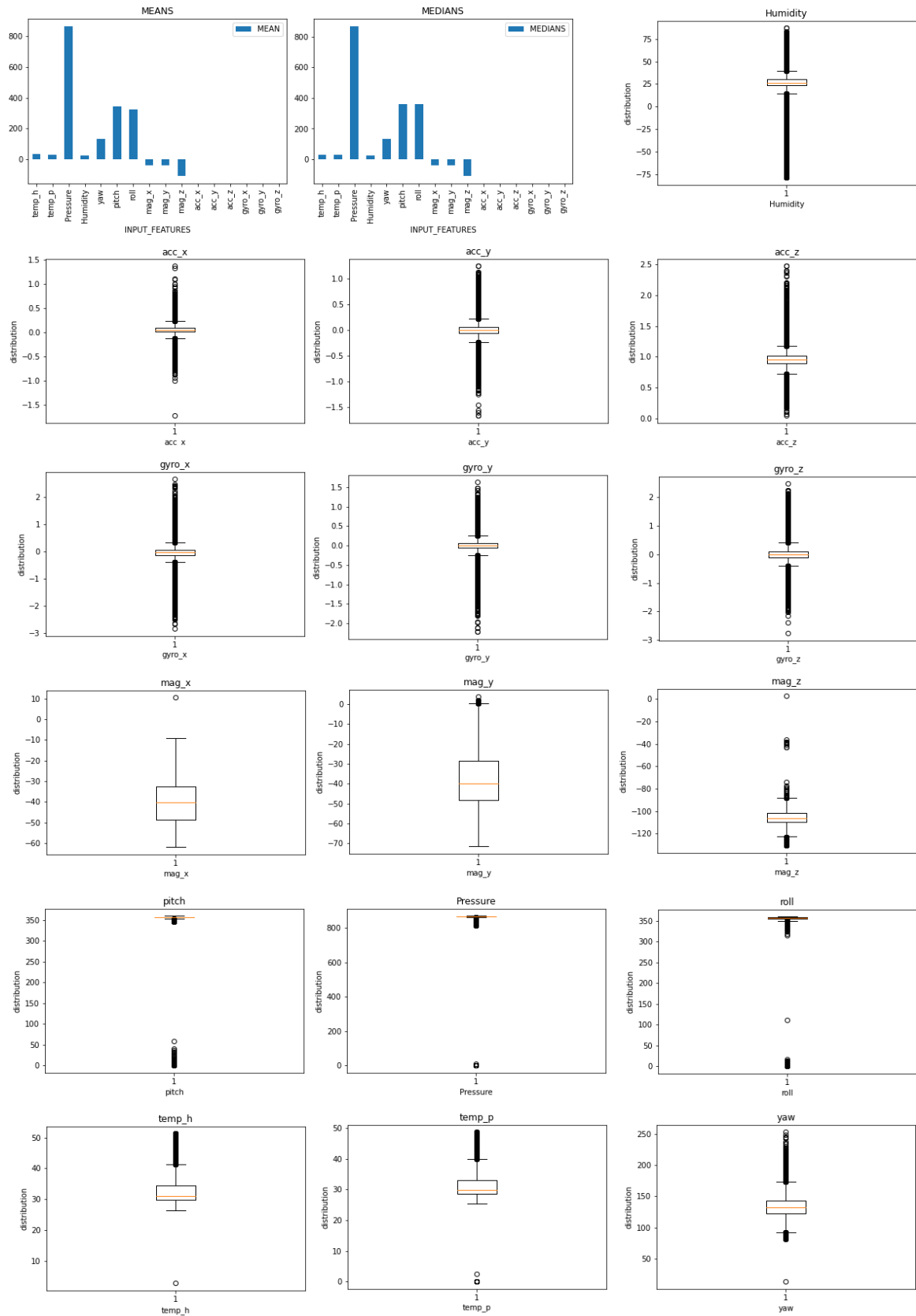


Figure 14: Descriptive statistics about the combined data set.

As indicated in Figure 14, the averages of each field vary by large margins from one another. The ranges of each field also vary by a large degree and the distribution of the data of each field also vary by some margin. Each field also have a number of outliers outside the quartiles of each field. These variances in the data could potentially hinder the effectiveness of the training of the model. Therefore, several data pre-processing techniques will be utilised to prepare the data set for training. This will be discussed in the next section.

4.2.2 Pre-processing the combined data set

In this section, a discussion on the various techniques used to prepare combined data set for training the model will be conducted. A sequence of steps was used to pre-process the data for the model. The first step was to standardize the data in the data set. This was achieved by utilising the `StandardScaler` class of the `sklearn` library.

According to Pedregosa *et al.* (2011), when using the `StandardScaler` class the features are standardized by removing the mean and scaling them to unit variance. The standard score of a sample is calculated as $z = (x - u) / s$. Where u is the mean of the training samples and s is the standard deviation of the training samples. Computing statistics on the samples centring scaling can occur independently of each feature. Standardizing a data set is a crucial requirement for many deep learning models, as the models could potentially behave unexpectedly due to the fact that individual features are not standard normally distributed data. Every input feature of the combined data set was standardized using the `StandardScaler` class.

Once the combined data set had been standardized using the `StandardScaler` class, the next step was to remove all the outliers in the standardized data set. This was achieved by using the `pandas` library. This was achieved by inspecting each individual value in each field and investigating if the value is outside the .15 and .85 quantile for each field. If this was the case, the value was removed from the data set.

The next step was to fill these removed values with the average of each field. This was also achieved by using the `pandas` library. The data set was inspected for empty values. Once an empty value was found it was replaced with the average of that field. The final step was to split the pre-processed data set into two data sets referred to as the input features and output features. The input features contain all the fields required to train the model and the output features contain all the different terrains that needs to be classified. The data set was now ready for the feed-forward neural network. The results of the data pre-processing can be seen in Figure 15.

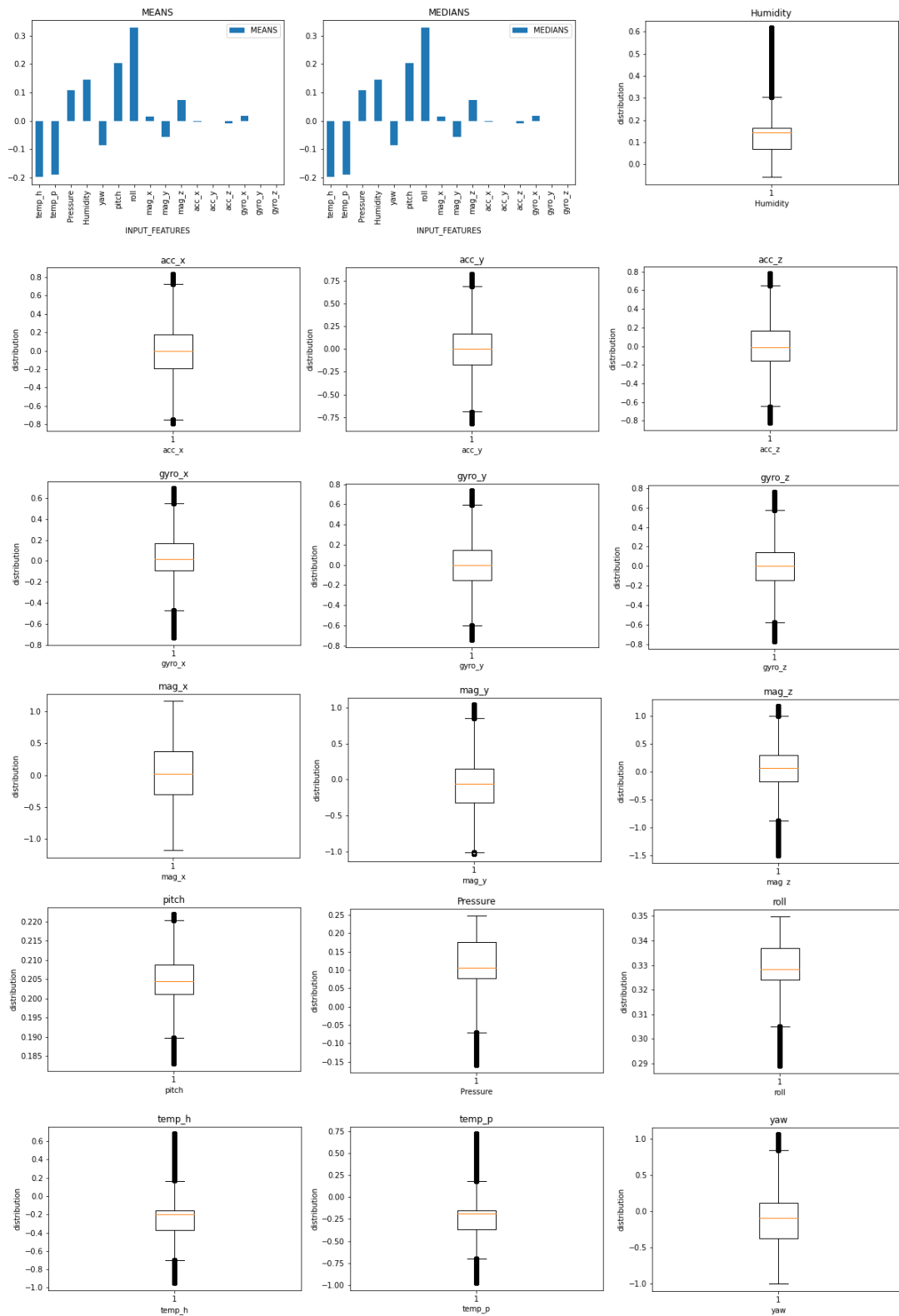


Figure 15: Descriptive statistics about the pre-processed data set.

As indicated in Figure 15, the averages of each field vary by considerably smaller margins from one another than in the original data set. The ranges of each field have been brought closer to each other and the distribution of the data of each field is more equally distributed. There are still some outliers in the data set, however, these are within the range specified in the previous section. The data set is now ready to be used for training the feed-forward neural network. However, the data set must be split into three smaller data sets, namely a training set, a testing set, and a validation set. The process of achieving this will be discussed in the next section.

4.2.3 Creating the three smaller data sets

In this section, a discussion on how the training set, testing set, and validation set had been created will follow. The training set and the testing set was generated by using the *train_test_split* function of the sklearn library.

Pedregosa *et al.* (2011), states that the *train_test_split* function divides arrays or matrices into random train and test subsets. This function receives four parameters. The first two parameters are the input features and output features that are read in from the pre-processed data sets. The next two parameters are used to specify the size of the test set as well as a random state which controls the shuffling applied to the data before the split operation is applied. The function returns four arrays containing the necessary data to train and test the Keras model. The validation set for the model is generated once the fit function is called when training the model.

4.3 Chapter summary

In this chapter, the various techniques used to prepare the data sets for the feed-forward neural network model to be constructed in the Keras API was discussed. Some key techniques worth mentioning is the StandardScaler class provided by the Sklearn library as well as the removal of outliers in the data set. This step in the development of the artefact was crucial as the data directly affects the effectiveness of the training of the model.

CHAPTER 5 – RESULTS

In this chapter, the results obtained from the artefact developed in this study will be presented. The results will be compared by using the Uber Ludwig no-code deep learning framework. This framework will be used as a yardstick to determine the success of the artefact.

5.1 Introduction

An artefact that can accurately predict various terrains using vibration data was developed, as a direct result of the research conducted on the various topics surrounding this artefact. The purpose of this chapter is to evaluate the success of the artefact. As indicated in Section 1.3, the success of the artefact is dependent on whether it can accurately classify the six different terrains.

To determine whether the artefact has succeeded in this regard, it will be compared to a model built in the open-source no-code deep learning framework known as Uber Ludwig. In Section 5.2, a comparative study is made between the results of the artefact and the model built in Uber Ludwig.

5.2 Comparative study between artefact and Uber Ludwig model

In this section, the accuracy of the artefact will be compared to the accuracy of the Uber Ludwig model. In this comparison, a few key hyperparameters will be identical in order to ensure that the comparison is as accurate as possible. These hyperparameters include the number of epochs as well as the batch size. The validation accuracy metric will be used as a yardstick to compare the two different models.

The artefact's test accuracy has been obtained and scored an accuracy of 98.27 per cent. A representation of the training accuracy, as well as the validation accuracy over the epochs, can be seen in Figure 16.

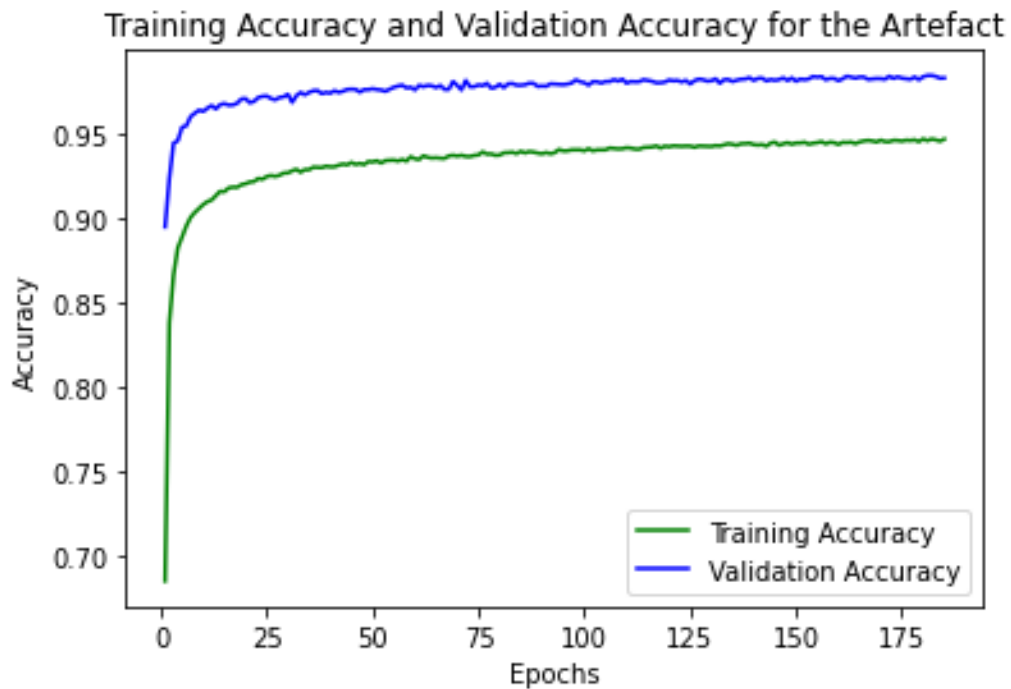


Figure 16: Training and validation accuracy over the epochs for the artefact.

From the figure above, it is evident that the artefact's training and validation accuracy steadily grows as the epochs progress. The gap between the training and validation accuracy is relatively small and therefore this model can be deemed a success. The figure shows that the artefact has comparable skill in accurately classifying different terrains based on vibration readings.

The validation accuracy of the Uber Ludwig model will now be analyzed. The model built in the Uber Ludwig framework achieved an overall test accuracy of 84.35 per cent. A representation of the training accuracy, as well as the validation accuracy over the epochs, can be seen in Figure 17.

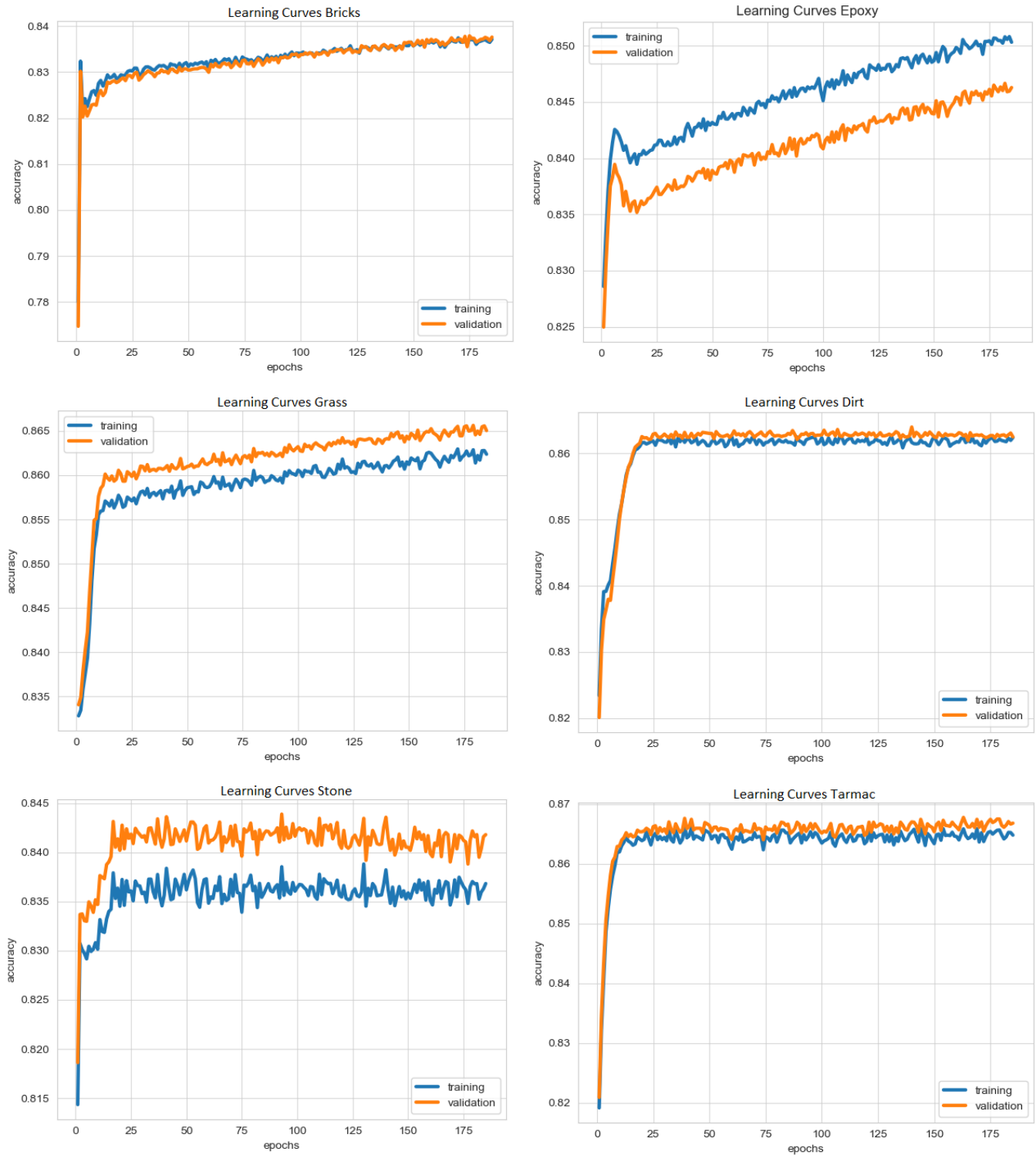


Figure 17: Training and validation accuracy over the epochs for the Uber Ludwig model.

Unfortunately, Uber Ludwig’s visualisation tools do not provide a graph that summarizes the entire training process as the Keras API. However, as indicated in the figure above it is evident that the Uber Ludwig model’s training and validation accuracy grows as the epochs progress. There is much more variance between the training and validation accuracy for this model than that of the

artefact. Therefore, it can be concluded based on the results of both models that the artefact is superior to the model created in the Uber Ludwig. Thus, this study was successful in its aims and objectives.

5.3 Chapter summary

From the results presented in Section 5.2 it is clear that the artefact created in this study is capable of accurately classifying different terrains based on vibration readings. The purpose of this chapter was to evaluate the success of the artefact built for this study. The study in which the artefact was compared to a model created in the Uber Ludwig no-code deep learning framework concluded that the artefact was successful in its aims and objectives as stipulated in Section 1.3.

The next chapter, Chapter 6, will provide a summary of the study as well as the opinion of the researcher regarding the success of the study.

CHAPTER 6 - REFLECTION

In this chapter, a reflection of the entire study will be conducted. In this reflection, a discussion on the experience gained from conducting this study will be followed as well as a discussion on whether the aims and objectives of the study were fulfilled. Thereafter, a discussion on the management of the project will be conducted and an evaluation of the entire process that had been followed in the study will also be conducted.

6.1 Experience gained from the study

In this study, a number of new skills have been learned. These include the ability to write a research proposal as well as a literature review. This new skill has led to the development of academic writing as well as how to structure formal research.

In terms of the process followed during the development of this artefact, in retrospection, a few strong and weak points have been identified. The Keras API is relatively easy to work with and the researcher was able to build a feed-forward neural network with minor issues. However, the hyperparameter optimisation proved to be more difficult than anticipated. The hyperparameter optimisation for the model proved to be the primary concern to ensure that the artefact could be deemed successful. The data pre-processing also proved to be a challenge to the researcher. The data for neural networks proved to be the most important aspect of the entire study. The standardization of the data is crucial to construct a feed-forward neural network that can be deemed successful in its specific task.

The researcher identified a few key items that should have been done differently. The data pre-processing phase of the study should have had more time and resources allocated to this phase as this proved to be a major issue when training the feed-forward neural network. Therefore, in future studies, the researcher will ensure that more time and energy be placed on this issue. The researcher also identified that the building of the artefact should have commenced earlier than scheduled in the project plan. Neural networks take a considerable amount of time to optimise their hyperparameters as well as to train them. Therefore, in future studies, the researcher would ensure that the construction of neural network models is done much earlier.

6.2 Evaluation of aims and objectives

The aims and objectives as stated in Section 1.3 will now be discussed and evaluated. A detailed evaluation of the aims and objectives can be seen in Table 4 below.

Table 4: Evaluation of the aims and objectives of the study.

Aims and objectives:	Completed (Yes/No):	Comments:
1. Literature review on key concepts regarding the study.	Yes	The literature review is completed and is included in Chapter 2.
2. Data set obtained and modified to suit the study's needs.	Yes	The data set has been obtained by the supervisor of the project and has been modified as indicated in Chapter 4.
3. Construction of an Uber Ludwig feed-forward neural network model.	Yes	The model was constructed to be able to compare this model with the artefact.
4. Train the Uber Ludwig feed-forward neural network model.	Yes	The model was trained using the data set modified in aim and objective 2.
5. Construction of a feed-forward neural network in the Keras API.	Yes	The artefact was constructed in the Keras API.
6. Train the Keras feed-forward neural network.	Yes	The model was trained using the data set modified in aim and objective 2.
7. Compare the accuracy of the Keras API model with the Uber Ludwig framework model	Yes	The two models were compared to each other based on their validation accuracy as indicated in Chapter 5.
8. Conclude on the artefact's accuracy and reflect.	Yes	The artefact's accuracy was demonstrated and a reflection on the study is evident.

As indicated in Table 4, all the necessary aims and objective specified in Section 1.3 has been met and documented throughout the study. Therefore, the artefact, as well as the study, can be viewed as a success.

6.3 Management of the study

The researcher is of the opinion that there were no major issues in meeting the target dates of the study. Every deliverable of the study was handed in to the supervisor of the study one to weeks prior to the submission of each deliverable. This was done so that the supervisor could provide feedback about the deliverable and that the necessary changes could be made to the deliverable.

6.4 Concluding remarks about the study and future work

The study, overall, was successful and all of the aims and objectives of the study was met as indicated in Section 6.2. However, there were some minor issues during the development of the artefact. The deliverables were completed on time, but the researcher experienced some issues with the various deep learning frameworks as the researcher was not as well acquainted with the various frameworks.

Although a feed-forward neural network was implemented, the researcher also learned about other related deep learning models in the process. The researcher also learned how to use various other tools in the Python programming language. These skills include how to pre-process data sets for neural network models as well as how to visualise data in different plots and graphs. The researcher also learned about the potential dangers of scope creep and that keeping to the original aims and objectives of the study was crucial in completing this study successfully.

This study could be extended in the future by creating an autonomous self-driving vehicle that is capable of accurately classifying the terrain it is currently traversing. The benefit of this is that the vehicle could automatically adjust the torque settings of its wheels to more effectively traverse the given terrain.

6.5 Chapter summary

In this chapter, a reflection on the study was conducted. The various new skills that had been learned as well as the problems experienced during the study was discussed. Thereafter, an evaluation on whether the various aims and objectives of the study had been met was highlighted. The management of the study was also discussed as well as the concluding remarks concerning the study. The potential to extend this study for future work was also discussed.

BIBLIOGRAPHY

- Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010, June). Agile software development: Impact on productivity and quality. In 2010 *IEEE International Conference on Management of Innovation & Technology* (pp. 287-291). IEEE.
- Bai, C., Guo, J., & Zheng, H. (2019). Three-dimensional vibration-based terrain classification for mobile robots. *IEEE Access*, 7, 63485-63492.
- Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27-31.
- Bhalerao, S., Puntambekar, D., & Ingle, M. (2009). Generalizing Agile software development life cycle. *International journal on computer science and engineering*, 1(3), 222-226.
- Brooks, C. A., & Iagnemma, K. (2005). Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6), 1185-1191.
- Brownlee, J. (2016). *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.
- Dupont, E. M., Moore, C. A., Collins, E. G., & Coyle, E. (2008). Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24(4), 337-347.
- Fine, T. L. (2006). *Feedforward neural network methodology*. Springer Science & Business Media.
- Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.
- Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9-22). Springer, Boston, MA.
- Hutson M. (2019). Bringing machine learning to the masses. *Science* (New York, N.Y.), 365(6452), 416–417. <https://doi.org/10.1126/science.365.6452.416>

Jiang, Z., & Shen, G. (2019, November). Prediction of house price based on the back propagation neural network in the keras deep learning framework. In *2019 6th International Conference on Systems and Informatics (ICSAI)* (pp. 1408-1412). IEEE.

Komma, P., Weiss, C., & Zell, A. (2009, May). Adaptive bayesian filtering for vibration-based terrain classification. In *2009 IEEE International Conference on Robotics and Automation* (pp. 3307-3313). IEEE.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1), 6765-6816.

Molino, P., Dudin, Y., & Miryala, S. S. (2019). Ludwig: a type-based declarative deep learning toolbox. *arXiv preprint arXiv:1909.07930*.

Nusairat, J. F. (2020). *Rust for the IoT: building Internet of Things apps with Rust and Raspberry Pi* / Joseph Faisal Nusairat. (1st ed. 2020.). Apress.

Park, Y. S., Konge, L., & Artino, A. R. (2020). The positivism paradigm of research. *Academic Medicine*, 95(5), 690-694.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Rehman, A. A., & Alharthi, K. (2016). An introduction to research paradigms. *International Journal of Educational Investigations*, 3(8), 51-59.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: a modern approach* (Fourth, Ser. Pearson series in artificial intelligence). Pearson.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.

Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. *Procedia Computer Science*, 181, 746-756.

Valk, L. (2014). The LEGO Mindstorms NXT 2.0 Discovery Book: a Beginner's Guide to Building and Programming Robots,(2010) p. cm. *Includes index*. ISBN-13, 978-1.

Wang, S. (2019). *Road Terrain Classification Technology for Autonomous Vehicle*. Berlin, Germany: Springer.

Weiss, C., Frohlich, H., & Zell, A. (2006, October). Vibration-based terrain classification using support vector machines. In *2006 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4429-4434). IEEE.

Wieringa, R. J. (2014). What is design science?. In *Design science methodology for information systems and software engineering* (pp. 3-11). Springer, Berlin, Heidelberg.

APPENDIXES

7 Appendix A: Research Ethics Form




ITRI 671 Research ethics form: Honours project

Name: Hendrik Fick Renaldo Potgieter

Title of project: Automated terrain classification using vibrations and a feed-forward neural network.

Supervisor: Prof. Tiny du Toit

Starting and end dates of project: 18/02/2021 – 01/11/2021

1. Have you read the information available related to research ethics (Chapter 5 of Researching Information Systems and Computing; BJ Oates and Chapter 13 of Writing for computer science, J Zobel; Manual for post graduate studies, available on efundi)?	Yes 	No
2. Do you make use of people as source of data in your project (for example the completion of questionnaires or evaluation of products)?	Yes	No 
3. Are there any aspects of your research that you need permission from another party to use (for example use of property or tools)? If yes, provide more detail. The data provided to train the deep learning algorithm was collected by a Masters student in the previous year. The supervisor of this project has already obtained permission to use this data.	Yes 	No

4. Describe your research question and give a short description of your plans for the collection of data.

The primary purpose is to develop a feed-forward neural network to classify terrains based on a dataset's vibrations. The dataset will be divided into three separate datasets: *the training set*, *the validation set* and *the test set*. These three datasets will be used as input to train the feed-forward neural network. The neural network will classify several terrains based on the datasets. Therefore the question this project will attempt to answer is: how can a feed-forward neural network be constructed in order to classify multiple terrains accurately?

The supervisor of this project provides the dataset. Therefore, no collection of data will be necessary for this project.

5. Describe how you plan to provide information about yourself and the goals of your research to participants.

Not applicable. There are no participants involved in this project.

6. Describe what methods you will use to get permission from participants in your study.

Not applicable.

7. Will you be able to ensure that participants' information will be used in an anonymous, private and confidential way? How?

Not applicable.

Yes

No

8. Are there any foreseeable risks of damage (physical, social or psychological) to participants or the environment? If you answer yes, give detail of the preventative measures you will follow.

Not applicable.

Yes

No

9. Are there any foreseeable risks to the NWU, for example lawful actions that may follow the research, or damage to the image of the university? If yes, give detail.	Yes	No ✓
--	-----	-------------

10. Are there any other ethical issues that may occur during the execution of the research (for example conflicting interests)? If yes, provide detail and explain how you plan to handle them.	Yes	No ✓

I declare that the information contained in this form is accurate. I have attempted to identify the risks that may arise in conducting this research and acknowledge my obligations and the participants' rights. I confirm that the study will be conducted in line with all University legal and ethical standards.

Name of student: HFR Potgieter

Signature: 

Date: 14/04/2021

Name of study leader:

Signature:

Date:

Name of additional moderator:

Signature:

Date:

8 Appendix B: Research proposal form honours project

SUBJECT GROUP COMPUTER SCIENCE AND INFORMATION SYSTEMS

RESEARCH PROPOSAL FOR HONOURS PROJECT

The student and the supervisor must consult the *Manual for Postgraduate Studies* prior to writing the research proposal. The *Manual for Postgraduate Studies* explains in detail what is expected at each of the subheadings below. The proposal should not be longer than 5 pages.

The Subject Group requires that the research proposal will be submitted through the use of this form and in the format below. Please complete using a computer.

1. Student initials, surname and student number

Initials	HFR	Surname	Potgieter	Student number	29971349
----------	-----	---------	-----------	----------------	----------

2. Degree for which student is registered

B.Sc. Hons in Computer Science and Information Systems
--

3. Name of supervisor

Initials surname	and	Prof. JV du Toit
---------------------	-----	------------------

4. Proposed title

Title (preferably not more than 12 words)	Automated terrain classification using vibrations and a feed-forward neural network
---	---

5. Problem statement and substantiation

Autonomous ground vehicles are employed in many different operational fields like supply and logistics, surveillance, search and rescue missions, and agricultural applications. These operations may be necessary to traverse some indoor or off-road terrain, affecting the vehicle's performance. The efficiency of these vehicles can be improved by detecting their environment. In any outdoor setting, the surface is inherently diverse, some territories may be flat and smooth, and some may be rugged and bumpy. Thus, the surfaces outdoor can be described as hazardous environments and therefore, any autonomous vehicle must know what kind of surface it is currently traversing (Weiss *et al.*, 2006). Autonomous vehicles will play an essential role in the transport industry in the future. Therefore, developing a system that can classify terrains autonomously will be very beneficial for these autonomous vehicles.

However, the problem is to build a feed-forward neural network model to train a deep learning algorithm to classify terrains accurately with the data supplied by the supervisor.

However, the researcher will have to use pandas to manipulate the dataset that the supervisor of this project has supplied to ensure that the data is in the correct format.

The question this project will attempt to answer is: how can a feed-forward neural network be constructed in order to classify multiple terrains accurately?

6. Research aims and objectives

The primary objective of this project is to construct a feed-forward neural network that can accurately classify terrains based on the input data provided by the supervisor.

1. A thorough study must be conducted on the primary concepts in regards to the following:
 - a. Terrain classification systems.
 - b. Feed-forward neural networks
 - c. Keras open-source software library.
 - d. The Uber Ludwig deep learning framework.
 - e. Previous attempts of building terrain classification systems using feed-forward neural networks will be studied.
2. Obtain the dataset provided by the supervisor that will be used for this project:

- a. Modify the dataset to meet the requirements of the supervisor previously mentioned in the "Problems that can be addressed"-section.
 - b. Split the dataset into a training set and a testing set to train the Uber Ludwig framework.
3. Use this data set to construct a feed-forward neural network using the Keras library.
4. Train the feed-forward neural network.
5. Use this dataset to construct the model definition for Uber Ludwig.
6. Train the Uber Ludwig deep learning framework.
7. Compare the accuracy of the Uber Ludwig framework with the Keras library.
8. Conclude on the artefact's accuracy and reflect on the insights obtained and any potential shortcomings identified.

7. Basic hypothesis (where applicable)

Not Applicable.

8. Method of investigation

8.1 Literature study

A literature study will be conducted on the following concepts (more sources may also be included later):

1. Lego Mindstorms robots – a brief overview of these robots and how they work. This is necessary as these robots were used to collect the data provided by the supervisor. Klassner and Anderson (2003), explain the overall workings of Lego Mindstorms. Other sources may be included.
2. Raspberry Pi Sense HATs – an introduction to how these sensors work. This is applicable because this sensor was used to collect the data provided by the supervisor. Chatterjee and Debnath (2018) explain the primary sensors and functions of Raspberry Pi Sense HATs. Other sources may be included.
3. Terrain classification DuPont *et al.* (2008), classifies various terrains using a deep learning algorithm. Other sources on terrain classification may be consulted.
4. Feed-forward neural networks – Fine (2006), explains the overall architecture of feed-forward neural networks. Additional sources may be included.
5. The Keras library – Géron (2019), includes an explanation on building feed-forward neural networks in Keras. Additional sources may be consulted.

6. The Uber Ludwig framework -Molino *et al.* (2019), explains the overall structure of the Uber Ludwig is Other sources may be included

8.2 Methods of investigation

Design science is the study and design of artefacts built to interact with a problem to improve something in the problem's context (Wieringa, 2014). According to Hevner and Chatterjee (2010), design science is a research paradigm wherein answers to questions are provided using an artefact in context to human problems. Therefore, this project will use the design science research methodology as the primary goal of this project is to build an artefact that provides a solution to a problem in a context. The design science research methodology has six steps: problem identification and motivation, identifying the objectives for the solution, the design and development of the artefact, and demonstration, evaluation, and communication (Hevner & Chatterjee, 2010). Therefore, these steps will ensure that a satisfactory artefact is designed and developed.

Since the accuracy and effectiveness of the artefact can be measured, an experimental research method will be used. Therefore, the positivism research paradigm will be used. The core function of the positivism research paradigm is to be able to make a prediction based on the phenomena in question (Park *et al.*, 2020).

The supervisor provides the dataset that will be used for this project. This dataset will be divided into three smaller datasets: the training set, the validation set and the test set. These three datasets will be used to train, validate and test the feed-forward neural network model.

To build the artefact, the Agile software development life cycle will be utilised. According to Ahmed *et al.* (2010), agile means active, swift, and responsive, and this is the core definition of this methodology. This software development life cycle has several incremental and iterative phases (Bhalerao *et al.*, 2009). Therefore, if changes are made during the artefact's development, they can be made swiftly. This approach also allows stakeholders to react quickly to changes that might be required and provide the necessary flexibility to accommodate quick changes (Thesing *et al.*, 2021). Thus, the supervisor will always be able to provide regular feedback on the artefact's development progress.

9. Provisional chapter division

The final project document will be divided into several chapters.

General:

- Title page
- Table of contents

- List of figures
- List of tables

The provisional chapter division is as follows:

Chapter 1. Introduction

A brief description of the problem being researched, building an automated terrain classification neural network using vibration readings, is discussed in this chapter. The necessary information regarding the background of the study is highlighted, the aims of the study are brought forth, the procedures, as well as the methods that will be utilised, is discussed, the project plan is presented, and the ethical and legal implications are explained.

Chapter 2. Literature review

This chapter is a discussion on related literature to the study. These include key terms such as terrain classification, feed-forward neural networks and any other relevant technology that will be used to build the artefact.

Chapter 3. Development of artefact

Chapter 3 focuses on the creation of the artefact. This discussion includes the development process of the artefact as well.

Chapter 4. Data collection and pre-processing

The various methods utilised to gather, process and analyse the data that will be used to train the feed-forward neural network is discussed in this chapter.

Chapter 5. Results

The results obtained from the study are discussed and reflection on the results will also be discussed.

Chapter 6. Reflection

A brief reflection on what has been learned in this study will be brought forth.

10. Literature references

- Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010, June). Agile software development: Impact on productivity and quality. In *2010 IEEE International Conference on Management of Innovation & Technology* (pp. 287-291). IEEE.
- Bai, C., Guo, J., & Zheng, H. (2019). Three-dimensional vibration-based terrain classification for mobile robots. *IEEE Access*, 7, 63485-63492.
- Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27-31.
- Bhalerao, S., Puntambekar, D., & Ingle, M. (2009). Generalizing Agile software development life cycle. *International journal on computer science and engineering*, 1(3), 222-226.
- Brooks, C. A., & Iagnemma, K. (2005). Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6), 1185-1191.
- Brownlee, J. (2016). *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.
- Dupont, E. M., Moore, C. A., Collins, E. G., & Coyle, E. (2008). Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24(4), 337-347.
- Fine, T. L. (2006). *Feedforward neural network methodology*. Springer Science & Business Media.
- Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.
- Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9-22). Springer, Boston, MA.
- Hutson M. (2019). Bringing machine learning to the masses. *Science* (New York, N.Y.), 365(6452), 416–417. <https://doi.org/10.1126/science.365.6452.416>
- Jiang, Z., & Shen, G. (2019, November). Prediction of house price based on the back propagation neural network in the keras deep learning framework. In *2019 6th International Conference on Systems and Informatics (ICSAI)* (pp. 1408-1412). IEEE.
- Komma, P., Weiss, C., & Zell, A. (2009, May). Adaptive bayesian filtering for vibration-based terrain classification. In *2009 IEEE International Conference on Robotics and Automation* (pp. 3307-3313). IEEE.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1), 6765-6816.
- Molino, P., Dudin, Y., & Miryala, S. S. (2019). Ludwig: a type-based declarative deep learning toolbox. *arXiv preprint arXiv:1909.07930*.

- Nusairat, J. F. (2020). *Rust for the IoT: building Internet of Things apps with Rust and Raspberry Pi* / Joseph Faisal Nusairat. (1st ed. 2020.). Apress.
- Park, Y. S., Konge, L., & Artino, A. R. (2020). The positivism paradigm of research. *Academic Medicine*, 95(5), 690-694.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Rehman, A. A., & Alharthi, K. (2016). An introduction to research paradigms. *International Journal of Educational Investigations*, 3(8), 51-59.
- Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.
- Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: a modern approach* (Fourth, Ser. Pearson series in artificial intelligence). Pearson.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
- Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. *Procedia Computer Science*, 181, 746-756.
- Valk, L. (2014). The LEGO Mindstorms NXT 2.0 Discovery Book: a Beginner's Guide to Building and Programming Robots,(2010) p. cm. *Includes index. ISBN-13*, 978-1.
- Wang, S. (2019). *Road Terrain Classification Technology for Autonomous Vehicle*. Berlin, Germany: Springer.
- Weiss, C., Frohlich, H., & Zell, A. (2006, October). Vibration-based terrain classification using support vector machines. In *2006 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4429-4434). IEEE.
- Wieringa, R. J. (2014). What is design science?. In *Design science methodology for information systems and software engineering* (pp. 3-11). Springer, Berlin, Heidelberg.



.....

Student

Supervisor

Date