

Filipe do Ó Cavalcanti

Real-time Pavement Classification Using an Embedded Artificial Neural Network

João Pessoa - Brazil

March, 2023

Filipe do Ó Cavalcanti

Real-time Pavement Classification Using an Embedded Artificial Neural Network

Dissertation presented to the Electrical Engineering Graduation Program of the Federal University of Paraíba, as requirement for obtaining the title of Master in Electrical Engineering.

Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Programa de Pós-Graduação em Engenharia Elétrica

Advisors: Fabrício Braga Soares de Carvalho, D.Sc.

Waslon Terllizzie Araújo Lopes, D.Sc.

João Pessoa - Brazil

March, 2023

Catálogo na publicação
Seção de Catalogação e Classificação

C376r Cavalcanti, Filipe do Ó.

Real-time pavement classification using an embedded artificial neural network / Filipe do Ó Cavalcanti. - João Pessoa, 2023.

67 f. : il.

Orientação: Fabrício Braga Soares de Carvalho, Waslon Terllizzie Araújo Lopes.

Dissertação (Mestrado) - UFPB/CEAR.

1. Engenharia elétrica - Sistemas embarcados. 2. Redes neurais artificiais. 3. Classificação de pavimento. 4. Acelerômetro. 5. RTOS. I. Carvalho, Fabrício Braga Soares de. II. Lopes, Waslon Terllizzie Araújo. III. Título.

UFPB/BC

CDU 621.3:681.51(043)

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação


**REAL-TIME PAVEMENT CLASSIFICATION USING AN EMBEDDED ARTIFICIAL
NEURAL NETWORK**

Elaborada por


FILIPPE DO Ó CAVALCANTI

como requisito parcial para obtenção do grau de
Mestre em Engenharia Elétrica.


COMISSÃO EXAMINADORA

Documento assinado digitalmente
 **FABRICIO BRAGA SOARES DE CARVALHO**
Data: 30/03/2023 16:26:05-0300
Verifique em <https://validar.iti.gov.br>


PROF. DR. FABRÍCIO BRAGA SOARES DE CARVALHO

Documento assinado digitalmente
 **WASLON TERLLIZZIE ARAÚJO LOPES**
Data: 30/03/2023 16:48:57-0300
Verifique em <https://validar.iti.gov.br>

PROF. DR. WASLON TERLLIZZIE ARAÚJO LOPES
Coorientador – UFPB

Documento assinado digitalmente
 **CLEONILSON PROTÁSIO DE SOUZA**
Data: 30/03/2023 17:27:18-0300
Verifique em <https://validar.iti.gov.br>

PROF. DR. CLEONILSON PROTÁSIO DE SOUZA
Examinador Interno – UFPB

Documento assinado digitalmente
 **Carlos Danilo Miranda Régis**
Data: 30/03/2023 16:32:06-0300
Verifique em <https://validar.iti.gov.br>

PROF. DR. CARLOS DANILO MIRANDA RÉGIS
Examinador Externo – IFPB

Acknowledgements

First, I would like to thank my wife Sara Mirthis, who is a brilliant engineer and assisted me with this research. She was very supportive during the entire time I was dedicated on it, and I am thankful.

I also want to thank my family, especially my mother Poliana, who never doubted my ideas and skills to make any of my projects work.

Professors Fabrício and Waslon, who are two great professionals I first met during my undergraduation, and now we worked together on this. They embraced this idea and never looked back. I am thankful for working with such professionals.

My work colleagues at Lumentum in Campinas, who gave me inspiration to develop everything using the best tools of the industry, and always striving for quality.

Lastly, I thank my father Homero who was my inspiration to become an electrical engineer with absolutely no caution in having out of the box ideas. He inspired me to take part in the academic side of the world. Curiously, he was a professor to both my advisors. I guess this gave them some confidence.

“If I wait for the genius to come, it just doesn’t arrive.”
(Ian Fleming)

Resumo

A análise da condição de pavimentos pode ser uma funcionalidade muito importante para automóveis e também para pesquisas sobre qualidade de pavimentação em geral. Tal análise fornece dados que podem ser utilizados desde referência para decidir se uma rua necessita de manutenção, até como uma maneira de selecionar uma rota diferente no mapa durante uma viagem, baseado em uma definição de conforto selecionada pelo motorista. Existem métodos para análise de pavimentação, como o *Pavement Condition Index* (PCI), que requer grande análise manual do pavimento. Neste contexto, este trabalho apresenta um sistema que utiliza técnicas de aprendizado de máquina para classificação automática do tipo de pavimentação baseando-se em amostras de um acelerômetro acoplado a um veículo. Isto permite classificar uma rua entre duas categorias de pavimento: asfalto ou paralelepípedo. Um acelerômetro triaxial e um módulo capaz de obter sinais do Sistema de Posicionamento Global (GPS) são utilizados como periféricos em um sistema operacional de tempo real (RTOS) capaz de executar aquisição de dados com precisão e também classificar um terreno em tempo real. Tal operação requer que o módulo acelerômetro esteja montado próximo ao centro de gravidade do veículo e calibrado conforme o ponto de apoio na montagem ao chassi. O modo de aquisição de dados foi testado na cidade de Campinas, SP - Brasil, cujos dados obtidos são a aceleração nos eixos longitudinal, lateral e vertical e dados de geolocalização a partir do módulo GPS. Estes dados são analisados e uma extensa seleção de atributos é realizada para escolher as melhores métricas que podem ser utilizadas para treinar uma Rede Neural Artificial capaz de classificar entre as duas categorias de pavimento. O modelo utiliza valores de entrada extraídos das informações de aceleração tanto no domínio do tempo quanto na frequência e atinge uma acurácia de 94% no conjunto de testes. O modelo treinado é inserido no sistema embarcado, permitindo a classificação do terreno em tempo real.

Palavras-chave: Sistemas Embarcados, Redes Neurais Artificiais, Classificação de Pavimento, Acelerômetro, RTOS.

Abstract

The capacity to analyze pavement condition can be a very important feature for either automobiles or general road quality surveys. This information can be used as reference to decide when a road requires maintenance or as a means to select a different route when travelling, based on a driver requirement for comfort. There are methods available for general road analysis such as the Pavement Condition Index (PCI) that requires extensive manual survey of the pavement. In this context, this work presents a system that uses machine learning techniques for automatic pavement classification, based on accelerometer readings that are used to classify roads between two categories: asphalt or paving stone. A triaxial accelerometer and Global Positioning System (GPS) modules are used as peripherals to a real-time operating system (RTOS) that can execute high precision data acquisitions and also classify the road in real time. This operation requires that an accelerometer module is mounted next to the vehicles center of gravity and is calibrated to the vehicle mounting point. The data acquisition mode is used to obtain data in the city of Campinas, SP - Brazil, containing acceleration from longitudinal, lateral and vertical axis and geolocation data from the GPS. Then, this data is analyzed and an extensive feature selection process is executed to filter the best metrics that can be used for training the Artificial Neural Network (ANN) for pavement classification between the two types of road. The model uses features extracted from acceleration data in both time and frequency domains and achieved an accuracy of 94% in the test set. The model was added to the embedded system, allowing classification of the pavement in real time.

Keywords: Embedded Systems, Artificial Neural Networks, Pavement Classification, Accelerometer, RTOS.

List of Figures

Figure 1 – A simple I2C data transfer (NXP SEMICONDUCTORS, 2021).	20
Figure 2 – Task states (adapted from FreeRTOS Website).	20
Figure 3 – Illustration of the co-operative scheduler.	21
Figure 4 – Illustration of the round-robin scheduler.	21
Figure 5 – Illustration of the preemptive scheduler.	22
Figure 6 – ROC curve example (COMMONS, 2021).	25
Figure 7 – Points on the ROC Curve. Adapted from (GöNEN, 2007).	25
Figure 8 – The nonlinear neuron model (HAYKIN, 1999).	27
Figure 9 – Common activation functions.	28
Figure 10 – A 4-5-2 Neural Network.	29
Figure 11 – Correlated data example: -0.7, 0.0 and 0.7 correlation.	31
Figure 12 – Overlap between two distributions.	32
Figure 13 – Accelerometer module GY521.	33
Figure 14 – Accelerometer axis (adapted from (INVENSENSE, 2013)).	33
Figure 15 – Accelerometer mounting plane relative to the ground plane.	35
Figure 16 – Dataflow on acquisition mode.	38
Figure 17 – Data flow on classification mode.	38
Figure 18 – Test bench.	40
Figure 19 – Vehicle assembly.	41
Figure 20 – Accelerometer axes relative to the ground.	41
Figure 21 – Data acquisition.	44
Figure 22 – Correlation matrix of metrics after filtering by correlation.	46
Figure 23 – Distribution plot of AccX-Mean and AccX-Peak.	46
Figure 24 – Distribution plot of AccX-RA5-Stddev and AccX-RMS.	47
Figure 25 – Distribution plot of AccX-RV-Peak and AccY-Mean.	47
Figure 26 – Distribution plot of AccY-Peak and AccY-RA5-Stddev.	47
Figure 27 – Distribution plot of AccY-RMS and AccY-RV-Peak.	48
Figure 28 – Distribution plot of AccZ-RV-Peak and Spd-Mean.	48
Figure 29 – Visualization of the selected features.	49
Figure 30 – Visualization of the selected features (continuation).	50
Figure 31 – Training accuracy for configuration one.	52
Figure 32 – Validation accuracy for configuration one.	53
Figure 33 – Loss values for training and validation (dotted curves) for configuration one.	53
Figure 34 – Training accuracy for configuration two.	54
Figure 35 – Validation accuracy for configuration two.	54

Figure 36 – Loss values for training in configuration two.	55
Figure 37 – Loss values for validation in configuration two.	55
Figure 38 – Comparing training accuracy of the two training configurations.	56
Figure 39 – Comparing validation accuracy of the two training configurations.	56
Figure 40 – Training and validation loss compared for the two configurations (dotted line represents validation).	57
Figure 41 – Selected model ROC curve.	58
Figure 42 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.	60
Figure 43 – Comparison of real-time and computer analyzed prediction output.	60
Figure 44 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.	61
Figure 45 – Comparison of real-time and computer analyzed prediction output.	61
Figure 46 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.	62
Figure 47 – Comparison of real-time and computer analyzed prediction output.	62

List of Tables

Table 1 – Register used to for power management on the MPU.	34
Table 2 – Register used to set the accelerometer full-scale range.	34
Table 3 – Accelerometer sensitivity for each full scale setting (INVENSENSE, 2013).	34
Table 4 – Registers that store the most recent accelerometer measurements.	35
Table 5 – Fields of interest in the GPS output message, adapted from (U-BLOX, 2013).	36
Table 6 – RTOS tasks.	37
Table 7 – Training and validation dataset composition.	41
Table 8 – Testing dataset composition.	42
Table 9 – Values representing terrain.	43
Table 10 – Normal distribution mean, standard deviation and overlapping coefficient between asphalt (1) and paving stone (2) samples.	49
Table 11 – MLP model parameters.	51
Table 12 – Summary of the best training results for both configurations.	56
Table 13 – Metrics for the test dataset using threshold 0.75.	58
Table 14 – Real-time prediction dataset.	59

List of abbreviations and acronyms

AI	Artificial Intelligence
ADC	Analog-to-digital Converter
ANN	Artificial Neural Network
API	Application Programming Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
DNIT	Brazilian National Department of Transportation Infrastructure
ESC	Electronic Stability Control
FFT	Fast Fourier Transform
FWT	Fast Wavelet Transform
GPS	Global Positioning System
GSL	GNU Scientific Library
I2C	Inter-Integrated Circuit
ISR	Interrupt Service Routine
IDE	Integrated Development Environment
MEMS	Micro-Electro-Mechanical Systems
MLP	Multi Layer Perceptron
OVL	Overlapping Coefficient
PCI	Pavement Condition Index
ROC	Receiver Operating Characteristics
RMS	Root Mean Square
RTOS	Real-time Operating System
SVM	Support Vector Machine
SUV	Sport Utility Vehicle
UART	Universal Asynchronous Receiver/Transmitter

Contents

1	INTRODUCTION	13
1.1	Context	13
1.2	Motivation	14
1.3	Main Objectives	15
1.3.1	Specific Objectives	15
2	LITERATURE REVIEW	17
2.1	Accelerometers	17
2.1.1	MEMS Accelerometers	18
2.2	Global Positioning System (GPS)	18
2.3	I2C Communication	19
2.4	Real-time Operating Systems	20
2.4.1	Co-operative Scheduling	21
2.4.2	Round-robin Scheduling	21
2.4.3	Preemptive Scheduling	21
2.4.4	FreeRTOS	22
2.5	Machine Learning	23
2.5.1	Supervised Learning	23
2.5.2	Unsupervised Learning	23
2.5.3	Binary Classification Metrics	23
2.5.4	ROC Curve	24
2.6	Artificial Neural Networks	25
2.6.1	Neuron Model	26
2.6.2	Activation Function	27
2.6.3	Layers and Network Architectures	28
2.6.3.1	Single-Layer Feedforward	29
2.6.3.2	Multilayer Feedforward	29
2.6.3.3	Recurrent	29
2.7	Feature Selection	29
2.7.1	Correlation-based Feature Selection	30
2.7.2	Overlapping Coefficient	30
3	METHODOLOGY	33
3.1	Embedded System Actions	33
3.1.1	Accelerometer	33
3.1.1.1	Accelerometer Calibration	35

3.1.2	GPS Module	36
3.1.3	RTOS	37
3.1.4	Data Acquisition Mode	37
3.1.5	Classification Mode	38
3.1.6	Libraries and Frameworks	39
3.2	Experimental Setup	39
3.3	Data Acquisition	39
3.3.1	Training and Testing Datasets	40
4	RESULTS	43
4.1	Preprocessing	43
4.2	Feature Extraction	44
4.3	Feature Selection	45
4.4	Training the Artificial Neural Network	50
4.4.1	Data Scaling and Splitting	50
4.4.2	Training Strategy	51
4.4.3	Selecting Best MLP Configuration	52
4.4.4	Test Dataset Results	57
4.5	Validating Real-Time Prediction	58
4.5.1	Comparing Results	59
5	CONCLUSIONS	63
5.1	Future Work	64
5.2	Publications	64
5.3	Software Registration	64
	BIBLIOGRAPHY	65

1 Introduction

1.1 Context

The Brazilian road network is the main form of transportation of goods across the country. The road fleet has grown 7.5% per year from 2000 to 2016, which is an increase from 29.7 to 93.9 million licensed vehicles. All those vehicles move in approximately 1.8 million kilometers of roads of which 79% is not paved, 12% is paved and 9% is planned, as of 2015 ([Departamento da Indústria da Construção, 2017](#)).

Looking at the paved roads, the National Department for Infrastructure and Transportation (DNIT) shows that 23.6% of those roads are in bad or very bad conservation conditions and only 34.7% are considered regular ([Departamento da Indústria da Construção, 2017](#)).

These data correspond to the road network where trucks and buses are present. In most Brazilian cities, it is common to have streets paved with paving stones in residential areas due to its lower installation and maintenance costs ([MEGIER et al., 2018](#)). Paving stones are irregular blocks of stone grouted with mortar of cement and sand, commonly found in locations of historical preservation, residential areas, industrial driveways and private roads. This type of pavement slows down traffic, provides less comfort and can deteriorate vehicles faster when compared to asphalt.

The subjective understanding of comfort from the point of view of drivers and passenger in a vehicle comes from the vibrations. When driving in an asphalt road, the vehicle experiences little vibration due to a smoother surface. In a stone paved road this vibration is higher due to the irregular surface of the stone. Driving in excess speeds on this surface can cause damages to vehicles and certainly discomfort to drivers and passengers.

During the development phase of vehicles such as Sport Utility Vehicles (SUVs), sedans and hatchbacks, there is an effort in analyzing vibrations as a parameter to tune engine and transmission response for the best performance and comfort possible ([WANG; CAO; QU, 2019](#)). This form of analysis requires the use of advanced software and accelerometers in smooth test tracks, so that most vibrations are originated from the vehicle itself and not by irregularities on the road surface.

In this context, there are studies that take a similar approach to ride comfort analysis. However, those studies aim at analyzing road quality instead of vehicle quality while using very similar data collection techniques and statistical analysis.

1.2 Motivation

The work in [Weiss, Frohlich e Zell \(2006\)](#) uses a Support Vector Machine (SVM) to classify the road in seven types of pavement, using only an accelerometer on the vertical axis that is coupled to a small transport cart limited to a speed of 1 m/s. The work compares statistical metrics of the temporal data to metrics obtained from the frequency domain, such as Power Spectral Density (PSD) and the Fast Fourier Transform (FFT). The authors conclude that the model trained for seven types of pavement leads to some improvements when using metrics from both frequency and time domains. However, for only three types there is no advantage, and it is better to use temporal metrics only.

Some papers try to detect the occurrence of potholes or general anomalies using an Artificial Intelligence (AI) implementation for a more detailed classification of the terrain. The approach in [Mednis et al. \(2011\)](#) is focused on detecting potholes and compares some techniques used by other authors. The Z-THRESH, Z-DIFF and STDDEV algorithms were compared to the proposed G-Zero algorithm. Z-THRESH will detect an anomaly when the vertical axis acceleration goes above a certain threshold. Z-DIFF will seek two consecutive values with the greatest difference in amplitude, showing when a sudden change in acceleration occurs. STDEV evaluates the standard deviation in a set of data, and it will classify a pothole if it crosses a threshold. In addition to the three methods mentioned above, the proposed G-Zero algorithm analyzes instants where the three accelerometer values approach zero g , which can be an indication of a free fall. In the analysis, it was stated that Z-DIFF provides the best performance when comparing the rate of true positives.

Similar to the approach in [Mednis et al. \(2011\)](#), the work on [Carlos et al. \(2018\)](#) seeks to identify anomalies on a surface. The data were obtained from accelerometers available in smartphones. The author uses SVM and compares it with the techniques proposed by other authors. The comparison occurs in 30 different streets, and it is concluded that the proposed SVM has better F1 score although the STDEV technique proposed in [Mednis et al. \(2011\)](#) shows better result among the analyzed works.

A vertical axis accelerometer was used on a truck and compared an Artificial Neural Network (ANN), SVM and Naive Bayes for terrain classification, while searching for a speed independent classification model. This work also used a vehicle model for simulation ([WANG; KHUSHABA; KODAGODA, 2012](#)). On different terrains, data acquisitions at speeds of 20, 30 and 40 km/h were performed and two main metrics were compared: FFT and Fast Wavelet Transform (FWT). The work suggests that if a classification independent of speed is desired, the metrics generated from the FWT will yield a better output.

In addition to the use of triaxial accelerometer, the work on [Meocci, Branzi e Sangiovanni \(2021\)](#) also has a triaxial gyroscope and geolocation by the use of the Global

Positioning System (GPS). The proposed technique detects anomalies on the ground up to 2 cm long but makes an analysis based on length of asphalt, where every 100 m provides data for a classification. The proposed classification method is based on the Pavement Condition Index (PCI), an index heavily dependent on manual survey of the pavement condition where a grade is assigned to a length of asphalt that indicates its quality. By using this equipment, the work creates an equivalent scale to the PCI proposal. This approach provides different metrics from the ones proposed in other works, such as the peak of the moving variance.

In the previous work in [Cavalcanti, Lopes e Carvalho \(2021\)](#), a SVM classifier was developed to classify between two types of pavement: asphalt and paving stone. The features contain metrics in time and frequency domain for classification with precision up to 97%. In [Cavalcanti, Lopes e Carvalho \(2022\)](#), preliminary results of the approach explained in this work are shown, using a simplified Artificial Neural Network (ANN) to validate the concept of real time terrain classification.

1.3 Main Objectives

This work proposes a system for data acquisition and real-time classification of pavements. Acceleration data from a vehicle must be obtained from a triaxial accelerometer mounted on the vehicle chassis and a GPS is used to record position and speed data. Using data from those peripherals, an ANN is trained to classify pavements in two categories: asphalt or paving stone.

1.3.1 Specific Objectives

The main goal is to train an ANN capable of real-time classification using the limited processing power of the embedded system used. This requires that a data acquisition, data processing, training environment and real-time data manipulation procedures be implemented.

The implementation can be separated into the following steps:

- Preparation of a firmware development environment;
- Implementation of the necessary firmware for configuration and data acquisition of GPS and accelerometer modules;
- Development of a stable RTOS environment for data acquisition and data offloading;
- Determination of a standard output format for data acquisitions;

- Implementation of the process for real-time classification using an embedded ANN model;
- Development of a software in Python capable of processing all data acquisition files;
- Use the concepts of feature engineering to study possible features to be extracted from the data;
- Selection of the best features to train the ANN based on correlational analysis;
- Training the ANN;
- Adaptation of the ANN model to the embedded system;
- Validation of the real-time classification accuracy.

This work is organized as follows: Chapter 2 explains the main concepts required for this work, starting with accelerometer theory, GPS and then electronics subjects, such as I2C communication and RTOS. An introduction to machine learning theory is presented, followed by a more detailed explanation of artificial neural networks and feature selection. Chapter 3 describes the three main areas of development of this research: the first is embedded system actions, that describes the RTOS and its interaction with accelerometer and GPS for data acquisition. The experimental setup is demonstrated as well. Finally, the data acquisition process dataset separation into training and testing is detailed. Chapter 4 describes the data processing required to clean and filter the datasets in a process called feature extraction and selection. Then, the training strategy for two ANN configurations is detailed, followed by the training results and selection of the best architecture. In the last section, the trained model is embedded on the firmware and a new data acquisition is executed with real-time prediction data, that is later validated on a computer and its accuracy is analyzed. Lastly, Chapter 5 presents the conclusion around the results obtained.

2 Literature Review

This chapter introduces the key concepts used in the development of this work. Section 2.1 introduces the theory behind accelerometer construction and applications. Section 2.2 describes how the GPS constellation works. In Section 2.3 the I2C communication protocol is explained. Section 2.4 introduces the key concepts of real-time operating systems. Finally, Sections 2.5, 2.6 and 2.7 describe the key concepts of Machine Learning and Neural Network.

2.1 Accelerometers

Accelerometers are devices that measure proper acceleration. On planet Earth, a triaxial accelerometer at rest, parallel to the ground and with its vertical axis pointing upwards, will measure -9.8 m/s^2 due to gravitational pull. The unit g is commonly used to represent acceleration where 1 g equals to 9.807 m/s^2 .

These devices have applications in many areas of science and consumer products. In industrial scenarios, accelerometers are used to measure vibration in motors and pumps, where the analysis of acceleration patterns can reflect the state of life of the equipment. On biological applications, accelerometers can be used as microphones by acting just like a human eardrum, which basically responds to oscillations in pressure (MATIC; OSMANI; MAYORA, 2012) and can be used to identify speech. General consumer products are also equipped with accelerometers. Modern smartphones have the capability to change its screen rotation based on accelerometer readings. Videogames that require motion input also use accelerometers to analyze movements and translate these data to a game input.

Vehicular applications also use accelerometers in many ways. For instance, crash detection requires a certain acceleration threshold to deploy the airbags and prevent injuries to driver and passengers on an accident (GERHARD, 2013). Electronic Stability Control (ESC) is a vehicular application that can brake the wheels of a car to correct its direction if a spin is pre-detected, such as in the case of aquaplaning or losing grip on black ice, all based on accelerometer data. In the automotive industry, accelerometers can be used as the main reference to analyze vehicle vibration when developing a new vehicle. Combining accelerometers and general vehicle data such as throttle, speed, gear, and engine speed, engineers can analyze the quality of common vehicle operation. Gear shifts, idle vibration, acceleration, start up vibration and many others maneuvers result in vibrations that can be felt by consumers (WANG; CAO; QU, 2019). Comparing data from multiple vehicles, engineers can evaluate which approach is best when designing new products.

2.1.1 MEMS Accelerometers

Accelerometers are the most common and widely used sensor implemented using Micro-Electro-Mechanical Systems (MEMS) technology. In general, MEMS devices are used for motion, pressure, light and ultrasound sensing. There are also implementations of microswitches and microactuators for control applications ([RASRAS; ELFADEL; DUONG, 2019](#)). Those devices can range from 20 micrometers to 1 millimeter in size, usually consisting of a microprocessor for data processing and components that interact with the surroundings, such as the microsensors ([QUADRI, 2020](#)).

A traditional MEMS accelerometers employ a proof mass that is suspended using springs, which displaces in response to an external acceleration. A proof mass can be used for one or multiple axis sensing. There are a variety of transduction mechanisms used for detecting displacement, such as capacitive, piezoelectric, piezoresistive, thermal, tunneling and optical ([RASRAS; ELFADEL; DUONG, 2019](#)).

2.2 Global Positioning System (GPS)

The GPS was originally developed in the United States in 1973 under the name NAVSTAR. Instead of traditionally using angular measurements from natural stars for navigation, the proposal was to use ranging measurements of artificial satellites. This system revolutionized navigation as we know ([PARKINSON et al., 1996](#)). It took 20 years for the system to be fully operational, and today it is available in almost every smartphone, twenty-four hours a day.

The GPS can be divided into three segments ([U.S. DEPARTMENT OF DEFENSE, 2008](#)):

- Space Segment: a constellation of 27 satellites that continuously transmit navigational messages (position and time);
- Control Segment: a ground control that tracks the constellation and updates each satellite with future positions and clock correction;
- User Segment: the receiver on the final user that obtains GPS signal and calculates the three-dimensional position and local time.

Among its main objectives, the GPS must provide high accuracy, real-time position, velocity and time for military users. For civilian use, it must provide good position accuracy (100 m) ([PARKINSON et al., 1996](#)). Today, accuracy for civilian use is much higher and this is controlled by a deliberate degradation of the range signal, which is a tool available to control who has signal access in times of war.

The GPS constellation was originally made of 24 satellites which is the necessary number for complete Earth coverage and since 2011, the constellation was expanded to 27 satellites. For a user to have access to three-dimensional navigation (horizontal position and elevation) and time accuracy, it is necessary at least four satellites in range ([PARKINSON et al., 1996](#)).

2.3 I2C Communication

The Inter-Integrated Circuit (I2C) is a standard bidirectional interface that uses a controller known as the master, to communicate with slave devices ([VALDEZ; BECKER, 2015](#)). The I2C bus is a world standard implemented in over 1000 different Integrated Circuits (ICs) manufactured by more than 50 companies. It is also used in various control architectures such as System Management Bus (SMBus), Power Management Bus (PMBus), Intelligent Platform Management Interface (IPMI), Display Data Channel (DDC) and Advanced Telecom Computing Architecture (ATCA) ([NXP SEMICONDUCTORS, 2021](#)).

This communication bus can achieve multiple speeds depending on the application. Data transmission is serial, bidirectional, 8-bit oriented and the transfer speeds can go up to 100 kbit/s on Standard-mode, 400 kbit/s on Fast-mode, 1 Mbit/s on Fast-mode plus or 3.4 Mbit/s on High-speed mode. If unidirectional data transfer is used, it can achieve up to 5 Mbit/s on Ultra Fast-mode ([NXP SEMICONDUCTORS, 2021](#)).

I2C has only two electrical information lines: serial-clock line (SCL) and serial data (SDA), both connected to a fixed voltage through a pull-up resistor. Each of those lines are open-drain/open-collect with an input buffer on the same line, which means the device is only capable of pulling the bus low. When idle, the bus will stay high ([VALDEZ; BECKER, 2015](#)). This simple implementation allows that prototyping can be as simple as connecting a device to both bus lines, while the only limitation being the line capacitance.

The slave device only transmits data if requested by a master. Each slave has a specific address to differentiate from other devices and may contain one or multiple registers for data storage, written or read ([VALDEZ; BECKER, 2015](#)). Every data written on the SDA line must be eight bits long and there are no limits on bytes transmitted per transfer. The data transfer starts on the Most Significant Bit (MSB). Figure 1 shows the start condition for a transfer, the Acknowledgment (ACK) signals and the stop condition to end a data transfer.

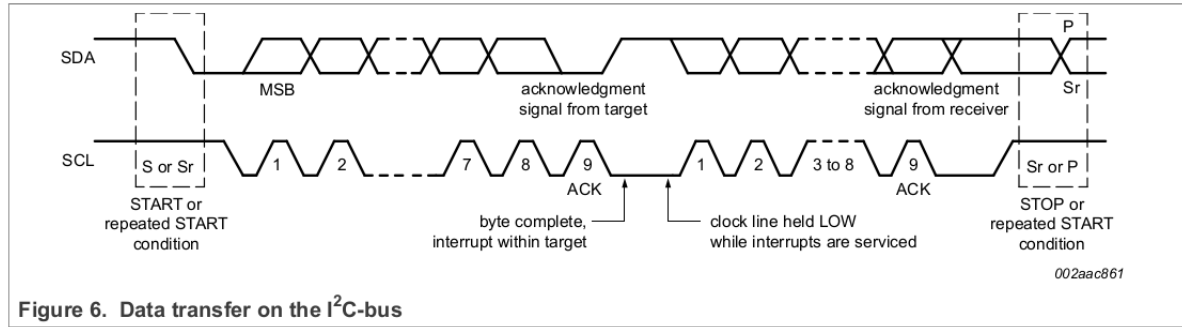


Figure 1 – A simple I2C data transfer (NXP SEMICONDUCTORS, 2021).

2.4 Real-time Operating Systems

Real-time operating systems (RTOS) are systems developed for applications that require very specific timings. It is based on the concept of multitasking where several tasks (or programs) are executed concurrently on the same Central Processing Unit (CPU) but not at the same time. The RTOS itself is a program that manages system resources, schedules and synchronizes tasks, manages resource allocation and provides inter-task communication (IBRAHIM, 2020).

A task is an independent thread of execution, usually with its own local set of data (IBRAHIM, 2020). These data can be transmitted between tasks using inter-task communication methods such as signals, queues and event flags. Each task can have a priority that is static (not changed throughout the execution) or dynamic (can be changed during runtime). A task can also have multiple states as shown in Figure 2.

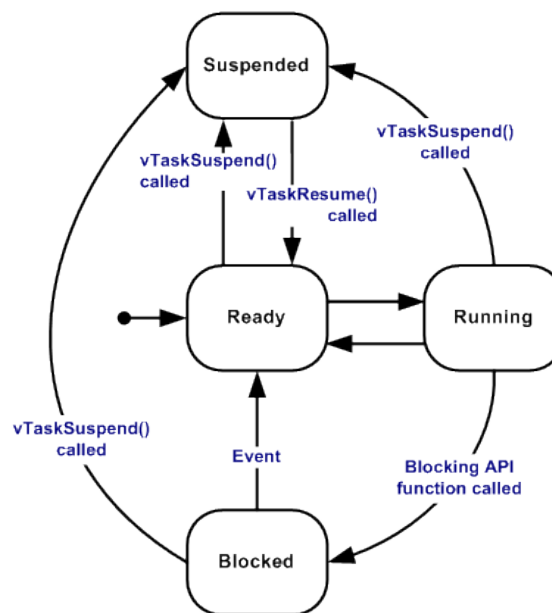


Figure 2 – Task states (adapted from FreeRTOS Website).

Real-time systems usually operate based on priorities, where tasks with higher priority can take complete use of the CPU until its execution releasing processing power to a lower priority task. This depends on the type of scheduler selected.

A scheduler switches from one task to another by performing a context switch. The context switch stores the current state of execution of a thread and can continue execution later without any data loss. There multiple schedulers available, where the most common are: co-operative, round-robin and preemptive (IBRAHIM, 2020).

2.4.1 Co-operative Scheduling

The co-operative scheduling is one of the simplest forms of implementing a scheduler, usually when there is no time critical tasks. Each task will use the necessary CPU time for its operation and release it to the next task available. One task will never interrupt the execution of another task. The implementation is as simple as a while loop that calls three functions one after the other. Figure 3 illustrates the co-operative scheduling of three tasks.

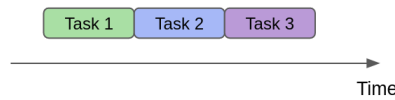


Figure 3 – Illustration of the co-operative scheduler.

2.4.2 Round-robin Scheduling

The Round-robin scheduler puts the tasks in a circular queue and allocates each task an equal amount of CPU time. When the task execution time expires, it is put on the end of queue and its current context is saved in memory, so it can continue execution from the point it stopped when it is given CPU time again.

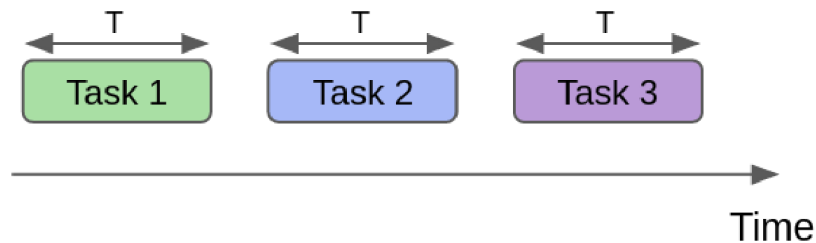


Figure 4 – Illustration of the round-robin scheduler.

2.4.3 Preemptive Scheduling

This is the most common scheduler used in real-time applications (IBRAHIM, 2020). Each task is executed based on their priority, where the highest priority task will

get all CPU time while necessary to execute its function. If a task with higher priority is ready to be executed, the current task will be stopped, its current state stored, and a context switch occurs to start the higher priority task. When this task is done, the lower priority task can resume operations.

If two tasks are assigned the same priority level, they will operate under a Round-robin scheduler, as illustrated in Figure 5. The Preemptive scheduler is more complex to operate than the Co-operative and Round-robin, requiring well planned priorities to avoid never releasing tasks.

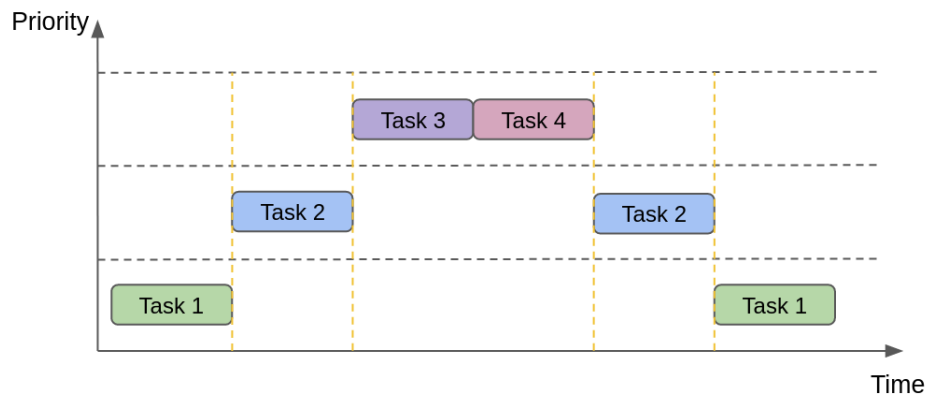


Figure 5 – Illustration of the preemptive scheduler.

2.4.4 FreeRTOS

FreeRTOS is a real-time kernel for embedded microcontroller-based multi-task applications where there are more than one task sharing CPU time. It is a high quality software with very strict quality control that is freely available even for commercial applications. It was originally developed by Richard Barry in 2003 and maintained by his own company Real Time Engineers (IBRAHIM, 2020). In 2017, it was sold to Amazon Web Services (AWS) and it is still the most used real-time kernel.

The FreeRTOS kernel is a set of C source files that must be built with the application code of the project. The main configuration file of the RTOS is the FreeRTOSConfig.h header file, where configurations such as tick rate (a periodic timer used by the scheduler to check task states), system clock, type of scheduler, and types of memory allocation are defined.

This kernel was selected to be used in the current application due to wide availability of community support, forums and general stability.

2.5 Machine Learning

Learning can be described as the process of converting experience into expertise or knowledge. Machine Learning is an interdisciplinary field, sharing common knowledge with statistics, information theory, game theory, optimization and it is also a subfield of computer science and artificial intelligence. It seeks a mathematical understanding of the learning concept by supplying a learning algorithm with training data as input (representing experience) and the output is some expertise, usually taking the form of another computer program that will perform some task (SHALEV-SHWARTZ; BEN-DAVID, 2014).

The learning process is usually divided in supervised and unsupervised learning (SHALEV-SHWARTZ; BEN-DAVID, 2014), where the main difference between those is the interaction between the learner and the environment. The environment can be seen as a “teacher” that supplies the labels (if any) and guides the learning process.

2.5.1 Supervised Learning

In supervised learning, the knowledge is represented by a set of labeled data, and is separated from the unlabeled test data. Iteratively, the learning model adjusts its parameters based on the labeled data in a training process. The expertise after training is used in an unlabeled test data to predict the missing label information (HAYKIN, 1999).

2.5.2 Unsupervised Learning

In unsupervised learning there is no distinction between train and test data, no teacher and there are no labels to assist the learning process. The learning model process the input data and tries to come up with some summary, or compressed version of that data. Clustering a dataset into subsets of similar objects is a typical approach (SHALEV-SHWARTZ; BEN-DAVID, 2014). Once the network has become tuned to statistical regularities of the input data, it is able to form internal representations for encoding features of the input and thereby to create new classes automatically (BECKER, 1991).

2.5.3 Binary Classification Metrics

There are four basic metrics used to evaluate the predicted and true value of a classification (CLASSIFICATION..., 2022):

- **True Positive (TP)**: the model correctly predicts a positive class.
- **True Negative (TN)**: the model correctly predicts a negative class.
- **False Positive (FP)**: the model incorrectly predicts a positive class.

- **False Negative (FN):** the model incorrectly predicts a negative class.

Based on those metrics, that are some common performance metrics that give a general overview of a model performance.

- **Recall** (or sensitivy): the fraction of actual positives identified correctly.

$$Recall = \frac{TP}{TP + FN} \quad (2.1)$$

- **Precision:** fraction of correct predictions among the positive labels predicted.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

- **Accuracy:** fraction of predictions that the model correctly predicted. Equation 2.3 shows the case for binary classification, however it can be used in multi-class classification as the total number of correct predictions over the total predictions.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (2.3)$$

- **F1-Score:** the harmonic mean of precision and recall.

$$F1 = \frac{2TP}{2TP + FP + FN} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (2.4)$$

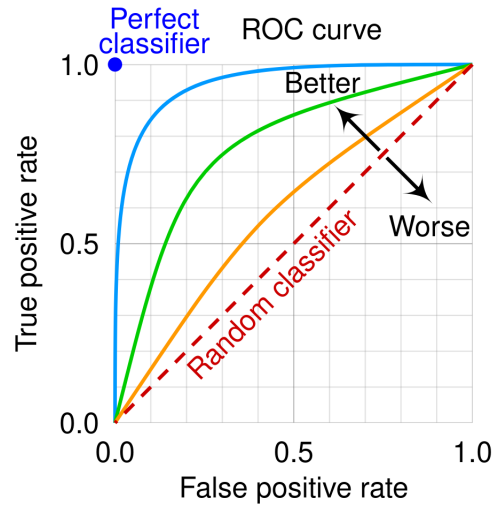
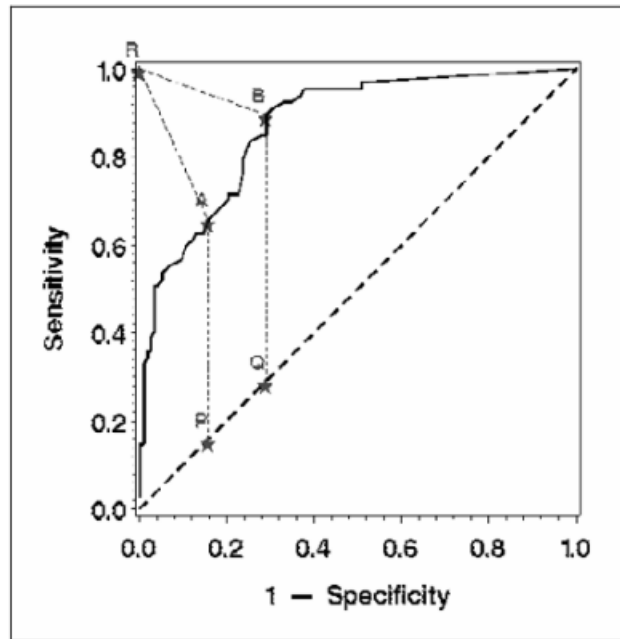
2.5.4 ROC Curve

The receiver operating characteristics (ROC) graph is a technique used to visualize and select classifiers based on their performance (FAWCETT, 2006). The ROC graph in Figure 6 is a two-dimensional graph where the false positive rate (FPR) is plotted on the X axis and the true positive rate (TPR) is plotted on the Y axis. It can be seen that the perfect classifier has a TPR of 1 and a FPR of 0. The closer a point on the ROC curve is to the top left, the better the threshold perform.

The ROC curve can be used to select the optimal threshold for a binary classifier (GÖNEN, 2007). There are two methods to select the optimal threshold: the first one makes the resulting binary prediction as close to a perfect predictor as possible. The second one makes the resulting binary prediction the most informative possible. This analysis requires calculating the Euclidian distance between points with coordinates $A(x_1, y_1)$ and $B(x_2, y_2)$, where:

$$d_{AB} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2.5)$$

The first method calculates the distance from the point (0,1) to the points A and B on the curve, as shown in Figure 7. The shortest distance corresponds to the best threshold. On the second method, the distance to the random classifier line is calculated (points P and Q), and the optimal threshold corresponds to the longest line.

Figure 6 – ROC curve example ([COMMONS, 2021](#)).Figure 7 – Points on the ROC Curve. Adapted from ([GöNEN, 2007](#)).

2.6 Artificial Neural Networks

Artificial Neural Networks (ANN) are a means of doing machine learning ([HARDESTY, 2017](#)). They are essentially a machine designed to model the way a brain executes a particular task or function of interest ([HAYKIN, 1999](#)).

The study of ANN has been inspired in part by the observation of biological systems, that are built of complex interconnections of neurons. ANN are built of a densely interconnected set of units, where each unit takes a number of real-valued inputs or event outputs from other units, producing a single real-valued output value ([MITCHELL, 1997](#)).

A Neural Network can be seen as an adaptive machine ([HAYKIN, 1999](#)). It is a massively parallel distributed processor, made up of simple processing units with the capacity to store experiential knowledge and making it available for use. It has similarities to the human brain in two respects: knowledge is obtained by the network from its environment through a learning process and interneuron connection strengths (or weights) are used to store the acquired knowledge.

Some benefits of neural networks are ([HAYKIN, 1999](#)):

- **Nonlinearity:** a neuron can be linear or nonlinear. This is important when the underlying physical mechanism that generates the input signal is inherently nonlinear (such as speech).
- **Input-Output Mapping:** modification of the synaptics weights are possible when using supervised learning, which modifies the network based on the training with labeled data.
- **Adaptivity:** allows the network to adapt the weights according to changes in the surrounding environment. Essentially, it can be easily retrained to deal with minor changes in its operating environment.
- **Evidential Response:** when classifying patterns, the neural network may not only provide information on which pattern to select, but also the confidence in the choice. It can be useful to filter ambiguous output.

The main neural network building blocks are described on the following sections.

2.6.1 Neuron Model

A neuron or node is basically a processing unit that is the building block of a neural network. The neuron model is composed of three main elements ([HAYKIN, 1999](#)):

- A set of synapses or links, each containing a weight. A signal that travels from the input of a link to a neuron is multiplied by said weight.
- An adder (linear combiner) that sums all input signals to the neuron and takes the respective synapses weight in consideration.
- An activation function that limits the amplitude of the neuron output value. The neuron output amplitude is usually in the closed interval of $[0,1]$ or $[-1,1]$.

The neuron model is shown in Figure 8. A neuron can be mathematically described using the following pair of equations (HAYKIN, 1999):

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (2.6)$$

and

$$y_k = \varphi(u_k + b_k), \quad (2.7)$$

where x_1, x_2, \dots, x_m are the input signals to the neuron, $w_{k1}, w_{k2}, \dots, w_{km}$ are the synaptic weights of the neuron; u_k is the linear combiner output; b_k is the bias; $\varphi(\cdot)$ is the activation function and y_k is the output signal of the neuron.

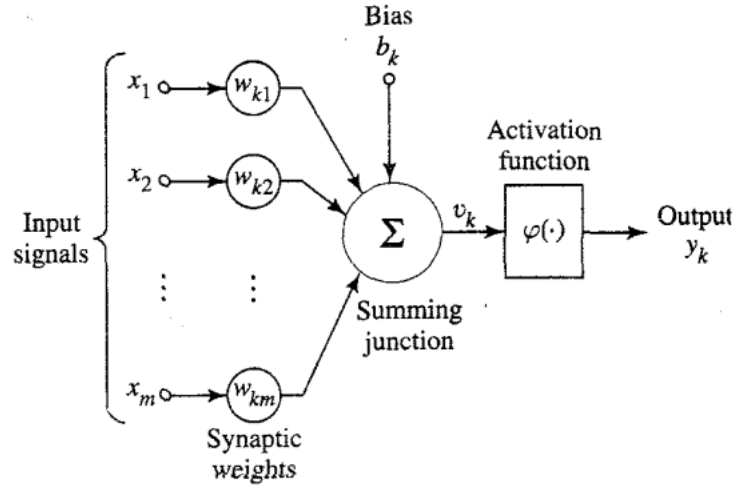


Figure 8 – The nonlinear neuron model (HAYKIN, 1999).

2.6.2 Activation Function

The activation function in a neuron defines the output of that node given an input or set of inputs (FUMO, 2017). There are three basic types of activation functions:

- Threshold Function (Heaviside Function): outputs zero if the input is negative and one otherwise. The Heaviside Function is given by

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad (2.8)$$

and its behavior is shown in Figure 9.

- Piecewise-Linear Function: its output value is in a linear, positive slope that ranges in the closed interval of $[-0.5, 0.5]$. If the input value is too large, it can be viewed

as a Threshold Function. This activation function is given by Equation 2.9 and can be observed in Figure 9.

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ v, & \frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \quad (2.9)$$

- Sigmoid Function: the most common form of activation functions used in ANN. It is an increasing function that shows a balance between linear and nonlinear behavior. At the center, the slope can be changed by adjusting the parameter a in Equation 2.10. In the limit, the slope parameter approaches infinity and it becomes a Threshold Function, as illustrated in Figure 9.

$$\varphi(v) = \frac{1}{1 + e^{-av}} \quad (2.10)$$

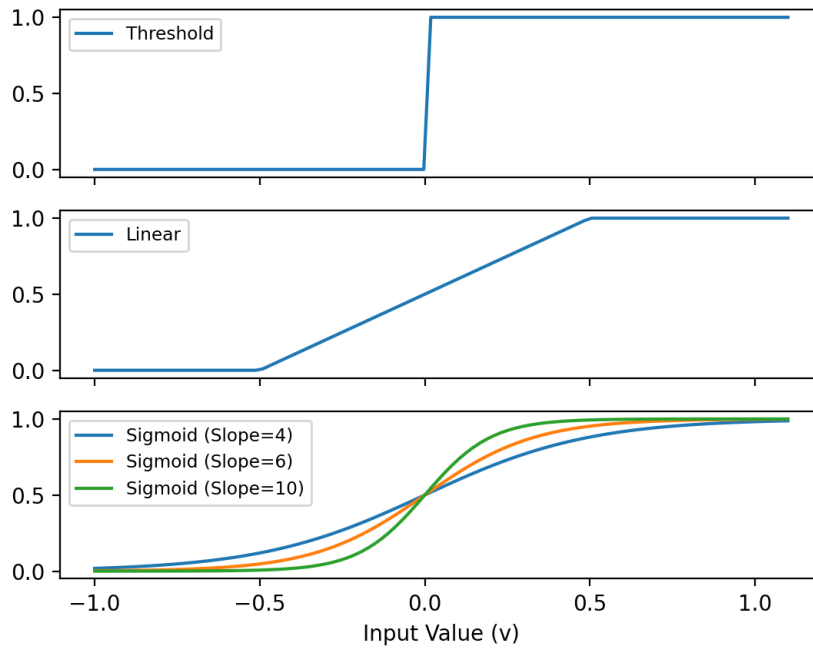


Figure 9 – Common activation functions.

2.6.3 Layers and Network Architectures

In a neural network, layers are essentially a group of nodes at a defined depth of the network. There is an input layer that is the entry point for the data, an output layer as the last layer that provides the results and between those two layers, there may be hidden layers which are groups of neurons to process the data (HAYKIN, 1999).

2.6.3.1 Single-Layer Feedforward

The Single-Layer Feedforward network has only one input layer that projects onto an output layer of neurons. It may have any number neurons on the output layer, but it is strictly feedforward: data only moves in one direction, hence the name feedforward.

2.6.3.2 Multilayer Feedforward

The Multilayer Feedforward network contains one or more hidden layers, allowing extraction of higher-order statistics. The input layer provides inputs to the first hidden layer, which outputs to the input of the second hidden layer and so on, until the output layer is reached.

When all nodes in a layer are connected to all nodes in the next layer, the network is called fully connected, otherwise it is called partially connected. Figure 10 shows a fully connected multilayer feedforward neural network. This network can be called 4-5-2, since it has 4 input nodes, 5 nodes on the first hidden layer and 2 output nodes (HAYKIN, 1999).

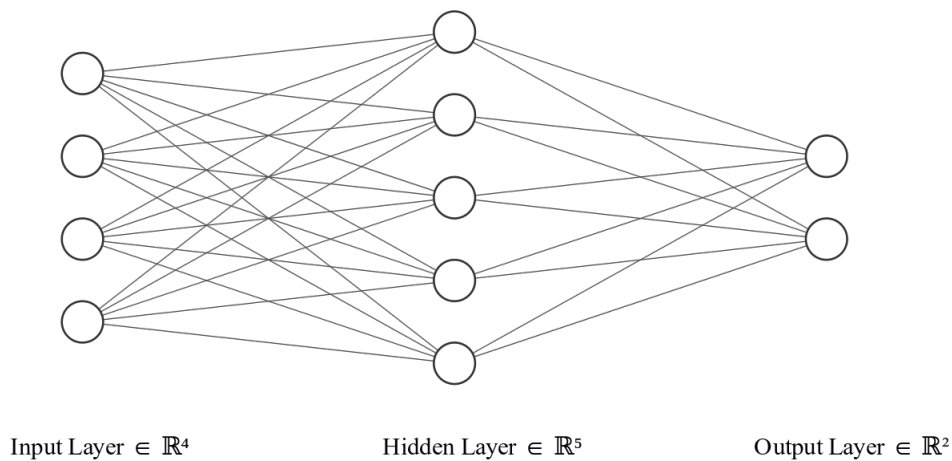


Figure 10 – A 4-5-2 Neural Network.

2.6.3.3 Recurrent

The recurrent network architecture has at least one feedback loop, where a neuron can feed its output signal back to all neuron inputs. The presence of feedback loops has a profound impact on learning and performance of the network. The feedback loop requires the use of branches composed of unit-delay elements.

2.7 Feature Selection

Feature selection is a process of choosing features to optimally reduce the feature space according to a certain criterion. It is effective in removing irrelevant and redundant

features and improve the learning model performance and enhance comprehensibility of the results (YU; LIU, 2003).

A feature is said to be “good” if relevant to the class concept but not redundant to any of the other features. So, a feature is good when highly correlated to the class but not to other features (YU; LIU, 2003).

There are three main methods for feature selection (GUYON; ELISSEEFF, 2003):

- Wrappers: uses the learning model as a black box to analyze and score subsets of variables according to their predictive power;
- Embedded: the selection of features happens during the training process;
- Filters: subsets of variables are selected as a pre-processing step. It can be less computationally intensive.

2.7.1 Correlation-based Feature Selection

Using correlation as a feature selection approach is a common filter technique. The Pearson Correlation is a measure of linear correlation between two datasets (GLEN, 2021). The correlation value ranges in the closed interval of $[-1, 1]$, where -1 represents a strong negative correlation, +1 represents a strong positive correlation and zero represents no correlation. The Pearson correlation parameter is defined as

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \quad (2.11)$$

where n is n-pairs of data $\{(x_1, y_1), \dots, (x_n, y_n)\}$. It is further simplified by the covariance of two variables x and y divided by the product of their standard deviations (SPSS..., 2022):

$$r = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)} \sqrt{\text{var}(y)}}. \quad (2.12)$$

When X and Y datasets are highly correlated to each another, r is or 1 or -1. Otherwise, r is zero. Figure 11 shows an example of an arbitrary dataset that contains positive, negative and no correlation.

2.7.2 Overlapping Coefficient

The Overlapping Coefficient (OVL) is a measure of agreement between two probability distributions or two populations represented by such distributions. It essentially gives a value between 0 and 1 representing the overlapping area of two distributions.

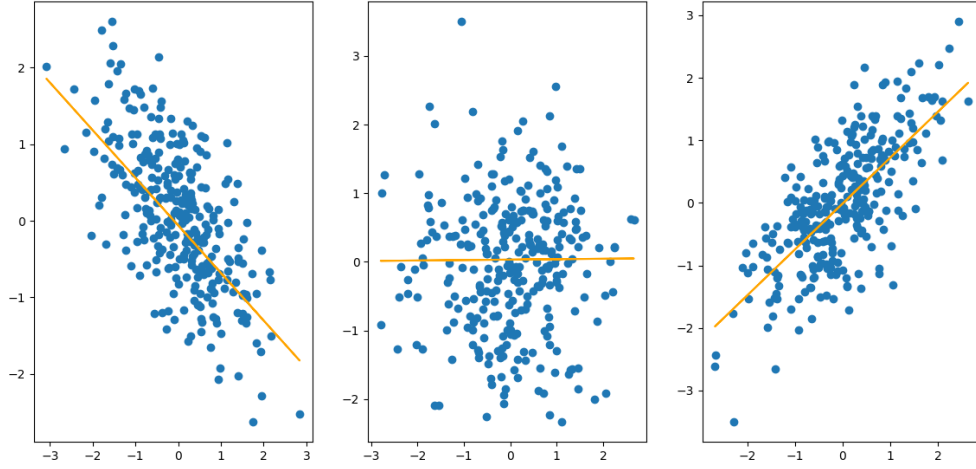


Figure 11 – Correlated data example: -0.7, 0.0 and 0.7 correlation.

Let $f_1(\underline{X})$ and $f_2(\underline{X})$ be probability density functions defined in R_n , the continuous case OVL is formally defined as (INMAN; JR, 1989)

$$OVL = \int_{R_n} \min[f_1(\underline{X}), f_2(\underline{X})] dX \quad (2.13)$$

and the discrete case OVL is defined as

$$OVL = \sum_{\underline{X}} \min[f_1(\underline{X}), f_2(\underline{X})]. \quad (2.14)$$

The OVL provides a common approach for the measure of the similarity of these distributions in any distributional setting and is based on a simple, easily comprehended concept of the agreement or similarity of probability distributions (INMAN; JR, 1989).

When two normal distributions have unequal variances, they will cross at two points. The OVL can be computed by first finding the intersection points using Equation 2.15 and computing the coefficient with Equation 2.16.

$$X = \frac{\mu_1\sigma_2^2 - \mu_2\sigma_1^2 \pm \sigma_1\sigma_2[(\mu_1 - \mu_2)^2 + (\sigma_2^2 - \sigma_1^2)\log_e\left(\frac{\sigma_2^2}{\sigma_1^2}\right)]}{\sigma_2^2 - \sigma_1^2} \quad (2.15)$$

$$OVL = 1 - |\Phi(x_2) - F_2(x_2)| - |\Phi(x_1) - F_2(x_1)| \quad (2.16)$$

Figure 12 shows a green area representing an OVL of 0.60993 between the distribution in blue ($\mu=0, \sigma=0.1$) and orange ($\mu=1, \sigma=2$).

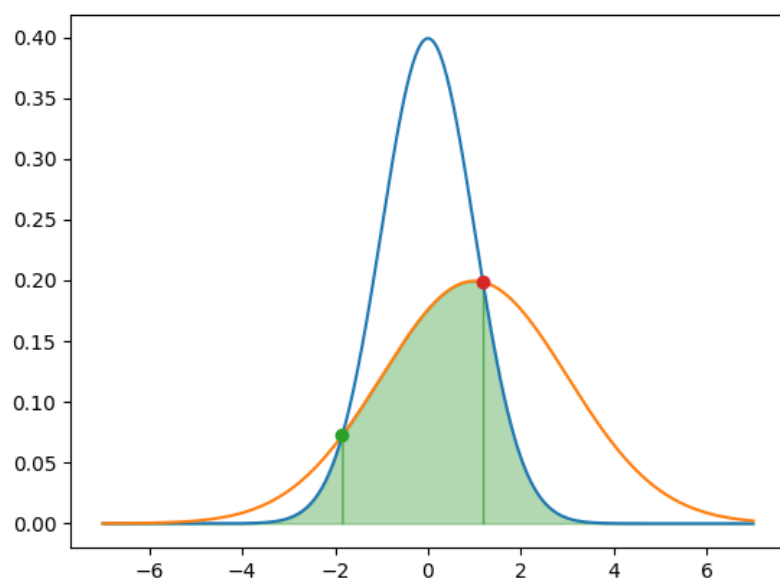


Figure 12 – Overlap between two distributions.

3 Methodology

This chapter describes the methodology used on three main areas of development of this research: embedded systems actions, feature engineering and model training. Section 3.1 describes the operation of the embedded system peripherals, how data is transferred between tasks and how the RTOS operates in data acquisitions and real-time predictions. Then, Section 3.2 presents the experimental setup for testing. Finally, Section 3.3 explains how data are obtained, labeled and separated into training/validation and testing datasets.

3.1 Embedded System Actions

3.1.1 Accelerometer

The InvenSense MPU6050 MotionTracking device combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP), ideal for low power, low cost and high-performance applications. It is embedded in the GY521 module shown in Figure 13 and the reference axes for the accelerometer are shown in Figure 14.

This module connects to the Tiva-C development board by a I²C bus. On start, the module is in low power (sleep) mode and requires a sequence of commands to activate and configure it for proper operation. All register information is available on the register map supplied by the manufacturer. Table 1 shows the power configuration register.

It is configured to disable the sleep state and set the accelerometer clock source to the 8 MHz PLL gyroscope reference. When the module has exited sleep mode, the

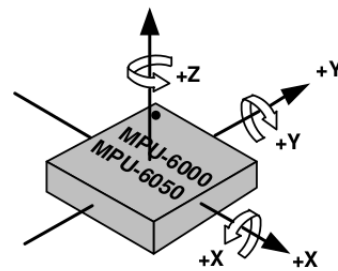
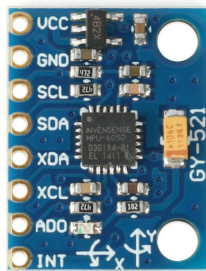


Figure 13 – Accelerometer module GY521. Figure 14 – Accelerometer axis (adapted from (INVENSENSE, 2013)).

Table 1 – Register used to for power management on the MPU.

Register (Hex)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
6B	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

accelerometer full scale range (AFS_SEL) of the 16 bit Analog-to-Digital Converter (ADC) is set to ± 2 g on register 0x3B (Table 2) and is ready to provide data. Internally, the default accelerometer sample rate is 1 kHz and does not require further configuration.

Table 2 – Register used to set the accelerometer full-scale range.

Register (Hex)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
3B	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]	-	-	-	-

The accelerometer data for each axis is made of a 16 bit value in 2-complement, available to read in two 8 bit registers that must be concatenated according to the example for the X axis on Equation 3.1. The resulting value must be a 16 bit signed integer that will later be divided by the respective AFS_SEL value of Table 3 to obtain the actual acceleration in g , using Equation 3.2.

$$AccX = (AccX_MSB \ll 8) + AccX_LSB \quad (3.1)$$

$$AccX_g = \frac{AccX}{AFS_SEL} \quad (3.2)$$

In order to read data from the accelerometer, the I²C burst mode must be used. This form of sequential register read guarantees that data from axis X, Y and Z registers are from the same sample instant. Table 4 shows the registers that must be accessed to retrieve acceleration data (INVENSENSE, 2013).

Table 3 – Accelerometer sensitivity for each full scale setting (INVENSENSE, 2013).

AFS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g

As an example, reading from an axis returned the value 2 on MSB and 210 on LSB. According to Equations 3.1 and 3.2 and using a sensitivity of ± 4 g, the retrieved acceleration on this axis is:

$$Acc = \frac{(2 \ll 8) + 210}{8192} = 0.3525g \quad (3.3)$$

Table 4 – Registers that store the most recent accelerometer measurements.

Register (Hex)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
3B	ACCEL_XOUT[15:8]							
3C	ACCEL_XOUT[7:0]							
3D	ACCEL_YOUT[15:8]							
3E	ACCEL_YOUT[7:0]							
3F	ACCEL_ZOUT[15:8]							
40	ACCEL_ZOUT[7:0]							

3.1.1.1 Accelerometer Calibration

Before use of the accelerometer in any of the operation modes, it must be calibrated to compensate for ground or seat rail mounting inclinations. It is important that the XY plane of the accelerometer module be as parallel as possible to the ground.

In an ideal mounting position, the gravitational force would only interfere on the vertical axis. Mounting the accelerometer with a 5° inclination of the XZ plane in respect to the ground plane would cause the gravitational pull to interfere on measurements with approximately 1 m/s^2 , which is a significant amount. For the plane XZ of Figure 15, the value read by the accelerometer in X_a and Z_a will be

$$X_a = g \sin(\theta) \quad (3.4)$$

and

$$Z_a = g \cos(\theta), \quad (3.5)$$

where θ is the angle between the ground and the accelerometer XZ plane.

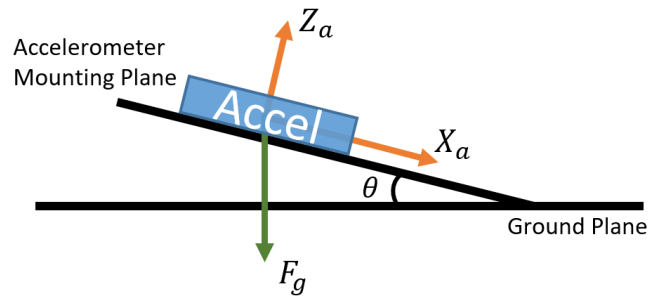


Figure 15 – Accelerometer mounting plane relative to the ground plane.

To mitigate the gravitational acceleration effects on X and Y axis, not only the mounting position must be parallel to the ground, but a calibration procedure is necessary. In practical terms, perfect parallel positioning is not possible, but it should be approximated as good as possible.

The calibration process requires that the vehicle is turned off to avoid unnecessary vibrations. The user pushes a button and the calibration procedure acquires 500 acceleration

samples and calculate the average acceleration for each axis. For the longitudinal and lateral axis, the offset is a simple average calculation. On the vertical axis, the average is subtracted by 1 to compensate the expected gravity acceleration. Those values are called acceleration offsets and are automatically subtracted in all following accelerometer data read by the user.

3.1.2 GPS Module

The NEO-6M GPS module transmits and receives data through a universal asynchronous receiver-transmitter (UART) bus. During initialization, a set of configurations are required to configure the GPS for operation. First, the baudrate is set to 115200 bits/s, which is the fastest speed available. This configuration requires that the firmware also modify its own baudrate to match the new speed. Then, all automatic position messages are disabled to avoid excess data on the UART bus. Data requests are based on a polling strategy where the firmware requests positional information when necessary. An external antenna is used to improve signal quality by attaching it to the top side of the vehicle.

The message structure is based on the NMEA 0183 standard and is returned as follows:

```
$PUBX,00,hhmmss.ss,Latitude,N,Longitude,E,AltRef,NavStat,
Hacc,Vacc,SOG,COG,Vvel,ageC,HDOP,VDOP,TDOP,GU,RU,DR,*cs<CR><LF>
```

Table 5 summarizes the items of interest on the NMEA example above. Time and position data are saved for visualization purposes, speed data is necessary for post-processing of the data and the checksum field is used to guarantee data integrity during parsing.

Table 5 – Fields of interest in the GPS output message, adapted from (U-BLOX, 2013).

Field No.	Example	Format	Name	Unit	Description
2	081350.00	hhmmss.sss	hhmmss.ss	-	UTC Time, Current time
3	4717.113210	ddmm.mmmm	Latitude	-	Latitude, Degrees + minutes
4	N	character	N	-	North (N) / South (S) Indicator
5	00833.915187	dddmm.mmmm	Longitude	-	Longitude, Degrees + minutes
6	E	character	E	-	East (E) / West (W) Indicator
11	0.007	numeric	SOG	km/h	Speed over ground
21	*5B	hexadecimal	cs	-	Checksum

A GPS message positional message is requested two times per second by polling the GPS module for the latest positional data. Upon arriving at the UART input buffer, it triggers an interrupt signaling that data are available to be read by the GPS Read task.

3.1.3 RTOS

The real-time operational system (RTOS) is responsible for managing multiple tasks and shared resources. The firmware has multiple tasks that are described in Table 6.

Table 6 – RTOS tasks.

Task Name	Priority	Description
Accelerometer	3	Read X, Y and Z accelerometer axis and put the data on a queue. Fixed execution period of 10 ms.
GPS Read	3	Receive data from the GPS. On a complete and valid message, put the data on a queue for parsing.
GPS Parse	2	Parse the GPS message field, extracts data of interest and put in a queue to be ready for use.
UART Offload	2	Formats accelerometer and GPS data in a tabulated string and outputs data to the computer using UART.
Prediction	2	Receives data from the accelerometer and classifies the terrain type.

3.1.4 Data Acquisition Mode

The data acquisition mode is used for accelerometer data acquisition at precise intervals of 10 ms (sampling frequency of 100 Hz), obtaining and parsing position data from the GPS and offloading all data to a computer using the serial interface.

Communication between each tasks on the data acquisition flow is executed through queues, which are resources used for task communication in a RTOS, allowing message and data transfers. They are recommended when message exchange in complex programs are necessary, avoiding the use of global variables ([IBRAHIM, 2020](#)).

Tasks GPS Read and GPS Parse each have a common queue called GPSReadQueue. This queue has a defined memory space that allocates the most recent GPS data. When a complete and valid message is received, it is put on the queue for parsing (a message is valid when the GPS message checksum and the calculated checksum match). In this action, a notification is sent to the GPS Parse task that a valid message is ready, and it can be parsed. After parsing is finalized and the critical data has been properly extracted, these new data (as shown in Table 5) are put on a queue that will deliver the message to UART Offload task.

GPS data are updated 50 times slower than the accelerometer, which means that the last valid data must be reused for multiple acceleration reads on the UART Offload task.

UART Offload and Accelerometer tasks communicate through a queue called AccelerometerQueue. This queue has enough space for up to ten accelerometer readings, of the X, Y and Z axis. Since the Accelerometer task is called every 10 ms and the UART Offload has a lower priority, it will not output the data at the same rate of the Accelerometer task. However, data will be correctly sampled and be available on the queue.

When the Offload task is called, the entire queue will be offloaded with the most recent GPS data to the serial interface.

Dataflow of all peripherals on the acquisition mode is illustrated in Figure 16.

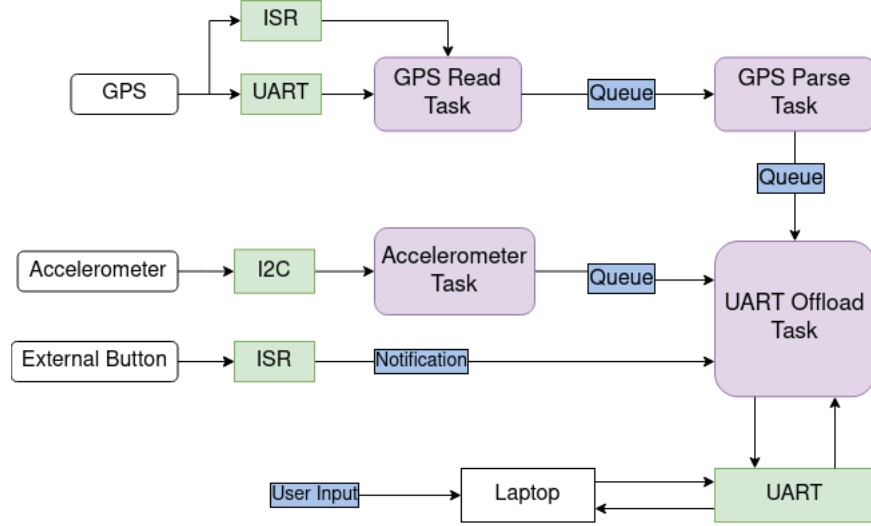


Figure 16 – Dataflow on acquisition mode.

3.1.5 Classification Mode

The classification mode is used for real-time road classification. This mode requires the use of a task called Prediction, while still using the Accelerometer Read task and UART Offload task. Upon receiving acceleration data, a buffer stores 128 samples for a single prediction. After the filling of the buffer, the Prediction Task will be called to calculate the required features (discussed in Section 4.3) and pass them to the ANN for prediction. The returned value from the ANN is shown on the serial interface by the UART Offload task.

Figure 17 illustrates the data flow for the Classification mode.

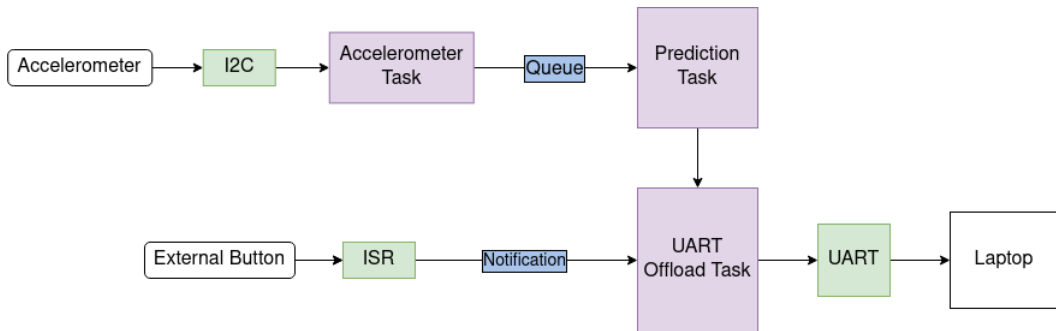


Figure 17 – Data flow on classification mode.

3.1.6 Libraries and Frameworks

The firmware was developed in C and C++ with the GNU compiler for ARM. A CMake script is implemented in various parts of the source code, allowing for faster and modularized compilation. Due to this organization, an Integrated Development Environment (IDE) was not necessary, only a text editor.

Peripheral manipulation on the development board required the use of TivaWare Peripheral Drivers Library and TivaWare Utilities Library, which are Texas Instrument's API (Application Programming Interface) that greatly assist in development by providing header files for all peripherals on board. The GNU Scientific Library (GSL) is used when mathematical processing is required, such as calculating features from a set of samples. Finally, the TensorFlow Micro library is cross-compiled to the ARM architecture, allowing Python trained models to be easily converted to C language and deployed on the embedded system.

3.2 Experimental Setup

Functional tests were executed in a test bench to guarantee that all peripherals are operating as expected. The performance test requires that the equipment can run for at least 30 minutes without errors or data loss. The GPS connectivity test was applied to evaluate how much time the GPS requires to acquire a satellite fix. Initial tests showed that the first satellite fix required over five minutes and sometimes would not acquire a fix at all, leading to an antenna replacement, where a vehicular grade antenna replaced the original one. Figure 18 shows the test setup, where connectors for accelerometer, GPS and the Tiva-C development board are available, allowing for one single board containing all peripherals.

3.3 Data Acquisition

After the initial embedded systems validation tests, the accelerometer module was mounted to the front passenger seat rail of a hatchback vehicle. Figure 19 shows the system assembly and Figure 20 shows the accelerometer module axes relative to the ground.

All data acquisition was executed in the city of Campinas, state of São Paulo, Brazil in two different days, for a total of approximately two hours that generated 24.7 MB of data. The calibration procedure (detailed in Section 3.1.1.1) was executed before data acquisition itself, and a second person assisted on labeling the data.

Some rules were defined for setting the terrain type to 0 (invalid) during data acquisition. Any other normal driving condition in the city such as acceleration, deceleration, coasting and curves are to be used and categorized. The rules are:

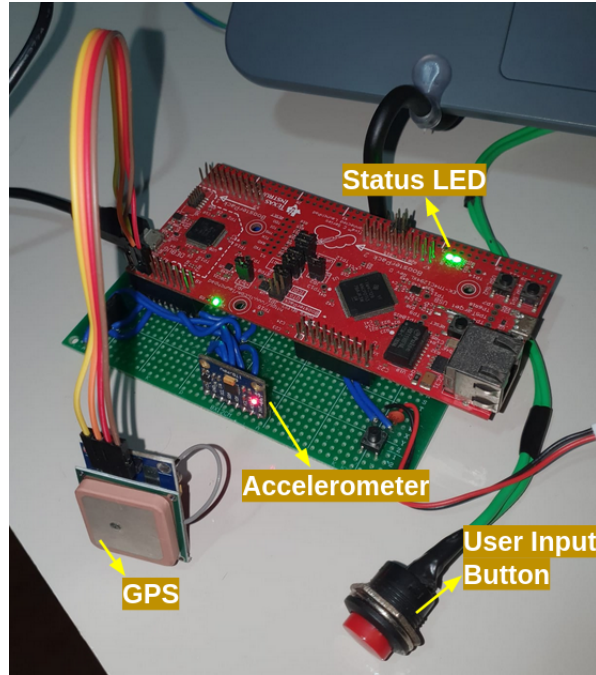


Figure 18 – Test bench.

- Two different and consecutive terrain types are in view. This helps in reducing the number of fast changing terrain types, which can contain few samples and pollute the dataset;
- Vehicle stopped in a traffic light;
- Vehicle stuck in slow traffic (under 10 km/h);
- Vehicle in rolling idle;
- Vehicle in reverse;
- Speed bump ahead.

The acquired data was analyzed by each type of terrain and the percentage of samples in those terrains. This information is helpful to evaluate data balancing between asphalt and paving stone. Too much asphalt data can bias the model or decrease its performance on classifying paving stone. To avoid this problem, the dataset selection for training/validation and testing were randomly selected and then manually moved to balance the paving stone samples.

3.3.1 Training and Testing Datasets

Tables 7 and 8 show which datasets were obtained and how they are separated. The columns paving stone, asphalt or invalid represent the labeled percentage of actual

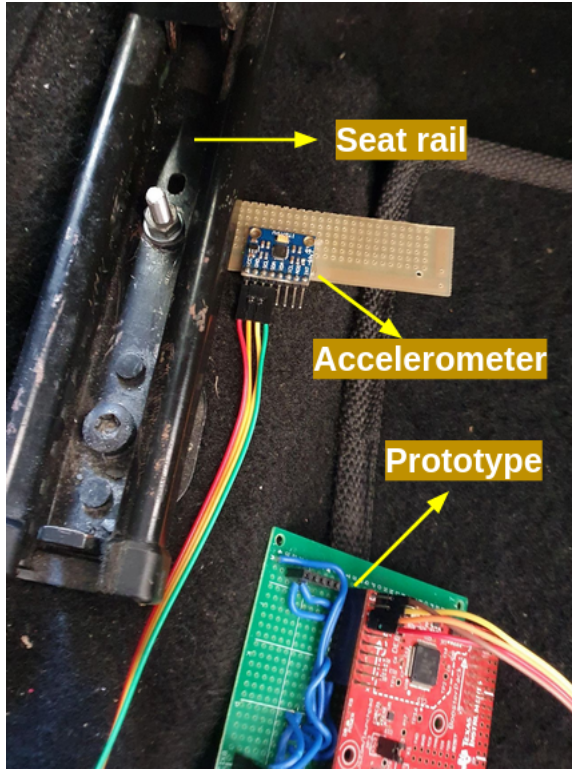


Figure 19 – Vehicle assembly.

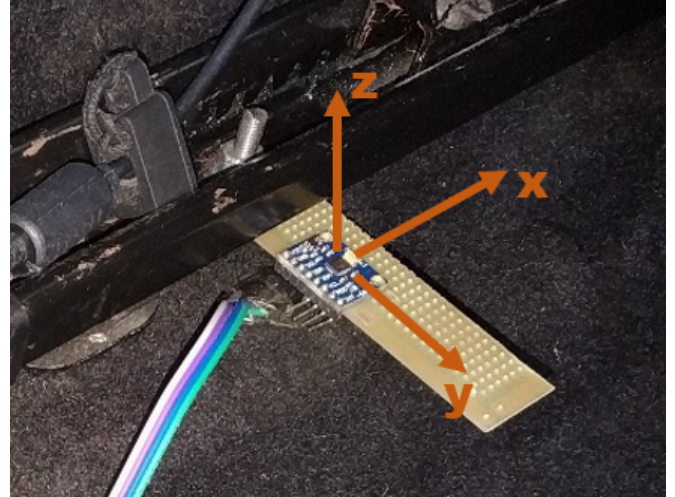


Figure 20 – Accelerometer axes relative to the ground.

terrain during acquisition, where invalid can represent that the car is stopped or any other scenario where data was not relevant.

Table 7 – Training and validation dataset composition.

Filename	Rows	Acq. Time [min]	Paving Stone	Asphalt	Invalid
Campinas_2022-04-16_16-15-22	63227	10.54	24.2%	37.50%	38.30%
Campinas_2022-10-16_18-44-38	31013	5.17	31.80%	34.20%	34.00%
Campinas_2022-10-16_17-35-36	3871	0.65	65.60%	0.00%	34.40%
Campinas_2022-10-16_17-41-21	26160	4.36	69.80%	3.80%	26.40%
Campinas_2022-04-16_15-49-50	27685	4.61	50.60%	34.40%	15.00%
Campinas_2022-10-16_18-50-03	19656	3.28	60.70%	33.30%	5.90%
Campinas_2022-10-16_17-53-39	37290	6.22	46.20%	0.00%	53.80%
Campinas_2022-04-16_15-34-43	22727	3.79	0.00%	73.70%	26.30%
Campinas_2022-04-16_15-52-59	18042	3.01	6.70%	84.70%	8.60%
Campinas_2022-04-16_16-04-43	12878	2.15	0.00%	66.2%	33.8%
Campinas_2022-10-16_17-31-20	8656	1.44	50.40%	12.90%	36.70%
Campinas_2022-04-16_16-02-10	52984	8.83	43.90%	36.60%	19.50%
Total	324189	54.03	33.39%	34.69%	28.92%

Table 8 – Testing dataset composition.

Filename	Rows	Acq. Time [min]	Paving Stone	Asphalt	Invalid
Campinas_2022-04-16_15-30-44	32749	5.46	0.00%	95.60%	4.40%
Campinas_2022-10-16_18-46-37	9176	1.53	53.70%	15.80%	30.60%
Campinas_2022-10-16_17-34-02	5724	0.95	59.40%	20.70%	19.90%
Campinas_2022-10-16_17-43-30	11027	1.84	69.50%	14.90%	15.60%
Campinas_2022-10-16_17-57-57	21477	3.58	54.30%	15.80%	29.90%
Total	80153	13.36	34.50%	48.63%	16.88%

In the training/validation dataset, 54.03 minutes of acquisition supplies almost 325.000 samples where paving stone samples represent 33.39% of the available data, asphalt represents 34.69% and the remaining are invalid data. In the test dataset, 13.36 minutes of data represents approximately 80.000 samples, where paving stone represents 34.50% of the samples and asphalt represents 48.63%.

After pre-processing the data, the training/validation and test datasets are each joined together in one single file, allowing for easier data manipulation capabilities since all training or testing samples are in a single place.

4 Results

This section describes the data analysis and feature selection of the datasets, that are used on training and testing of the ANN. Section 4.1 describes data loading, parsing and the initial feature generation from raw data. Section 4.2 explains which metrics were extracted from the available data, but not yet selected to be used in training the model. Next, Section 4.3 contains the process of analyzing each metric individually by their correlation and overlapping coefficient, filtering and selecting the best features to train the model. Finally, Section 4.4 shows the train/test split of the data, the training strategy used to select a good model, and the results obtained from the test dataset.

4.1 Preprocessing

Each available dataset is imported as it was recorded during data acquisitions. The algorithm executes a sequence of operations that will open the file, remove empty lines and convert the data to the correct type (integers, floats, strings). After the first part of pre-processing, sequential numbers are assigned to the samples for easier identification and some conversions are executed: latitude and longitude are converted from the NMEA format to geographic coordinates in degrees, minutes and seconds. Then, filtered data are generated to assist on future feature calculation on all three axis: rolling average (RA) and rolling variance (RV) with window sizes 5 and 20.

Figure 21 shows terrain, speed and X, Y and Z acceleration data from a dataset of approximately 10 minutes using the rolling average of window 5.

The terrain data consist of numbers from 0 to 3 where each represents a terrain according to Table 9. This value is set as label and represents the real terrain that vehicle was moving during the data acquisition. It is available in the datasets as the “Terrain” column.

Table 9 – Values representing terrain.

Number	Terrain
0	Undefined or Stopped
1	Asphalt
2	Paving Stone
3	Speed bump

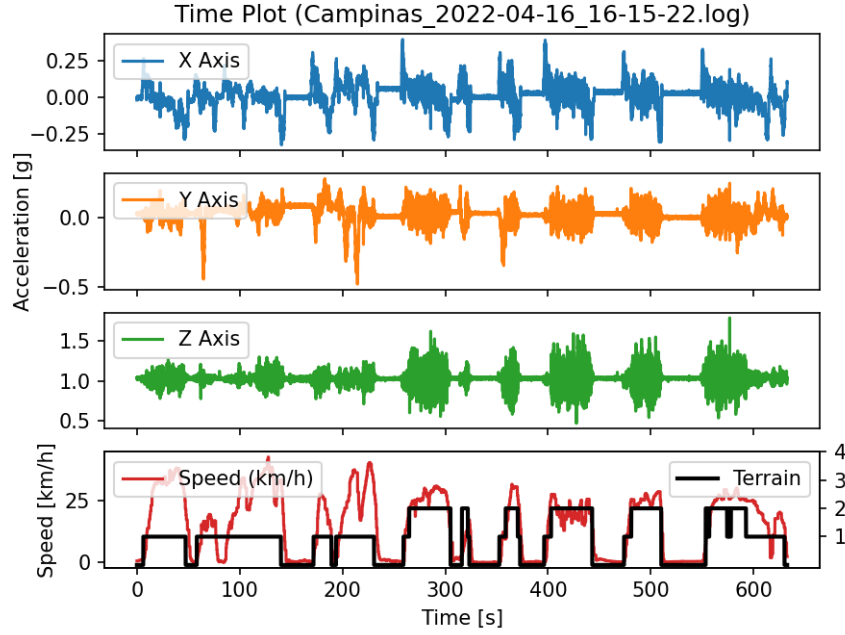


Figure 21 – Data acquisition.

4.2 Feature Extraction

In this work, a metric is a statistical value calculated from a number of samples of accelerometer data, which can be either on time domain or frequency domain. On the other hand, a feature is a metric used as input to the machine learning model.

The selection of features for classifying the type of pavement starts with the literature review in Section 1.2. The most common metrics in the literature were implemented in the software used for training the ANN. Other statistical metrics were analyzed and added to the software, resulting in a total of 22 metrics for each N samples in a dataset:

- Average acceleration per axis (Acc_Mean);
- Acceleration Root Mean Square (RMS) per axis (Acc_RMS);
- Peak acceleration per axis (Acc_Peak);
- Standard deviation using the rolling average acceleration per axis (Acc_RA5_Stddev);
- Peak acceleration using the rolling variance acceleration per axis (Acc_RV_Peak);
- Peak acceleration using the rolling average acceleration of the vertical axis (AccZ_RA5_Peak);
- Average of all three axis (Acc_Mean);
- RMS of the acceleration in each of the three axis (Acc_RMS);

- Maximum amplitude peak-to-peak in the vertical axis (Z_Max_Pk2pk);
- Average low frequency (0-15 Hz) amplitude of the Z axis FFT (Acc_FFT_LF);
- Average high frequency (16-50 Hz) amplitude of the Z axis FFT (Acc_FFT_HF);
- Average vehicle speed (Spd_Mean).

Each metric is calculated for each 128 samples, which is equivalent to 1.28 s of data acquisition due to the 100 Hz sampling rate. This value is based on previous work detailed in Section 1.2 and modified to better fit the requirements of FFT calculation, where the number of samples must be a power of 2. For each dataset available, the metrics are saved in a single table that will be used as the complete dataset for training or testing.

4.3 Feature Selection

Training the ANN for an embedded application requires the selection of features that can accurately classify the data while using the smallest number of neurons, due to processing power limitations.

Since the terrains of interest are asphalt and paving stone, data that does not belong to one of those classes are removed from the metrics. The metrics for feature selection are also filtered by the vehicle speed, where speeds below 10 km/h are removed due to similarities with traffic jams and samples above 50 km/h are also removed from the dataset, mostly because it is a speed with very few samples as shown in the Speed Mean distribution plot in Figure 28.

Feature selection is based on the filtering method using the Pearson correlation, as described in Section 2.7.1. It is applied on the training dataset where each of the 22 possible features that are compared using their correlation to each other. If the correlation is greater than or equal to 0.9, it is removed from the feature list because it does not add new information and will decrease the model complexity.

Figure 22 shows the correlation matrix for the training data after removing 10 metrics that were above the correlation threshold. The removed metrics were: AccZ_FFT_HF, AccZ_Mean, Acc_Mean, AccZ_RA5_Stddev, AccZ_FFT_LF, AccZ_Peak, AccZ_RA5_Peak, Z_Max_Pk2pk, AccZ_RMS, Acc_RMS.

The correlation matrix is mirrored on its upper right part. The first row represents the correlation of AccX_Mean in respect to all other metrics, such as the AccY_RMS with a 0.05 correlation, which in practice means that the data is not correlated. Since all features that showed correlation above 0.9 were removed, the highest correlated datasets are AccX_Peak and AccX_RV_Peak with a 0.8 correlation.

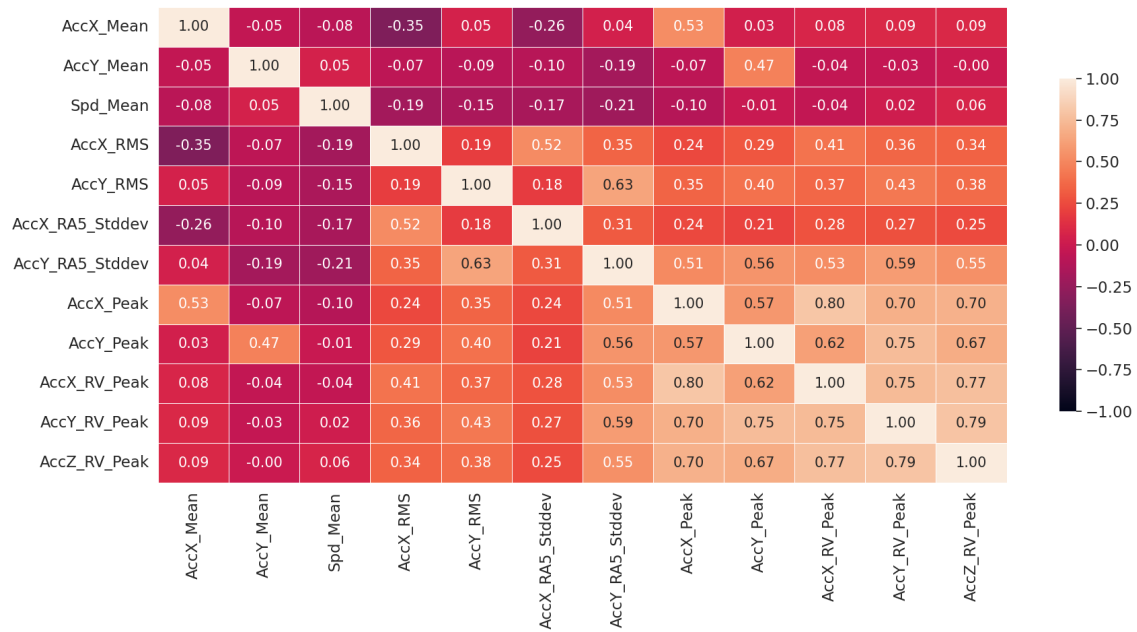


Figure 22 – Correlation matrix of metrics after filtering by correlation.

Using the 12 remaining metrics, a distribution plot is generated for each one to compare values for two of the terrain types of interest: asphalt and paving stone. The figures below show that feature value on the X axis and the number of occurrences of those values in the Y axis. Each color represents a terrain.

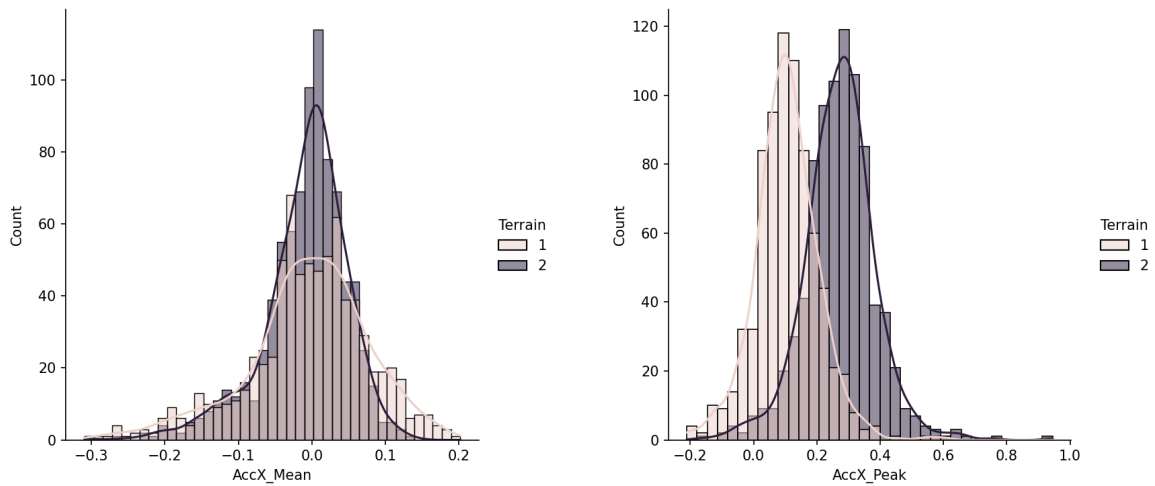


Figure 23 – Distribution plot of AccX-Mean and AccX-Peak.

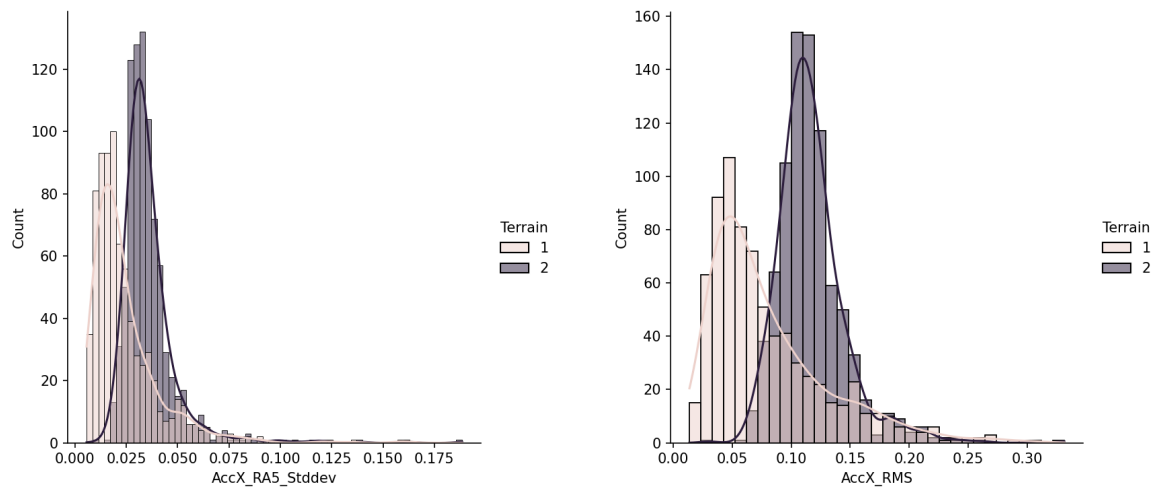


Figure 24 – Distribution plot of AccX-RA5-Stddev and AccX-RMS.

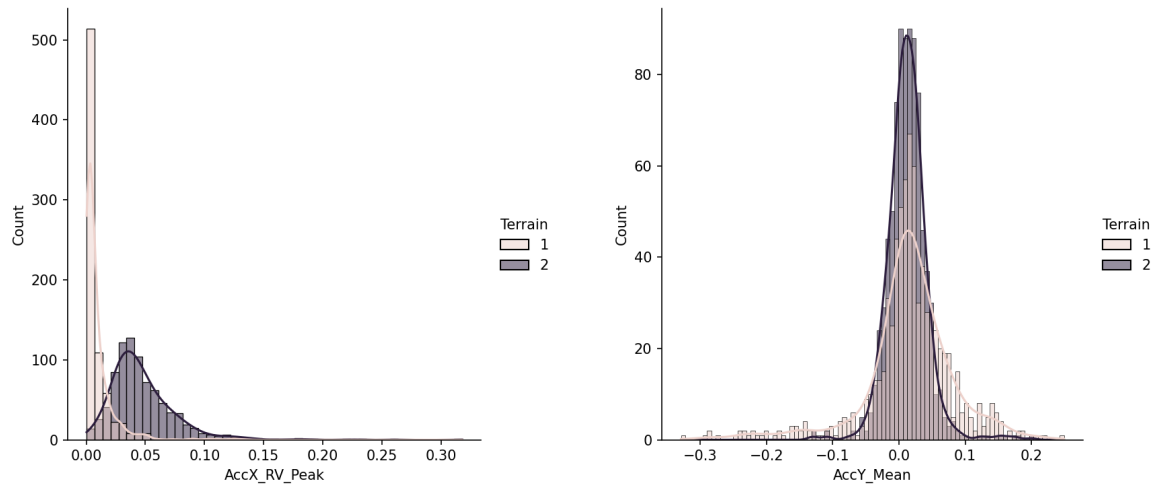


Figure 25 – Distribution plot of AccX-RV-Peak and AccY-Mean.

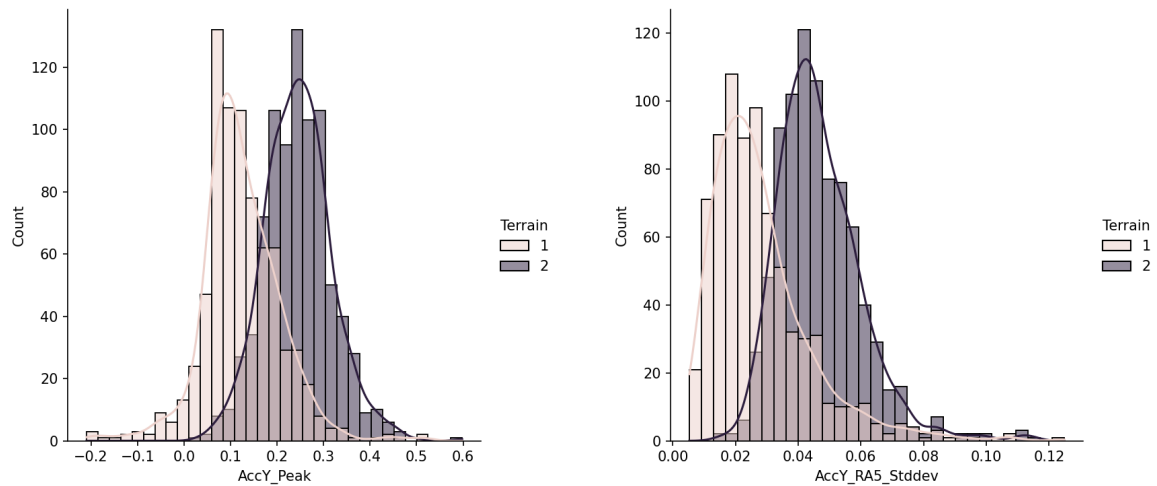


Figure 26 – Distribution plot of AccY-Peak and AccY-RA5-Stddev.

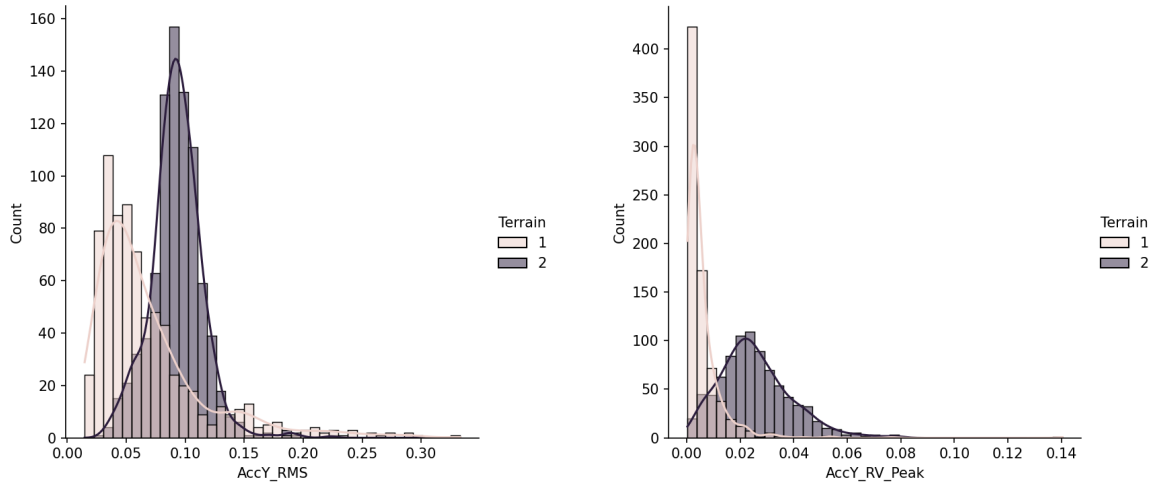


Figure 27 – Distribution plot of AccY-RMS and AccY-RV-Peak.

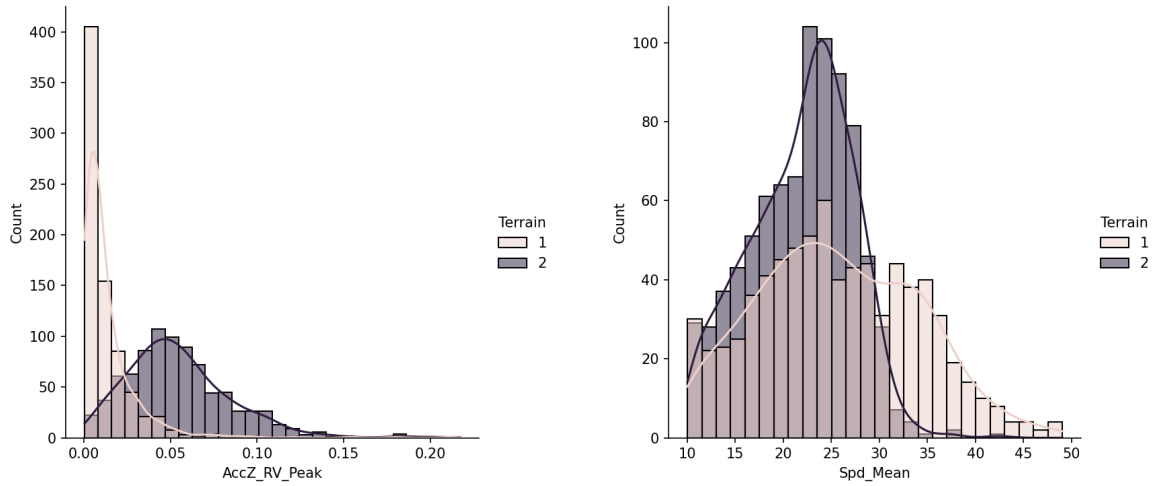


Figure 28 – Distribution plot of AccZ-RV-Peak and Spd-Mean.

All 12 features are analyzed by their Normal distribution in each of the two classes by using the overlapping coefficient detailed in Section 2.7.2. Table 10 shows the overlapping coefficient of the areas between each distribution, as well as their mean and standard deviation.

Table 10 – Normal distribution mean, standard deviation and overlapping coefficient between asphalt (1) and paving stone (2) samples.

Feature	Mean (μ) [1, 2]	Std. Dev. (σ) [1, 2]	Overlapping Coefficient
AccX_Mean	[-0.007, -0.009]	[0.085, 0.057]	0.807
AccY_Mean	[0.011, 0.013]	[0.067, 0.03]	0.628
Spd_Mean	[25.888, 21.682]	[8.367, 5.141]	0.684
AccX_RMS	[0.083, 0.117]	[0.005, 0.028]	0.599
AccY_RMS	[0.066, 0.093]	[0.047, 0.022]	0.561
AccX_RA5_Stddev	[0.025, 0.036]	[0.018, 0.014]	0.722
AccY_RA5_Stddev	[0.029, 0.047]	[0.016, 0.013]	0.524
AccX_Peak	[0.109, 0.275]	[0.102, 0.111]	0.434
AccY_Peak	[0.124, 0.247]	[0.083, 0.072]	0.426
AccX_RV_Peak	[0.010, 0.048]	[0.015, 0.029]	0.359
AccY_RV_Peak	[0.006, 0.026]	[0.008, 0.013]	0.356
AccZ_RV_Peak	[0.015, 0.055]	[0.019, 0.030]	0.390

For further filtering of the features that will be used in training the ANN, all classes with an overlapping coefficient above 0.5 will be removed from the dataset.

The following are the features selected to train the model: AccX_Peak, AccY_Peak, AccX_RV_Peak, AccY_RV_Peak, AccZ_RV_Peak.

Figures 29 and 30 show the selected features on the upper plot and the respective vehicle speed and terrain on the lower plot, for a specific sample window of a dataset. When comparing the amplitudes in each feature, a clear difference can be noticed where paving stone shows significant higher values than asphalt. For illustration purposes, the plots were scaled between [0,1] and filtered with a moving mean of window 3.

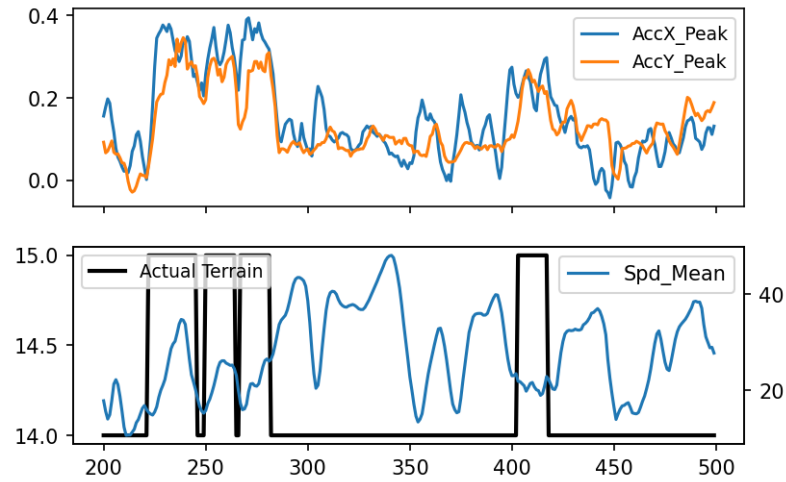


Figure 29 – Visualization of the selected features.

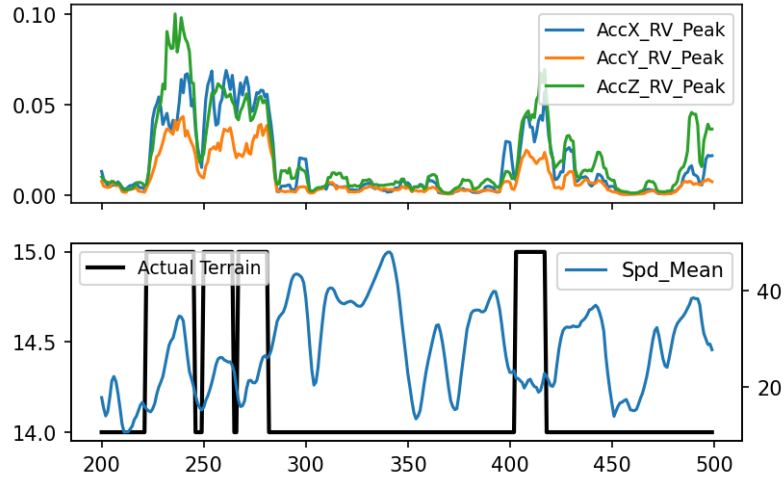


Figure 30 – Visualization of the selected features (continuation).

4.4 Training the Artificial Neural Network

This section describes the training process for the ANN using all data available from Section 4.3. Section 4.4.1 discuss the number of samples available and how they are separated in training and test datasets. Sections 4.4.2 and 4.4.3 discuss the training strategy that selects the optimal configuration for the model. Finally, Section 4.4.4 shows the results on the test dataset.

4.4.1 Data Scaling and Splitting

The feature selection procedure generated 1601 samples, each one containing 5 features to train the ANN. There is no definitive ratio in which the samples should be divided in training/validation. A common value used is 80/20 (80% training, 20% testing) which comes from the Pareto Principle (REH, 2019), but values such as 70/30 and 90/10 are also used and selected, depending on multiple factors such as total number of samples, variance and noise. This implementation uses a 60/40 ratio, resulting in 960 samples for training and 640 samples for validation. This value was selected based on manual testing of the training performance.

For each feature on the training set, a transformation is obtained using Equation 4.1. New samples can be scaled using this transformation by applying Equation 4.2, that scales the sample to a range of $[0,1]$. This transformation is required to make sure all features are on the same value range and no bias is added to the model because of data scaling differences. This type of scaling is applied using the MinMaxScaler from the Sklearn library. It is robust to small standard deviations of features and preserves zero entries in sparse data (PEDREGOSA et al., 2011). After applying the scaling on the training dataset, the scaler is saved to be later used on the test dataset.

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4.1)$$

$$X_{scaled} = X_{std}(max - min) + min \quad (4.2)$$

where X_{min} and X_{max} is the minimum and maximum values in the feature and min, max is the desired range, normally $[0,1]$.

As this is a binary classification, the output value for the ANN will come from a single output neuron representing the terrain, which will have a value between 0 and 1 representing the confidence in the classification. Values closer to zero mean corresponds to a high confidence that the terrain is asphalt, where an output closer to one means a high confidence that the terrain is paved in stone.

4.4.2 Training Strategy

This work used a Multi Layer Perceptron (MLP) architecture with two different configurations: the first configuration contains one hidden layer, while the second configuration contains two hidden layers. The training process evaluates multiple combinations of neurons in each configuration.

The number of epochs is set to 300, based on manual testing of this architecture where the number of epochs are sufficient before overfitting occurs and allows analysis of the epochs with optimal loss. Each training configuration is set for early stopping: if the validation loss does not improve (decrease in value) for 30 consecutive epochs, the training is stopped. The learning rate was selected to be 0.03 on the first configuration and 0.01 on the second, all values selected based on empirical tests that resulted in less noise of the loss values in the training process. Table 11 summarizes the training information.

Table 11 – MLP model parameters.

Parameter	Value
Hidden Layer Activation	ReLu*
Output Layer Activation	Sigmoid
Model Optimizer	Adam**
Loss	Binary Cross-entropy
Epochs	300
Early Stopping	Val. Loss - 30 epochs
Learning rate	0.03, 0.01

*ReLU stands for Rectified Linear Unit, a common activation function for ANN.

**The Adam optimizer is a stochastic gradient descent method (KINGMA; BA, 2014).

The first configuration evaluates the training performance for 8 scenarios of different number of neurons: 60, 70, 80, 90, 100, 110, 120 and 130. The second configuration tests 25 combinations between the first and second hidden layers, where the first layer has

scenarios of 140, 150, 160, 170 and 180 neurons that will be tested with the second layer with scenarios of 20, 30, 40, 50 and 60 neurons.

First, a comparison is executed between the two configurations to assess if there is any advantage in using a two hidden layer configuration instead of a single hidden layer. Then, the best model will be selected to run on the test dataset based on the accuracy and loss plots, where the optimal threshold is selected.

4.4.3 Selecting Best MLP Configuration

Starting with the first configuration (one hidden layer), Figures 31 and 32 show the training and validation accuracy, respectively. The best results were obtained for the 80 neurons scenario at epoch 221, with training accuracy of 0.9127 and validation accuracy of 0.9102. The training loss was 0.2359 and validation loss 0.2600 and can be seen on Figure 33.

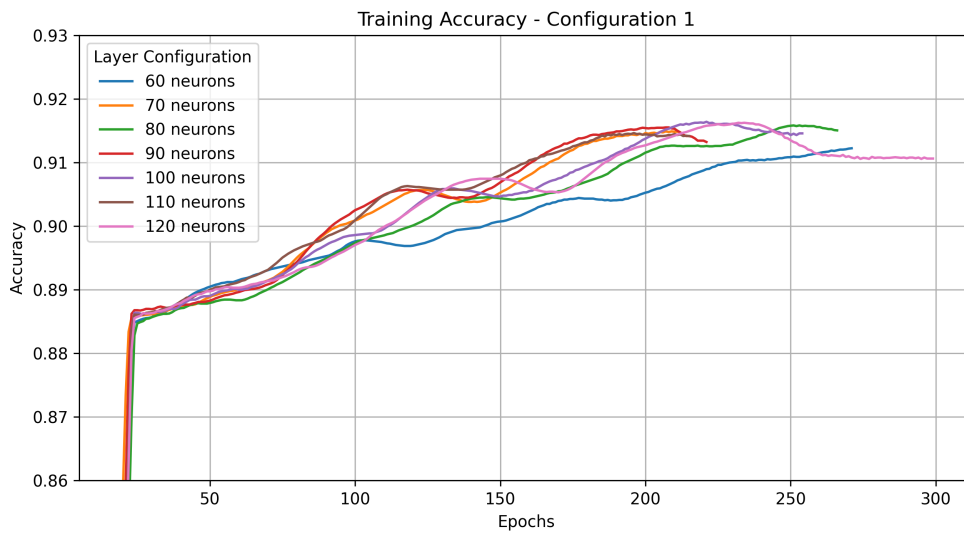


Figure 31 – Training accuracy for configuration one.

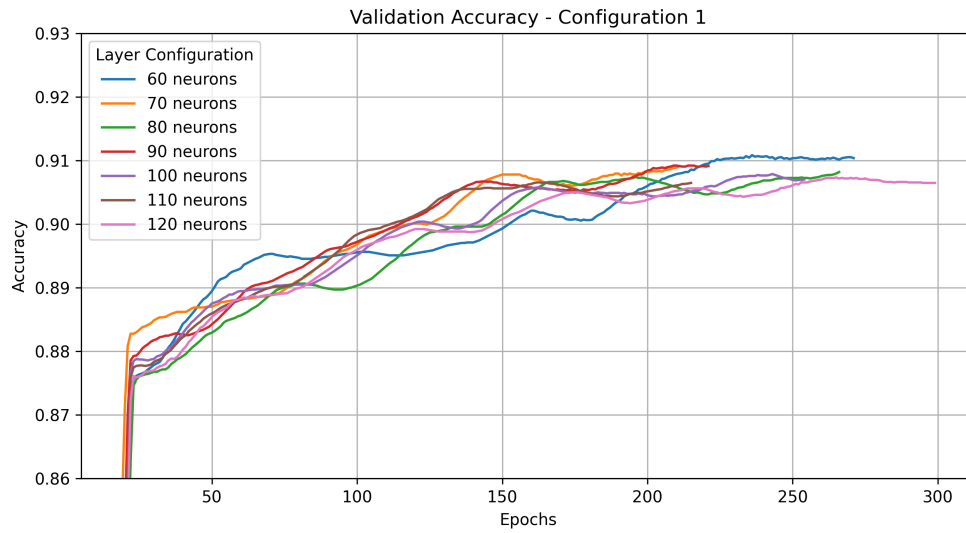


Figure 32 – Validation accuracy for configuration one.

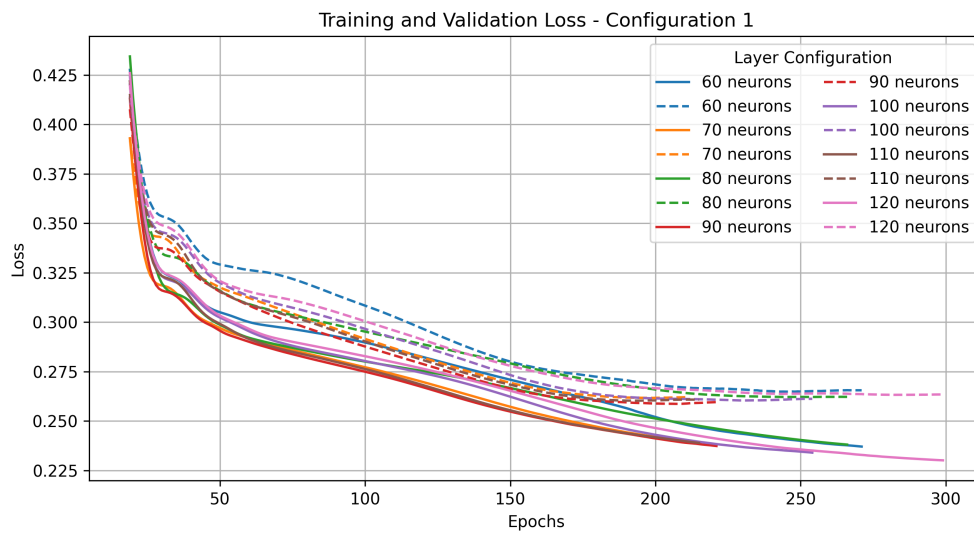


Figure 33 – Loss values for training and validation (dotted curves) for configuration one.

Figures 34 and 35 show the training and validation accuracy for configuration two (two hidden layers). The best results were obtained for 140 neurons on the first layer and 50 neurons on the second layer. Training stopped at epoch 286 with training accuracy of 0.9145 and validation accuracy of 0.9109. The training and validation losses were 0.2185 and 0.2509, respectively, and are available in Figures 36 and 37.

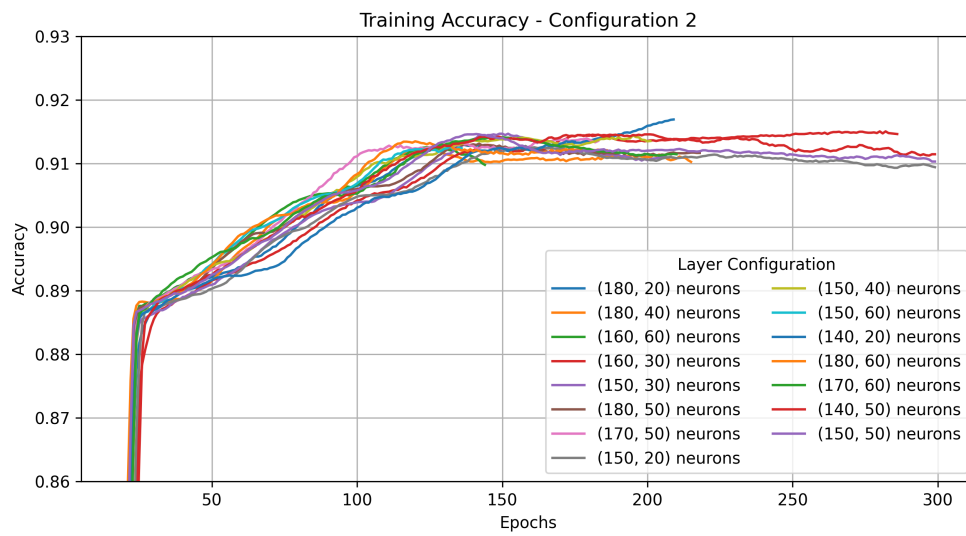


Figure 34 – Training accuracy for configuration two.

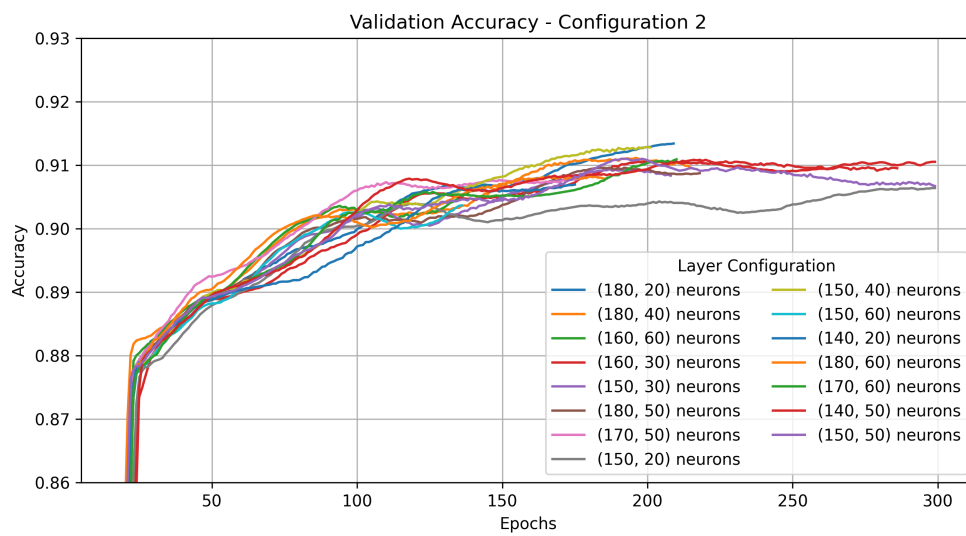


Figure 35 – Validation accuracy for configuration two.

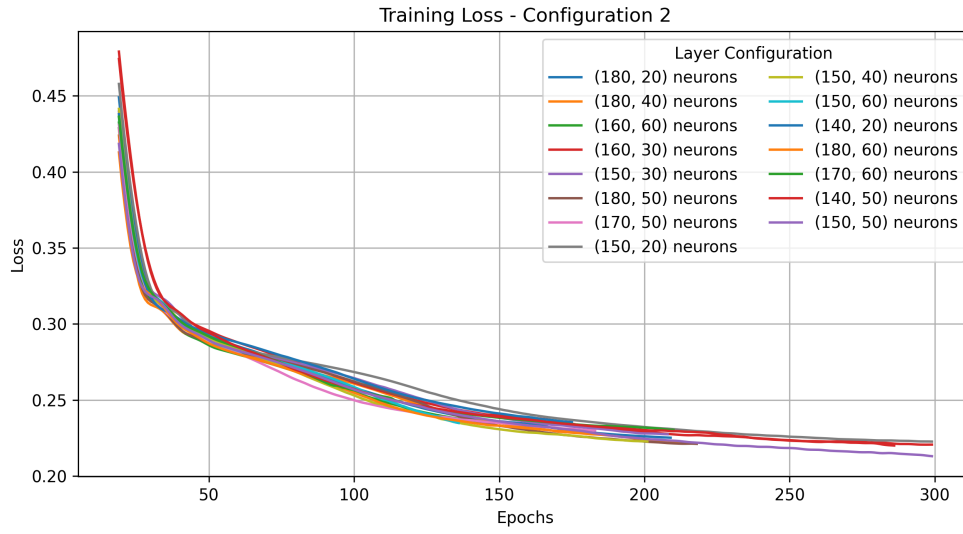


Figure 36 – Loss values for training in configuration two.

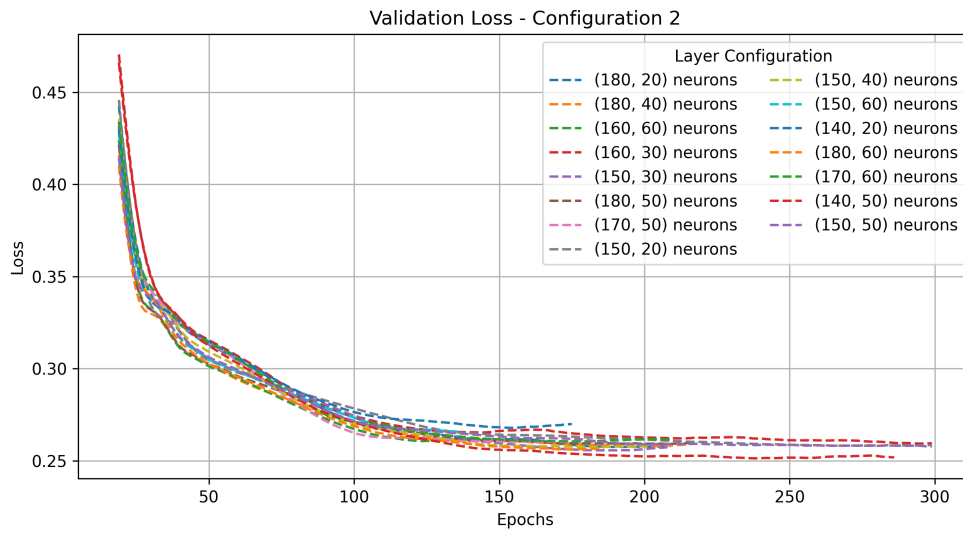


Figure 37 – Loss values for validation in configuration two.

Table 12 summarizes the results of the best epochs for both configurations and Figures 38 to 40 compare the two final results.

Table 12 – Summary of the best training results for both configurations.

Parameter	Configuration 1	Configuration 2
Neurons	80	140 (layer 1) and 50 (layer 2)
Final Epoch	221	286
Training Accuracy	0.9127	0.9145
Validation Accuracy	0.9102	0.9109
Training Loss	0.2359	0.2185
Validation Loss	0.2600	0.2509
Total Parameters	561	7941

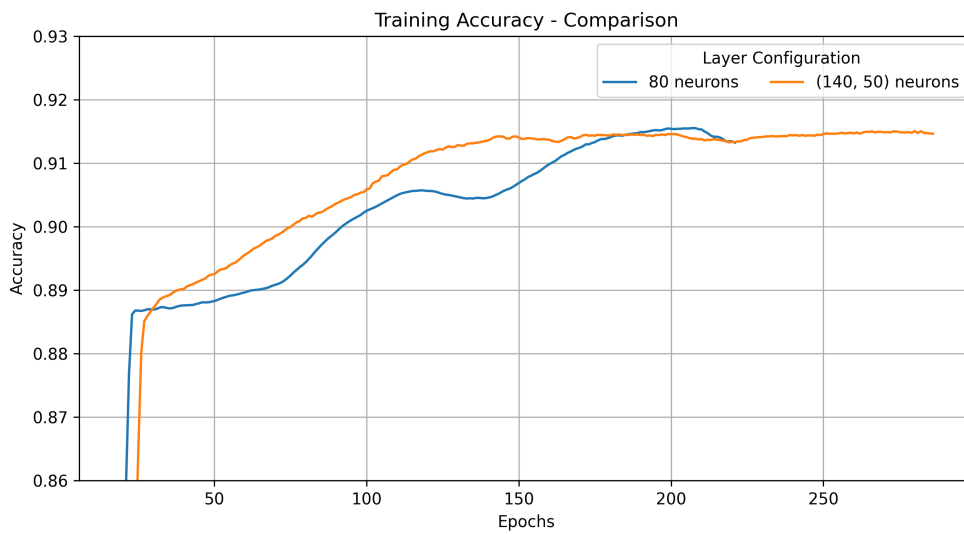


Figure 38 – Comparing training accuracy of the two training configurations.

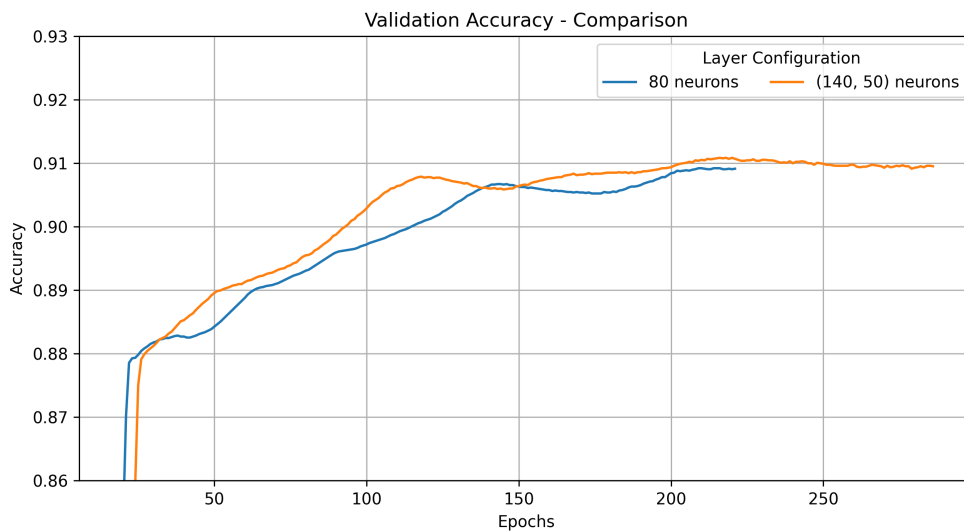


Figure 39 – Comparing validation accuracy of the two training configurations.

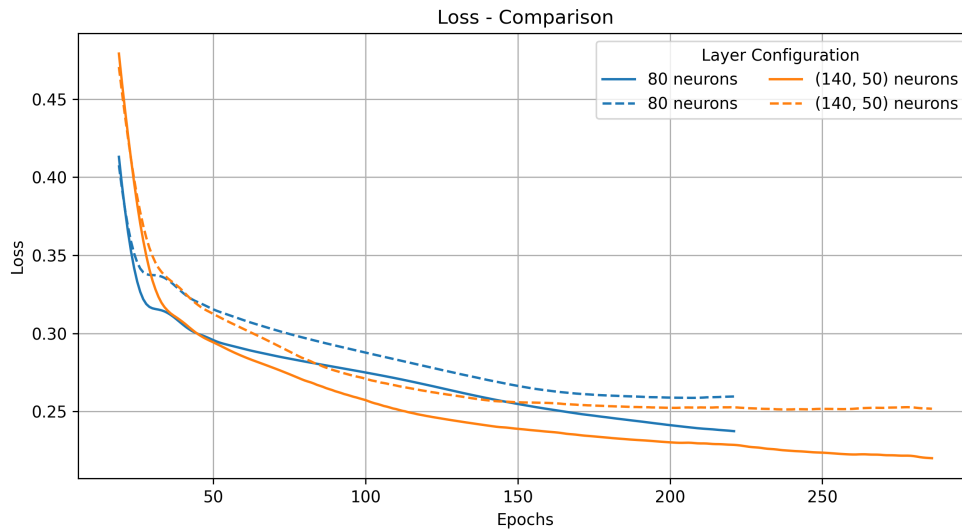


Figure 40 – Training and validation loss compared for the two configurations (dotted line represents validation).

Both configurations presented similar results. In absolute terms, the layer configuration 2 showed better results by a small margin. However, taking into consideration the embedded scenario where this model will be deployed, the small difference in accuracy does not justify the 14 times increase in the number of parameters. The first configuration of 80 neurons will be used for test and embedding.

4.4.4 Test Dataset Results

The ROC curve is required to analyze the best threshold to identify the terrain type. The closest point between coordinates (0, 1) and the curve represents the optimal threshold. Figure 41 shows the ROC curve for the selected model on the test dataset. The best threshold found is 0.753: if the model output is above this value, the terrain can be classified as paving stone. Otherwise, it can be classified as asphalt.

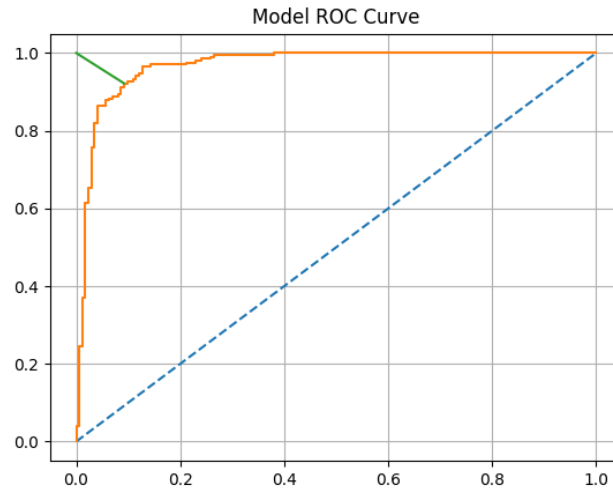


Figure 41 – Selected model ROC curve.

Using the optimal threshold, the test dataset is analyzed and the results are presented in Table 13. The overall accuracy of the model is 0.92.

Table 13 – Metrics for the test dataset using threshold 0.75.

Class	Precision	Recall	F1-Score	Samples
asphalt	0.91	0.91	0.91	174
stone	0.93	0.92	0.92	205

4.5 Validating Real-Time Prediction

The TensorFlow model obtained for the one-hidden-layer configuration is converted to a TensorFlow Micro format and added to the firmware.

The validation requires the real-time prediction mode detailed in Section 3.1.5, which outputs to the user the most recent prediction value and normal acquisition data. Comparing the real-time prediction values to the prediction on a computer using the same data, it is possible to validate if the real-time prediction showed correct results and also analyze its performance. The validation steps are:

- Load the model on the firmware;
- Record more driving sessions similar to the ones of Section 3.3, but now containing the real-time prediction;
- Execute prediction on the new data and compare to the embedded model prediction.

4.5.1 Comparing Results

The new dataset is analyzed using a computer, where it is expected that the real-time prediction value be the same as the one processed by the trained model in the computer. This dataset was obtained in normal city driving scenarios, with average speed below 60 km/h, following the same data acquisition rules of Section 3.3.

The terrain, prediction and X , Y and Z acceleration columns are necessary for validation. The plots in Figures 42 to Figure 47 show the actual terrain on the top plot and on the bottom, the real-time prediction in blue and the post-processed prediction in orange. The model is using the threshold value of 0.753 defined in Section 4.4.4 to evaluate a prediction as asphalt or paving stone. Overall accuracy of this dataset is 92.84% with more detailed information on Table 14.

Table 14 – Real-time prediction dataset.

Filename	Accuracy [%]	TN	FP	FN	TP
Campinas_2023-03-11_16-20-44.log	92.72	1632	128	0	0
Campinas_2023-03-11_16-22-32.log	90.44	6055	640	0	0
Campinas_2023-03-11_16-24-38.log	97.6	10424	256	0	0
Campinas_2023-03-11_16-26-46.log	98.87	11205	128	0	0
Campinas_2023-03-11_16-30-11.log	94.93	7321	462	0	1330
Campinas_2023-03-11_16-31-11.log	100	4503	4503	0	0
Campinas_2023-03-11_16-34-13.log	88.83	4696	490	891	6294
Campinas_2023-03-11_16-36-31.log	93.29	2023	274	128	3566
Campinas_2023-03-11_16-40-38.log	97.35	4393	0	238	4352
Campinas_2023-03-11_16-42-42.log	94.57	8594	412	226	2532
Campinas_2023-03-11_16-44-26.log	76.28	1640	357	1674	4891
Campinas_2023-03-11_16-45-27.log	89.9	1562	278	197	2666
Campinas_2023-03-11_17-07-34.log	91.96	8293	1039	435	8561
Campinas_2023-03-11_17-02-20.log	96.42	56505	1458	662	718
Campinas_2023-03-11_17-10-07.log	89.49	11168	640	807	1152
Average Accuracy	92.84				

Figures 42 and 43 show an asphalt terrain where the model predicted paving stones at some points. This was found to be caused due to potholes, manhole covers and general road issues. Average accuracy of the those two sets is 94.02%.

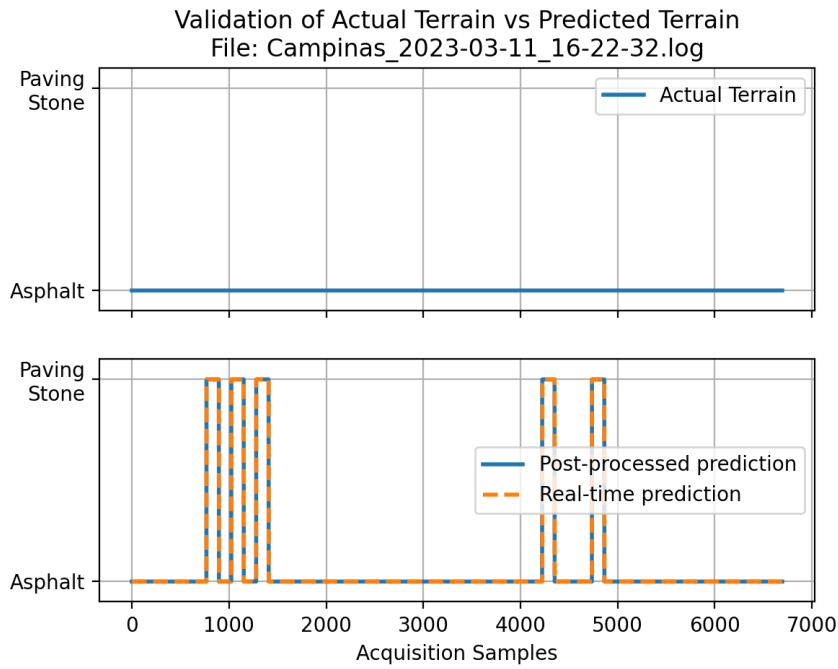


Figure 42 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.

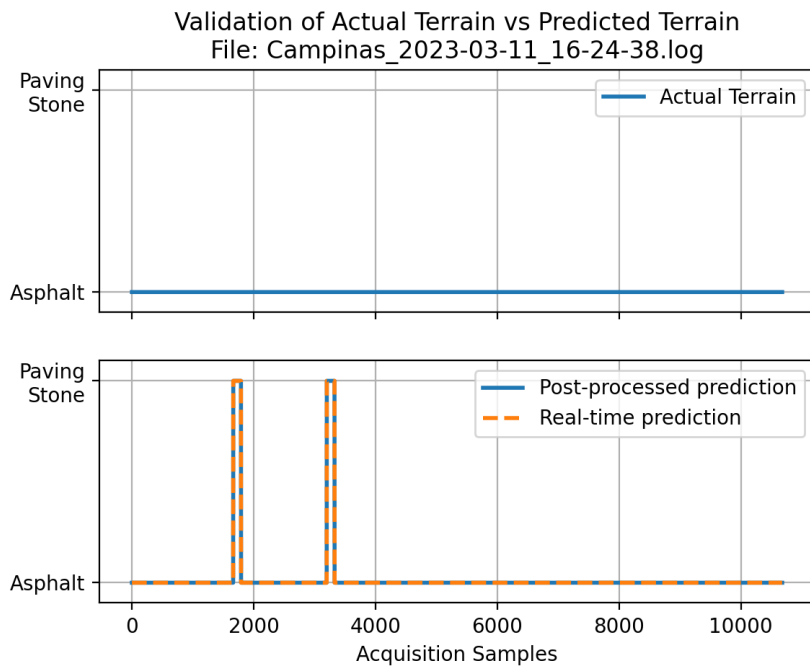


Figure 43 – Comparison of real-time and computer analyzed prediction output.

Figures 44 and 45 show mixed terrain acquisitions. The same issue noted before happens in this scenario, where paving stone is predicted when the terrain is asphalt. The average recall for those two sets is 94% on paving stone and 87% on asphalt. Overall accuracy for the two sets is 91%.

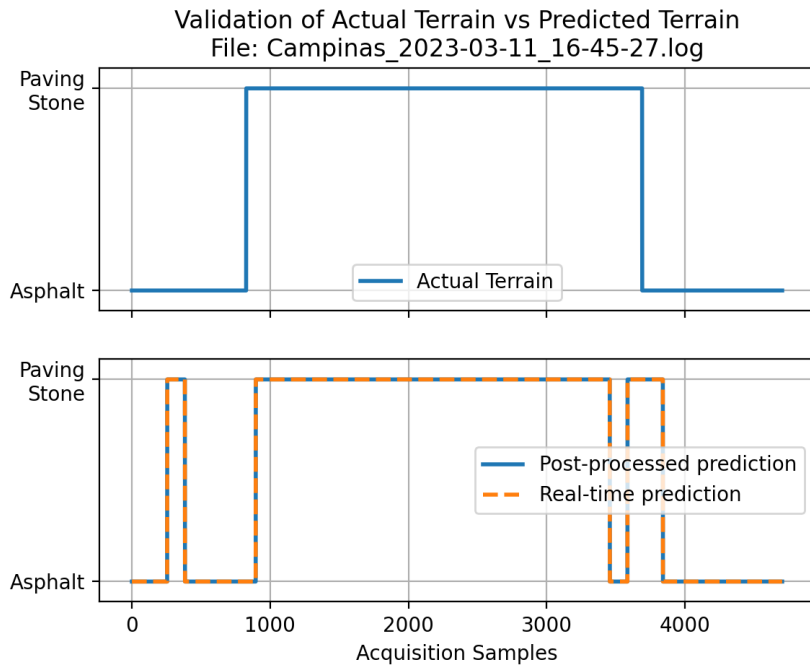


Figure 44 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.

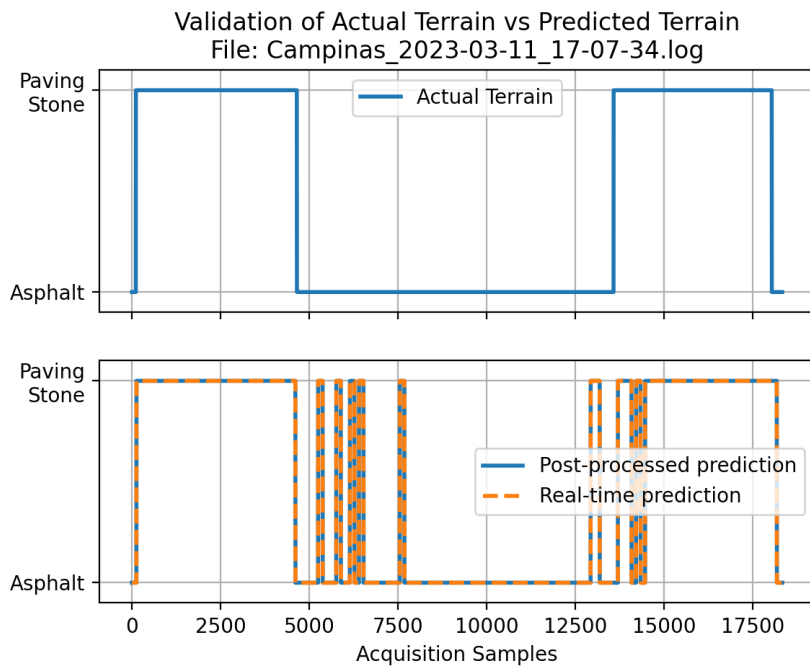


Figure 45 – Comparison of real-time and computer analyzed prediction output.

Finally, two long acquisition sets were obtained and are shown in Figures 46 and 47. Those were long drives across the city center with no traffic. Overall accuracy of both sets is 92.96%. However, the overall weighted precision is 89% due to the incorrectly predicted paving stone samples.

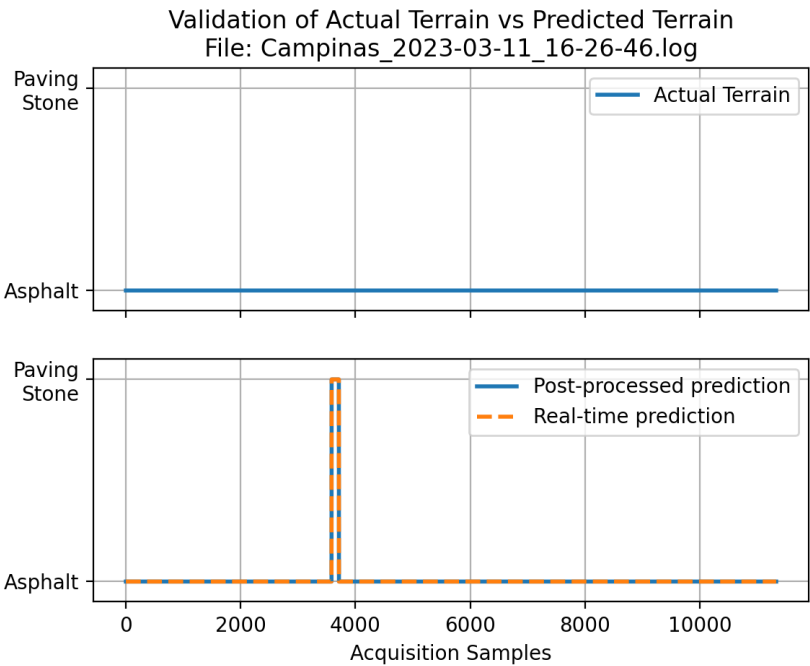


Figure 46 – Actual terrain asphalt, however road quality appears to cause a wrong prediction.

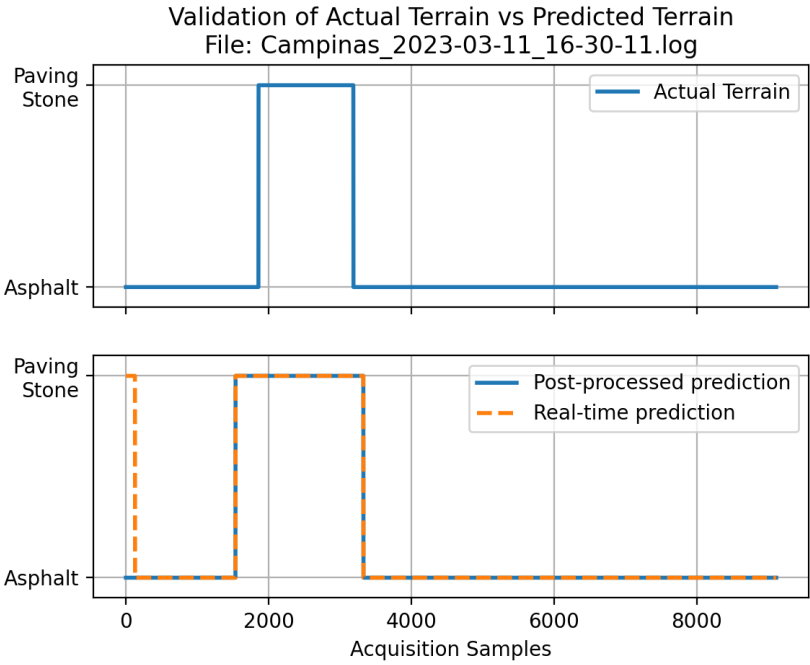


Figure 47 – Comparison of real-time and computer analyzed prediction output.

5 Conclusions

This work proposed an innovative approach based on embedded artificial intelligence in assessing road quality using accelerometer data.

A device was built using accelerometer and GPS modules, that provide accurate acceleration data and accurate location information. To operate all these peripherals, a real-time operating system kernel was used to allow better management of tasks, memory and control of the entire system, by using well-defined task priorities to manage I2C, UART and data processing in a single processing core.

Over one hour of labeled data was obtained in the city of Campinas (São Paulo), providing the necessary information to be split in a training, validation and test dataset. With this data in hand, a study was executed using multiple metrics to assess which one had the best potential to provide useful training information. This required first a correlation analysis that indicates which metrics are redundant and can be removed from the feature set, reducing the necessary processing power. After correlation, the overlapping coefficient between same metrics on different terrains were analyzed and used to further eliminate redundant metrics. This process reduced the number of metrics from 22 to 5, which are used as training features: peak X and Z axes acceleration, peak X, Y and Z acceleration on the rolling variance subset.

The features were separated in training, validation and testing datasets. Two MLP architectures were proposed, where the first one had one hidden layer and the second one had two hidden layers. The overall result of the training process was very similar in both MLPs, to a point where the trade-off in processing power for accuracy in a two-layer network does not justify its use in an embedded system, where the processing power is very limited and must be efficient. The final model had an average 92% accuracy on the test dataset.

Finally, the selected model was implemented in the embedded system and the real-time prediction was validated in two scenarios, comparing the logged values with post-processed prediction on a computer. The model capabilities on asphalt were found to be a possible indicator of bad pavement, which is the moment when the model predicts a paving stone terrain due to potholes, manhole covers or other asphalt issues. Overall, the results showed that the real-time prediction was consistent with the data available and proves that the system is capable of providing the real-time predictions it was initially intended to.

5.1 Future Work

Future research can be done to improve model performance and also embedded system stability. The firmware can be upgraded to offload data using Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), adding more robustness when compared to UART, since an Ethernet port is available.

The experimental setup can be upgraded to a PCB for better cable management and even add a SD card for local data logging.

The model can be improved with better feature selection by the use of a variable number of samples per metric, which may be defined by distance traveled in a second instead of a fixed number of samples.

5.2 Publications

The following publications are related to this work:

- CAVALCANTI, F.; LOPES, W. T. A.; CARVALHO, F. B. S. de. Real Time Pavement Classification Using an Embedded Neural Network. XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2022).
- CAVALCANTI, F.; LOPES, W. T. A.; CARVALHO, F. B. S. de. Sistema Classificador de Pavimentação Utilizando Acelerômetros e Máquina de Vetores de Suporte. XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2021), 2021. DOI 10.14209/sbrt.2021.1570723968
- CAVALCANTI, F.; LOPES, W. T. A.; CARVALHO, F. B. S. de. Análise de Pavimentação Utilizando Dados de Acelerômetro e Transformada Rápida de Fourier. XI Conferência Nacional em Comunicações, Redes e Segurança da Informação (ENCOM 2021), 2021.

5.3 Software Registration

The software developed in this work has been registered under the National Institute of Industrial Property (INPI) in Brazil.

- **Title:** Sistema Classificador de Pavimentação com Uso de Acelerômetros e Inteligência Artificial.
- **Number:** BR512023001464-8

Bibliography

BECKER, S. Unsupervised learning procedures for neural networks. *International Journal of Neural Systems*, v. 2, p. 17–33, 01 1991. Cited on page 23.

CARLOS, M. R. et al. Evaluation of detection approaches for road anomalies based on accelerometer readings—addressing who’s who. *IEEE Transactions on Intelligent Transportation Systems*, v. 19, n. 10, p. 3334–3343, 2018. Cited on page 14.

CAVALCANTI, F.; LOPES, W. T. A.; CARVALHO, F. B. S. de. Sistema classificador de pavimentação utilizando acelerômetros e máquina de vetores de suporte. *XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2021)*, 2021. Cited on page 15.

CAVALCANTI, F.; LOPES, W. T. A.; CARVALHO, F. B. S. de. Real time pavement classification using an embedded neural network. *XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2022)*, 2022. Cited on page 15.

CLASSIFICATION: True vs. False and Positive vs. Negative. 2022. <https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative>. Cited on page 23.

COMMONS, W. *File:Roc-draft-xkcd-style.svg* — *Wikimedia Commons, the free media repository*. 2021. Accessed: January 14, 2023. Disponível em: <https://commons.wikimedia.org/w/index.php?title=File:Roc-draft-xkcd-style.svg&oldid=588966985>. Cited 2 times on pages 7 and 25.

Departamento da Indústria da Construção. Pavimento de vias no brasil: infraestrutura de transportes terrestres rodoviários e cadeias produtivas da pavimentação. FIESP, 2017. Cited on page 13.

FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters*, v. 27, n. 8, p. 861–874, 2006. ISSN 0167-8655. ROC Analysis in Pattern Recognition. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016786550500303X>. Cited on page 24.

FUMO, D. *A Gentle Introduction To Neural Networks Series — Part 1*. 2017. Disponível em: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>. Cited on page 27.

GERHARD, D. Three degrees of “g’s: How an airbag deployment sensor transformed video games, exercise, and dance. *M/C Journal*, v. 16, n. 6, Nov. 2013. Disponível em: <https://journal.media-culture.org.au/index.php/mcjournal/article/view/742>. Cited on page 17.

GLEN, S. *Correlation Coefficient: Simple Definition, Formula, Easy Steps*. 2021. <https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula>. Cited on page 30.

- GUYON, I.; ELISSEEFF, A. An introduction of variable and feature selection. *Journal of Machine Learning Research Special Issue on Variable and Feature Selection*, v. 3, p. 1157 – 1182, 01 2003. Cited on page 30.
- GÖNEN, M. Analyzing receiver operating characteristic curves with sas. 2007. ISSN ISBN 978-1-59994-298-8. Cited 3 times on pages 7, 24, and 25.
- HARDESTY, L. *Explained: Neural networks*. 2017. Disponível em: <<https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>>. Cited on page 25.
- HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Prentice Hall Inc., 1999. Cited 7 times on pages 7, 23, 25, 26, 27, 28, and 29.
- IBRAHIM, D. *Arm-Based Microcontroller Multitasking Projects*. [S.l.]: Newnes, 2020. ISBN 978-0-12-821227-1. Cited 4 times on pages 20, 21, 22, and 37.
- INMAN, H. F.; JR, E. L. B. The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities. *Communications in Statistics - Theory and Methods*, Taylor & Francis, v. 18, n. 10, p. 3851–3874, 1989. Cited on page 31.
- INVENSENSE. *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2*. [S.l.], 2013. Cited 4 times on pages 7, 9, 33, and 34.
- KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. arXiv, 2014. Disponível em: <<https://arxiv.org/abs/1412.6980>>. Cited on page 51.
- MATIC, A.; OSMANI, V.; MAYORA, O. Speech activity detection using accelerometer. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society*, v. 2012, p. 2112–5, 08 2012. Cited on page 17.
- MEDNIS, A. et al. Real time pothole detection using Android smartphones with accelerometers. In: *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*. [S.l.: s.n.], 2011. p. 1–6. Cited on page 14.
- MEGIER, M. B. et al. Análise comparativa de pavimento asfáltico, pavimento em alvenaria polidédrica e pavimento intertravado em bloco de concreto. *2018: Salão do Conhecimento UNIJUÍ*, 2018. ISSN 2318-2385. Cited on page 13.
- MEOCCI, M.; BRANZI, V.; SANGIOVANNI, A. An innovative approach for high-performance road pavement monitoring using black box. In: *Journal of Civil Structural Health Monitoring (JCSHM)*. [S.l.: s.n.], 2021. v. 11, p. 485–506. Cited on page 14.
- MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997. Cited on page 25.
- NXP SEMICONDUCTORS. *I2C-bus specification and user manual*. [S.l.], 2021. Cited 3 times on pages 7, 19, and 20.
- PARKINSON, B. W. et al. *Global Positioning System: Theory and Applications*. [S.l.]: American Institute of Aeronautics and Astronautics, Inc, 1996. v. 1. Cited 2 times on pages 18 and 19.

- PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Cited on page 50.
- QUADRI, S. H. P. *Micro Sensors*. [S.l.], 2020. 8 p. Cited on page 18.
- RASRAS, M.; ELFADEL, I. A.; DUONG, H. Editorial for the special issue on MEMS accelerometers. *Micromachines*, v. 10, p. 290, 04 2019. Cited on page 18.
- REH, F. J. *Pareto Principle or the 80/20 Rule*. 2019. Disponível em: <<https://www.thebalancecareers.com/pareto-s-principle-the-80-20-rule-2275148>>. Cited on page 50.
- SHALEV-SHWARTZ, S.; BEN-DAVID, S. Understanding machine learning. In: UNIVERSITY OF CAMBRIDGE. [S.l.]: Cambridge University Press, 2014. cap. Introduction. Cited on page 23.
- SPSS Tutorials: Pearson Correlation. 2022.
<https://libguides.library.kent.edu/SPSS/PearsonCorr>. Cited on page 30.
- U-BLOX. *Receiver Description Including Protocol Specification*. 2013. Manual. Cited 2 times on pages 9 and 36.
- U.S. DEPARTMENT OF DEFENSE. *Global Positioning System Standard Positioning Service Performance Standard*. [S.l.], 2008. Cited on page 18.
- VALDEZ, J.; BECKER, J. *Understanding the I2C Bus*. [S.l.], 2015. Cited on page 19.
- WANG, S.; KHUSHABA, R.; KODAGODA, S. Towards speed-independent road-type classification. In: *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*. [S.l.: s.n.], 2012. p. 614–619. Cited on page 14.
- WANG, W.; CAO, L.; QU, F. Driving performance test of plug-in hybrid electric vehicle based on avl-drive. In: *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*. [S.l.: s.n.], 2019. p. 771–775. Cited 2 times on pages 13 and 17.
- WEISS, C.; FROHLICH, H.; ZELL, A. Vibration-based terrain classification using support vector machines. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2006. p. 4429–4434. Cited on page 14.
- YU, L.; LIU, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In: . [S.l.: s.n.], 2003. v. 2, p. 856–863. Cited on page 30.