The University of Nottingham

Faculty of Engineering

Department of Electrical and Electronic Engineering

WellnessAI+: Real-Time ECG-Based Emotion Classification deployed in Resource-Constrained ARM Cortex-M Microprocessor

AUTHOR	:	KOAY XIAN CONG
ID	:	20418760
SUPERVISOR	:	DR HERMAWAN NUGROHO
MODERATOR	:	PROF T. NANDHA KUMAR
DATE	:	29 APRIL 2024

Third year project thesis is submitted in partial fulfilment of the requirements of the degree of **Bachelor of Engineering**

Acknowledgement

I want to express my heartfelt gratitude to everyone who has contributed to the completion of this Final Year Project.

Firstly, I sincerely appreciate my supervisor, Dr Hermawan Nugroho, for his invaluable guidance, support, and motivation throughout this project. I have received a significant amount of knowledge and aid from his insightful advice as I carried out the project.

I am also thankful towards my moderator, Prof T. Nandha Kumar for his invaluable feedback and encouragement in ensuring my project is moving in the right direction.

Subsequently, I would like to extend my gratitude towards the Department of Electrical and Electronics Engineering at the University of Nottingham Malaysia, for providing this hands-on opportunity with the necessary knowledge and skills to undertake this project. The project has been enjoyable yet challenging, with each challenge and difficulty presenting opportunities for growth in both technical and intellectual aspects.

I want to acknowledge the support of my family for their unwavering encouragement and understanding during this academic journey. Their love, encouragement, and belief in my abilities have constantly motivated me.

Special thanks go to my friends and classmates for their continuous support, constructive discussions, and words of encouragement, which have helped me overcome challenges and stay focused on my goals.

Thank you all for being part of this journey and for your contributions to this thesis.

Abstract

The integration of electrocardiogram (ECG) technology with artificial intelligence (AI) holds a promising potential in healthcare monitoring systems, offering insights into both the physical and emotional state of an individual. This thesis presents a proposal for an emotion monitoring system utilizing a portable resource-constrained STM32MP157F-DK2 microcontroller powered by an ARM Cortex-M4 processor and integrated with an ECG sensor. Leveraging the WESAD dataset, the collected data undergoes preprocessing to refine it into smaller time frames and filter out unwanted noise. Different deep neural network (DNN) models are developed, trained, and tested for accuracy, with a focus on compatibility with the FLASH and RAM constraints in hardware, facilitating real-time emotion classification through the ECG sensor.

The project's deliverables include the development of an ECG sensor system capable of acquiring real-time heartbeat signals, data collection, preprocessing, and normalisation, an optimized emotion classification model, and a new design of wearable for comfort, compactness, and user-friendliness. The primary objectives of the project were achieved, with the developed model demonstrating lightweight and fast performance, allowing for real-time inference capabilities. Two distinct model architectures, artificial neural network (ANN) and convolutional neural network (CNN), are compared, with the CNN model exhibiting slightly higher accuracy though with a longer inference time.

The real-time classification model's performance is analysed, with the ANN model achieving an accuracy of 78.74% with an inference time of $6293\mu s$, and the CNN model achieved a slightly higher accuracy of 79.34% with an inference time of $13427\mu s$. Overall, the proposed system showcases the potential for leveraging ECG technology and AI algorithms in real-time emotion monitoring in the healthcare industry.

Table of Contents

List of Figures

List of Tables

1. Introduction

The background of the project, problem statement, proposed solution, aims and objectives, scope and limitations, deliverables, project timeline and thesis outline will be covered in this section and discussed.

1.1 Background

The convergence of electrocardiogram (ECG) and artificial intelligence (AI) technology holds a promising potential in healthcare monitoring systems. This is because ECG-based monitoring systems are able to offer insights into both the physical and emotional state of a human. However, challenges exist specifically within the deployment of AI models in resource-constrained wearables. For instance, the AI model inference and ECG sensor values need to be reliable and accurate.

Research focusing on stress classification has become apparent to be a crucial aspect of affective computing. [1] Prolonged stress significantly impacts humans' overall wellness, and hence there is a need for ongoing and automated stress monitoring systems. [2] While stress is traditionally monitored through electroencephalogram (EEG), recent research has delved into classifying and detecting stress through ECG signals, particularly with the rise of wearable sensors capable of discreetly gathering real-time biosignal data. [3] [4]

Moreover, extensive research has been made in deploying Deep Neural Networks (DNN) on ARM Cortex-A chips, which are designed for high performance and power efficiency in systems like Raspberry Pi and smartphones. However, the deployment of DNN on ARM Cortex-M chips, which is optimized for microcontroller tasks, is still a relatively novel area that requires further exploration and studies. [5] These chips, which are ubiquitous in embedded systems and IoT devices, prioritize real-time performance and consume minimal power. [5]

Hence, an emotion classification and monitoring system is proposed, using a portable STM32MP157F-DK2 powered by an ARM Cortex-M4 processor and integrated with an ECG sensor. Utilizing the WESAD dataset, which captures subjects across various affective states (neutral, stress, amusement and relaxation) during lab study, data preprocessing was applied to refine the dataset into smaller time frames and filter out unwanted noise. Different DNN models are created, trained and tested to determine their performance in terms of model accuracy, model size and inference time. The model trained is also tested to ensure compatibility with resource-constrained hardware. Finally, a comparison is made to compare the performance and accuracy of the conventional DNN with the Convolutional Neural Network (CNN) model.

1.2 Problem Statement

1.2.1 Emotion Classification with ECG

Every individual encounters varying levels of stress in their everyday lives. Stress is the body's natural response to a stimulus that disrupts our physical or emotional balance. [6] [7] At certain times, stress can be advantageous, which is also known as "eustress", as it helps individuals stay alert and prepared to evade hazards. [8] This form of stress can also enhance performance during

work which is beneficial to humans. However, stress turns negative when individuals experience continuous stimulation without intervals of relief or relaxation. [8] This is known as “distress”. Hence, they become overwhelmed, leading to the accumulation of stress-related tension that impacts their lifestyle. These unhealthy amounts of stress contribute to various health issues. [6]

Emotions are complex processors that are made up of multiple components, such as feelings, physiological changes, cognitive responses, behaviour, mental processes etc. [9] Although numerous theoretical frameworks have been proposed to understand how emotions can be classified, there remains no universally accepted model. [9] Several researchers in the past, such as [9] and [10], have studied the correlation between emotion and ECG signals, exploring the potential and limitations of utilizing ECG waveforms for emotion classification.

1.2.2 DNN Deployment in ARM Cortex-M4

Recently, there has been a shift towards utilizing specialized hardware accelerators like Graphics Processing Units (GPU) combined with high-bandwidth memory systems as the preferred architecture to handle significant computational requirements such as training State-Of-The-Art (SOTA) models while maintaining manageable power levels. [5] Therefore, cloud computing platforms like Google Colab, AWS Deep Learning AMIs, and Microsoft MLOps are frequently utilized to enhance the training, development, and testing of deep learning speed.

Issues arise in cloud computing when a substantial volume of computational tasks are carried out in the cloud. For instance, network congestion may occur when there is a high volume of users which leads to undesirable delays in certain real-time situations. [11] Moreover, data security during transmission between edge and cloud devices is also a concern. Consequently, on-device computing emerges as a novel solution that involves conducting AI computations directly at the microcontroller. [5]

Furthermore, there has been a trend for deploying DNN algorithms in microcontrollers with the integration of smart sensors to perform basic tasks such as classifications, recognitions and detections. Unlike mainstream applications such as large language models, these tasks exhibit lower complexity and require extremely low power consumption, where they can operate on batteries for extended periods. [5] Certain applications within this domain can be deployed on cost-effective, low-power microcontrollers, such as those based on ARM Cortex-M core. [5]

1.3 Proposed Solution

To classify emotions, several steps must be undertaken. Initially, hardware setup is required to acquire 16-bit ADC values from AD8232 ECG sensors, and these values need to be transmitted from the STM32MP157F-DK2 to a computer. Subsequently, data collection is conducted where the WESAD dataset that catered for wearable stress and affective detection is utilized in this study. [2] Additionally, preprocessing of the WESAD dataset is performed to align it with the developed model and optimized for memory usage. The dataset is segmented into 1-second intervals at a frequency of $500Hz$, and data noise is filtered and removed. As the data in WESAD is recorded in mV readings, conversion to 16-bit is being done. Following preprocessing, the data is normalised into equal class sizes to prevent bias during model training. Two DNN models, specifically ANN and CNN are proposed and developed. Upon completion of model training, it is converted into a TensorFlow Lite format to accommodate the ARM Cortex-M4 microprocessor,

facilitating real-time emotion inference. Figure 1 shows the simplified overview of the flow of the proposed solution.

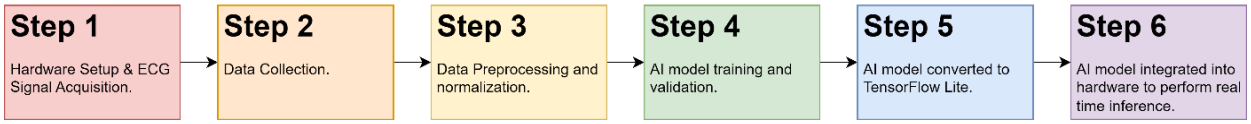


Figure 1. Proposed solution process.

1.4 Aims and Objectives

This project aims to develop a real-time emotion classification model utilizing ECG data that is powered by ARM Cortex-M4. Different model architectures will be tested to see if the model can be fitted into the hardware. The performance of the model is evaluated via various metrics such as accuracy, inference time and model size. The goal is to obtain a model that is small enough with a decent amount of accuracy that can run in real-time.

The main objectives of the project are listed below:

- To integrate the AD8232 ECG sensor with STM32MP157F-DK2 and obtain 16-bit data that can be printed on a computer.
- To collect and preprocess the dataset obtained.
- To train various DNN models that perform emotion classifications.
- To convert the model trained into TensorFlow Lite format so that it is optimized for the hardware.
- To investigate and optimize the performance of DNN models.
- To design a wearable for the real-time emotion classification system.
- To design a real-time dashboard for ECG and model inference visualization.

1.5 Scope and Limitations

The scope of this project includes the development of a real-time emotion classification model using ECG data, facilitated by the ARM Cortex-M4 microprocessor. One of the key focuses of this project is the training and optimization of DNN models for emotion classification. This involves experimenting with different model architectures, training techniques, and optimization strategies to achieve a balance between model accuracy and resource efficiency. The trained model will be converted into TensorFlow Lite format to optimize their compatibility with the hardware. Furthermore, the project includes investigating and optimizing the performance of the DNN models, considering the constraints imposed by the microprocessor's processing power and memory limitations.

In parallel with model development, the "stretch" goals such as the design of a wearable device capable of real-time emotion classification will be also performed. This involves integrating the optimized model and necessary hardware components into a wearable form factor, ensuring comfort, compactness and user-friendliness. Furthermore, the development of a dashboard user interface (UI) will be performed to visualize real-time ECG signals and model inference.

There are several limitations presented in the project. Firstly, hardware constraints imposed by the ARM Cortex-M4 microprocessor may restrict the complexity of the model deployed, impacting its accuracy and real-time performance. Furthermore, the memory limitation in the microcontroller may restrict the deployment of the AI model. The accuracy of the AD8232 sensor may also impact the accuracy of the output. Additionally, data availability and quality, as well as the trade-off between model complexity and accuracy, pose significant challenges. Balancing the complexity of the model with its accuracy while maintaining a small model size is crucial for successful deployment on the microprocessor. Moreover, ensuring the real-time performance of the classification system may require optimization efforts to minimize inference time without compromising accuracy. Lastly, generalization across different individuals and scenarios, as well as mitigating external factors such as motion and environmental noise, are essential considerations for the robustness and reliability of the emotion classification system.

1.6 Deliverables

The project deliverables are:

- ECG sensor system that is capable of acquiring real-time heartbeat signals.
- Data collection, data preprocessing, and data normalisation.
- An optimized emotion classification model that is accurate, fast and lightweight.
- A new design of wearable, ensuring comfort, compactness and user-friendliness.
- Functioning real-time emotion classification inference using data from an ECG sensor.

1.7 Project Timeline

At the beginning of the project, clear objectives and specifications were established, with preliminary literature reviews conducted to discover the feasibility and dataset accessibility of this project. Prior to project initiation, various functions on the STM32MP157F-DK2 were tested to gain familiarity with the firmware. Throughout the autumn semester, efforts were put to focus on configuring the microcontroller to read 16-bit values from the AD8232 ECG sensor. The sensor is connected to the microcontroller, and the values read are transmitted to a computer via UART for monitoring. Additionally, during this period, optional tasks such as the design of wearable devices compatible with the hardware were pursued, capitalizing on available time. Over the study break, attention turned to researching, downloading, preprocessing, and normalizing the WESAD dataset, laying essential groundwork for subsequent model training.

During the start of the spring semester, the project transitioned into active model development. An initial DNN model was created, and the dataset was imported for training purposes. Following successful training for emotion recognition, the model was converted into a TensorFlow Lite format optimized for deployment on the hardware platform. Integration of the optimized model into the STM32MP157F-DK2 was performed, accompanied by rigorous optimization procedures to enhance efficiency and performance. A program that integrates communication between the sensor and the AI model was developed, enabling real-time emotion classification through analysis of ECG signals.

As the project progressed, fine-tuning of hyperparameters and model layers became significant to optimize accuracy and minimize model size. Various experiments were performed to determine the optimal DNN algorithm and architecture. CNN layers were incorporated into the DNN

architecture to serve as feature extraction layers, providing additional insights and comparison metrics. Through systematic refinement and experimentation, the project aimed to deliver a robust, real-time emotion classification system capable of operating seamlessly within the constraints of the microcontroller powered by ARM Cortex-M4.

1.8 Thesis Outline

In Chapter 2, the literature review delves into a comprehensive analysis of research articles and studies about ECG signals, specifically how ECG signals can be interpreted as well as the effect of emotions towards heartbeat. The differences between the ANN and CNN architecture will also be covered, followed by a detailed breakdown of what is in the WESAD dataset. Exploration of previous AI and machine learning projects to classify emotion using ECG data will be performed, as well as investigate projects that utilized ARM Cortex-M processors to deploy DNN in the past. A comparison between different ARM processors will be made.

Chapter 3, the methodology section, outlines the approach adopted in this research. Beginning with a detailed setup of the hardware and software, the methodology proceeds to discuss the configuration of firmware essential for data acquisition and transmission. Subsequent sections include the preprocessing of acquired datasets, model training procedures, and the subsequent evaluation and conversion of trained models for optimized deployment. Furthermore, the methodology discusses the steps involved in deploying the trained model for real-time inference on the ARM Cortex-M processor.

Chapter 4 serves as the platform for presenting and discussing the outcomes of the research efforts. Various experiments performed will be discussed in this section. Detailed results about the performance of the developed model, as well as the efficacy of the ECG sensor in capturing relevant signals will be documented. Performance metrics such as model accuracy, size, and speed will be thoroughly analyzed and compared, with particular emphasis on contrasting the ANN and CNN architectures.

In Chapter 5, the conclusion discusses the findings and the implications of this project. Reflections on the adherence to the project timeline and the objectives set will be discussed. Additionally, this chapter will propose suggestions for future research plans aimed at enhancing the efficacy and scope of the current project.

2. Literature Review

2.1 Emotion Classification using ECG Signals

2.1.1 Interpretation of ECG Signals

An ECG serves as a fundamental medical tool for assessing the heart's electrical activity over time. This diagnostic instrument plays a crucial role in various medical applications, including the diagnosis of cardiac conditions and the monitoring of heart health. A sample signal of the ECG wave is illustrated in Figure 2.

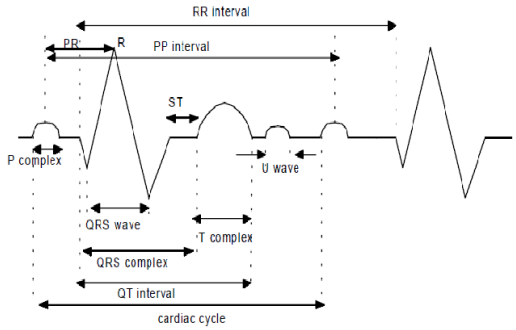


Figure 2. ECG Cycle.

The P-wave corresponds to the atrial contraction, facilitating blood transfer into the ventricles, while the QRS complex signifies ventricular depolarization, initiating the ejection of blood into the systemic circulation. [12] The inter-beat interval (IBI), calculated from the temporal distance between successive R peaks in the ECG trace, serves as a fundamental metric for determining heart rate. [13] Additionally, the T-wave denotes ventricular repolarization, signifying the heart's preparatory phase for subsequent contraction. [12]

2.1.2 Effect of Emotions towards Heartbeat

According to [14], the Autonomic Nervous System (ANS) plays a pivotal role in regulating various involuntary body functions, including heart rate, blood pressure and respiratory rate. Specifically, the paired cardiovascular control centres within the medulla oblongata of the nervous system – namely, the Sympathetic Nervous System (SNS) and the Parasympathetic Nervous System (PNS). SNS increases heart rate and force of contraction while PNS decreases the heart rate. While at rest, both centres exert subtle stimulation on the heart, collectively contributing to the autonomic tone. [15] The cardio accelerator triggers the release of the neurotransmitter norepinephrine, significantly elevating heart rate, through the activation of SNS and the sinoatrial (SA) node, whereas the cardioinhibitory mechanism releases acetylcholine (Ach) neurotransmitter within the PNS. [14] [16]

In response to situational cues and alterations of emotional experiences, the ANS dynamically adjust its activity levels. [15] For instance, a “flight-or-fight” situation [17] triggers the activation of the cardio accelerator, preparing the body for danger. Consequently, the interplay between the heart and brain during emotional encounters can induce distinguishable changes or variations in how emotions are perceived or expressed. [18] Thus, emotional experiences are not solely influenced by cognitive processes and subjective feelings but are also intertwined with the physiological interplay between the heart and brain. Consequently, leveraging ECG signals for emotion classification holds promise, as it offers a unique avenue to decipher emotional states based on the physiological responses of the cardiovascular system. [18]

2.1.3 Artificial Neural Network

An Artificial Neural Network mimics the information processing in the human brain. [19] For instance, it is made up of hidden layers between input and output layers which are crucial components that enable the network to learn complex data representations as illustrated in Figure 3. The development of the initial neural networks can be traced to as early as the 1940s which was performed by Warren McCulloch and Walter Pitts. [19] Since then, neural networks have evolved until the development of DNN with applications in various fields and industries.

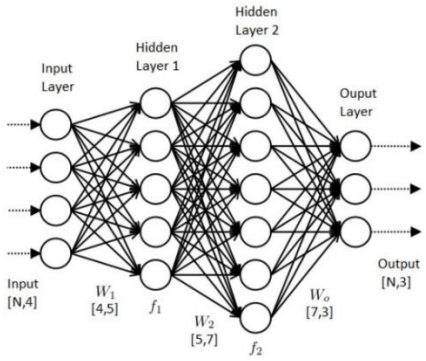


Figure 3. Architecture of Artificial Neural Network.

Inside each hidden layer, it consists of a different amount of nodes which are also called neurons where it receives inputs from previous layers applies weights and adds a bias term which then passes through an activation function. [19] Activation functions such as sigmoid, tanh, and ReLU add non-linearity to the network to enhance the learning capabilities.

2.1.4 Convolutional Neural Network

A CNN is a variant of the feed-forward neural network, commonly utilized in object detection, classifications, image recognition etc. CNNs consist of layers such as input, convolution, pooling, and fully connected layers. [20] The convolutional layers apply filters to extract different characteristics from input data, which are then passed through subsequent layers for further processing. [20] The architecture of CNN is illustrated in Figure 4.

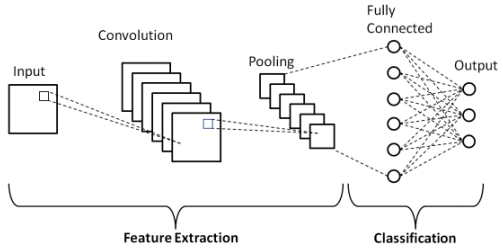


Figure 4. Architecture of Convolutional Neural Network.

In this study, 1D-CNN will be used. 1D-CNN operates on one-dimensional data, like sequences or time-series data, where the kernel moves in one direction. [21] Since ECG signals are based on sequences from the previous time frame, 1D-CNN is suitable to be used. On the contrary, 2D-CNN works on two-dimensional data, such as images, where the kernel moves in two directions. The main distinction lies in the dimensionality of the input data and the direction of convolutional operations. [21]

2.1.5 WESAD Dataset

The WESAD dataset emerges as a significant advancement within the affect recognition and stress monitoring field, addressing a need for standardized datasets tailored to wearable stress detection. [2] This dataset stands out for its provision of high-quality multimodal data, including physiological and motion data collected from both wrist-worn Empatica E4 and chest-worn RespiBan devices. [1] It incorporates various sensor modalities, including blood volume pulse (BVP), electrocardiogram (ECG), respiration (RSP), body temperature (TEMP) and three-axis acceleration (ACC) from 15 participants under different conditions, thus offering a comprehensive

platform for research in affect detection. [2] Notably, WESAD categorizes affective states into four classes - neutral, stress amusement, and meditation. [2] The specifications for the different conditions are outlined as shown:

1. Neutral: This phase spans 20 minutes and aims to record the participant's neutral emotion. Participants are instructed to remain seated or standing at a table with neutral reading material. [22]
2. Amusement: Participants view a series of 11 funny video clips. Each clip is separated by a brief neutral interval of 5 seconds. The total duration for this phase is 392 seconds. [22]
3. Stress: Participants undergo the Trier Social Stress Test (TSST), which involves delivering a 5-minute speech about their strengths and weaknesses in front of a panel comprising three human resource specialists. Additionally, participants are tasked with counting down from 2023 in decrements of 17, with instructions to restart if any errors occur. The total duration of this phase is approximately 10 minutes. [22]
4. Meditation: The meditation session involved a controlled breathing exercise guided by an audio track. Participants followed the instructions with their eyes closed, and seated in a comfortable position. The meditation lasted for 7 minutes. [22]

In studies evaluating the performance of the WESAD dataset, classification accuracies of up to 80% for the four-class classification (neutral, stress, amusement, meditaton) and up to 93% for the binary classification (stress, non-stress) have been reported. [2] [23] These findings underscore the dataset's efficacy in facilitating accurate stress and affect detection through machine learning approaches. However, it is important to acknowledge that there is an unequal distribution of stress and non-stress samples in the WESAD dataset, which causes bias in the modal developed. [22]

The WESAD dataset categorizes states through binary labels, indicating whether subjects are experiencing stress or not, derived from self-reported annotations collected during data acquisition. Subjects are prompted to report perceived stress or relaxation levels while engaging in various tasks or activities. [23] There are several questionnaires used to classify the labels which are Positive and Negative Affect Schedule (PANAS), State-Trait Anxiety Inventory (STAI), Self-Assessment Manikin (SAM), and Short Stress State Questionnaire (SSSQ) and the questions are shown in Figure 5.

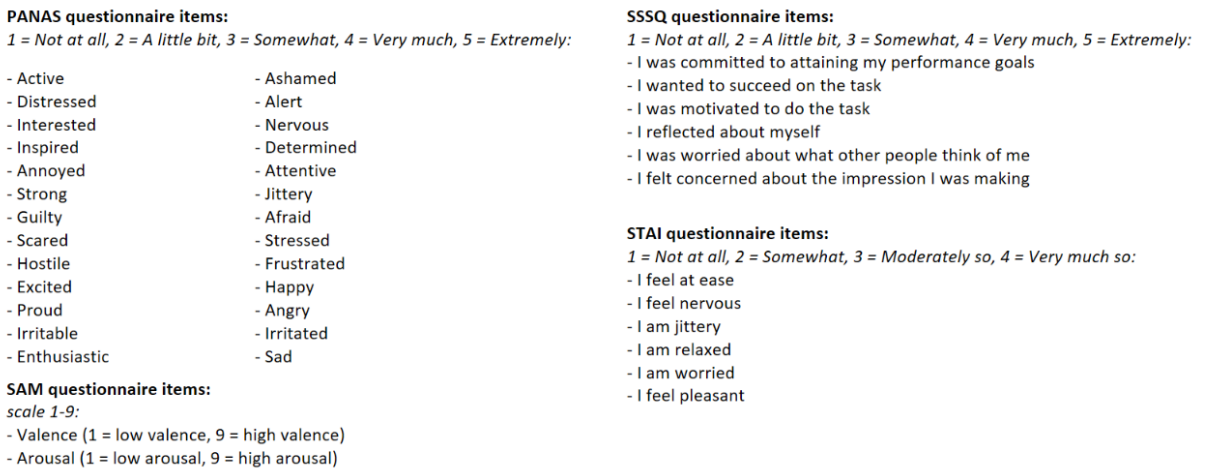


Figure 5. WESAD Questionnaires.

Given the aim of deploying a DNN into a wearable context, the WESAD dataset emerges as an ideal resource for this study. However, it's essential to consider the broader implications and potential limitations of relying solely on the WESAD dataset, including its representativeness across diverse populations and scenarios, as well as its applicability to real-world deployment scenarios. [23] Furthermore, exploring recent advancements in wearable sensor technology as well as different datasets and their implications for stress and affect detection systems could provide valuable insights into future research directions. [22] However, considering limitations in time and resources, these tasks could be planned for future improvement.

2.1.6 Related works of training emotion classification using ECG data

Thorough and extensive research and data collection have been undertaken for this paper, encompassing studies spanning from 2011 to 2023. Table 1 displays various emotion classification models that were trained with ECG data with the performance of the models. Various techniques exist for categorizing emotions:

1. Discrete Emotional Models (DEM) aim to identify and categorize specific emotional states such as happiness, sadness, anger, and others.
2. Affective Dimensional Models (ADM) segment emotions based on valence or arousal dimensions.
3. Stress vs. Non-stress Classification Model distinguishes between emotions associated with stress and those that are not.

It can be observed that the studies that only include ECG modalities can achieve 70% accuracy and above while studies that combine different modalities are not that accurate. Most of the studies carried out focus on the accuracy of the model and hence complex classification methods are applied. Several studies such as [24] [25] even proposed combining several datasets that are publicly available to perform classifications. Prior to the year 2020, conventional machine learning methods were utilised as illustrated in Table 1 while studies performed latterly implemented various deep learning methods.

Table 1. List of similar ECG-based emotion classification studies.

Year	Dataset	Modalities	Classifications	Model Type	Maximum Accuracy (%)	Ref
2023	CASE: 30 subjects	ECG	Valence, Arousal	1D CNN	73.8	[26]
2023	WESAD, CASE, APD	ECG, EDA	Stress, Arousal	SVM, LightGBM, RF, XGBoost	68.3	[27]
2022	PhysioNet: 9 subjects	ECG, GSR, EMG, RSP	Stress	Transfer Learning and Fuzzy Logic	98.11	[28]
2022	MIT-BIH: 47 subjects	ECG	Stress	SVM	91	[29]
2021	WESAD	ECG	Stress	CNN, LSTM	98.3	[30]
2020	Own: 20 subjects	ECG	Happy, Sad, Pleasant, Angry	KNN	92	[31]
2020	AMIGOS, DREAMER, WESAD, SWELL	ECG	Valence, Arousal, Stress	Self-Supervised CNN	Depends on dataset	[24]
2019	Own: 25 subjects	ECG	Anger, Sadness, Joy, Pleasure	Extra Tree	80	[32]
2019	Own: 16 subjects	ECG, EEG, RSP	Valence, Arousal	SVM	96.875 Valence 88.5417 Arousal	[33]
2019	DECAF: 30 subjects	ECG, MEG	Valence, Arousal	RF, SVM	63.4 RF 64.5 SVM	[34]
2019	AMIGOS, DEAP, DREAMER, MANHOB-HCI: 120 subjects	ECG, EEG, GSR, EDA, RSP, SKT, etc.	Emotion, Valence, Arousal, Liking	Transfer Learning	Combined Dataset: 58.57 Valence 61.84 Arousal	[35]
2018	AMIGOS: 40 subjects	ECG, EEG, GSR	Valence, Arousal, Basic Emotions	LinearSVM	54.5 Valence 55.1 Arousal	[36]
2018	DREAMER: 23 subjects	ECG, EEG	Valence, Arousal	SVM	62.37	[37]
2018	WESAD: 15 subjects	BVP, ECG, EDA, EMG, RSP, TEMP	Neutral, Stress, Amusement	DT, RF, NB, LDA, KNN	3 Class: 80 2 Class: 93	[2]
2017	Own: 11 subjects	GSR, ECG	Valence, Arousal	PNN	100	[38]
2016	ASCERTAIN: 58 subjects	EEG, ECG, GSR, facial activity	Valence, Arousal	Linear SVM, NB	56.5 SVM 59.5 NB	[39]
2014	Own: 25 subjects	Forehead Biosignals, ECG	Valence, Arousal	SVM	88.78	[40]
2011	Own: 44 subjects	ECG	Valence, Arousal	LDA	89	[41]

2.2 Deploying models into ARM Cortex-M

2.2.1 Comparison between ARM Cortex-A, Cortex-R and Cortex-M

The ARM Cortex-A series comprises application processor cores designed for performance-intensive systems, offering solutions tailored for devices operating on operating systems like Linux or Android. These cores find applications across a diverse spectrum, ranging from low-cost handsets to sophisticated devices such as smartphones, tablet computers, and enterprise networking equipment. [42]

In contrast, the ARM Cortex-R series caters to high-performance real-time applications, although it remains relatively lesser-known. The core targets domains like hard disk controllers and networking equipment. [42]

The ARM Cortex-M series serves as microcontroller cores suited for a broad array of embedded applications. While these cores can be implemented as soft cores in FPGA setups, they are predominantly found in microcontroller units (MCUs) with built-in memories, clocks, and peripherals. Variants within the Cortex-M series are tailored to meet specific market demands, with some emphasizing energy efficiency, others prioritizing high performance, and some targeting specialized market segments. [42] Based on [43], STM32 MCUs that utilize ARM Cortex-M processors are commonly utilized for existing machine learning research.

The ARM Cortex-M4, utilized in this study, is a 32-bit RISC core that can operate at speeds up to 209MHz. Notably, it includes a floating-point unit (FPU) supporting single-precision data processing instructions and data types. Furthermore, it offers comprehensive digital signal processing (DSP) instructions and includes a memory protection unit (MPU) to enhance application security. [44] Table 2 shows the assessment of performance and energy efficiency across different ARM Cortex variants based on studies performed by [45].

Table 2. ARM Cortex Series Comparison. [45]

Features	Cortex-A	Cortex-R	Cortex-M
Instruction Set Architecture	ARM	ARM	Thumb
Instruction bits	32	32	32
FPU	Yes	Yes	Optional
DSP Instruction	Yes	Yes	Yes (M4)
Dynamic Power	$80\mu W/MHz$	$120\mu W/MHz$	$8\mu W/MHz$
Application	High Performance	Real-Time	Embedded

It would be beneficial to explore the implications of utilizing ARM Cortex-M processors in deep learning applications beyond the existing research. Furthermore, investigating the potential challenges and opportunities associated with deploying deep learning models on microcontroller platforms, such as ARM Cortex-M4, could provide valuable insights for future research endeavours.

2.2.2 Related works of deploying DNN into ARM Cortex-M

Table 3 shows various studies and applications that were deployed in ARM Cortex-M. Most of the projects utilize ARM Cortex-M4 with X-CUBE-AI and TensorFlow Lite frameworks.

Table 3. Related Works for Deep Learning Inference with ARM Cortex-M.

Year	Model	Application	Tools, Support Frameworks	Hardware	Results	Ref
2024	FCN, CNN	Review and Comparison between AIfES, CNN and FCNN	TensorFlow Lite, CMSIS, AIfES	nRF52840 DK ARM Cortex M4	Memory Consumption Reduction: 54%	[46]
2023	CNN	Damage Classification for Concrete Materials	Tiny ML, TensorFlow Lite	nRF52840 ARM Cortex M4	Accuracy: 99.6% Inference Time: 166.822 ms Power Consumption: 0.555 mJ	[47]
2023	CNN	Effects of Neutron Radiation	CMSIS-NN	STM32-L476RGT6U board ARM Cortex-M4 STM32-F767ZIT6 board ARM Cortex-M7	Silent Data Corruption (SDC) failure reduction: 83% Runtime Overhead: 32%	[48]
2023	CNN	Eye Gaze Estimation for automotive	Transfer Learning, X-CUBE-AI	STM32H747I-DISCO ARM Cortex-M7	Inference Time: 870.5ms Accuracy: 73.66%	[49]
2022	CNN	Real-Time Arrhythmia Classification	TensorFlow Lite, FlatBuffer, CMSIS	nRF52840 ARM Cortex-M4	Accuracy: 97.7% Inference Time: 298ms Current Consumption: 3.55mA	[50]
2021	CNN	Arm Movements Recognition	TensorFlow Lite	Arduino Nano 33 BLE ARM Cortex-M4F	Accuracy: 98.9% Inference Time: 132-135ms	[51]
2020	LSTM, GRU,CNN-LSTM,CNN-GRU	Environmental Prediction	X-CUBE-AI , Keras, TensorFlow	STM32F401RET6 ARM Cortex-M4	NRMSE: 0.0328 NMAE: 0.0251	[52]
2020	C-RNN	Cardiac Arrhythmia Detection	Keras, CMSIS-NN	nRF52832 ARM Cortex-M4	Accuracy: 85.7% Inference Time:94.8 ms Power Consumption:20.65 mW	[53]
2020	CNN	Drowsiness Detection based on Eye Blink Sensing	X-CUBE-AI	STM32L451xx ARM Cortex-M4	Accuracy: 87.4–90.8% Power Consumption: 3.5–5 mW	[54]
2020	Q-CNN	Detection of Coffee Plant Diseases	X-CUBE-AI, TensorFlow Lite	STM32F746NG ARM Cortex-M7	Accuracy: 96% Power Consumption:134.12 mJ	[55]
2019	CNN	Monitoring the load applied to a powertrain system	X-CUBE-AI, Keras	STM32F469AI ARM Cortex-M4	Accuracy: 97.71%	[56]
2018	LSTM, RNN	Fall detection wearable system	TensorFlow, CMSIS	STM32L476JGY ARM Cortex-M4	Accuracy: 98%	[57]

2.3 Wearables Design

2.3.1 ECG Electrodes Placements

The common placement of ECG electrodes is shown in Figure 6. [58]

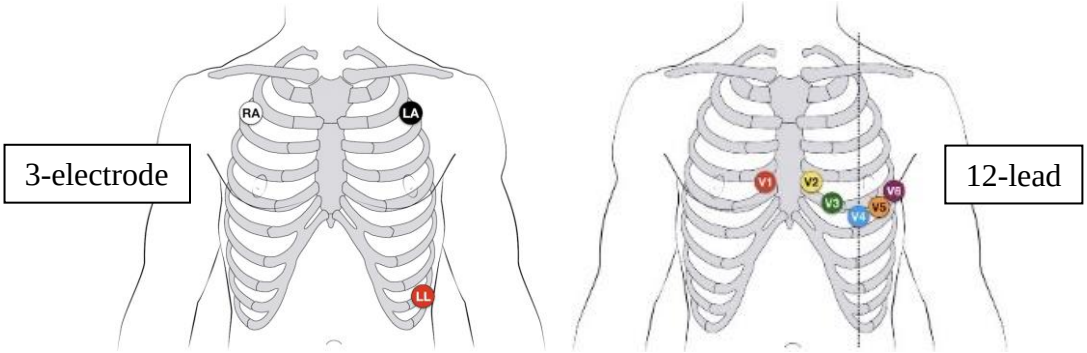


Figure 6. Placement of ECG Electrodes. (3 Electrodes vs 12 Leads)

The primary distinction between 3-electrode and 12-lead ECG sensors lies in their respective capabilities. The 3-electrode sensor offers insights into heart rate and rhythm, enabling the detection of arrhythmias. In contrast, the 12-lead sensor provides additional details including the electrical axis and ischemic regions across various segments of the heart. [59]

In this project, a 3-electrode sensor will be used. In a 3-electrode setup, typically, sensors can be positioned either on the forearms and legs or the chest near the arms and above the right lower abdomen, as illustrated in Figure 7. [60] After establishing the wiring connections, it's essential to cleanse the skin area with an alcohol swab to eliminate any sweat or dirt, ensuring the secure attachment of the electrodes to the skin and accurate data collection.

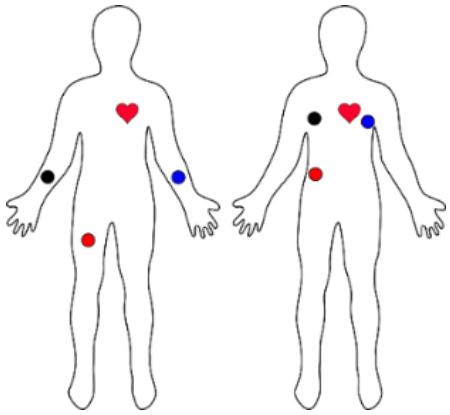


Figure 7. ECG electrodes placements.[60]

2.3.2 Related works of ECG wearables design

[61] evaluated a wireless and wearable electronic-textile EASI-based ECG. It demonstrated the effectiveness of textile electrodes in an EASI configuration for monitoring cardiac activities. The e-textile ECG monitor showed promising results with good signal quality comparable to traditional Holter monitors. This design as illustrated in Figure 8 offers a practical and comfortable solution for continuous and reliable cardiac monitoring during various activities, making it a viable option for patients requiring cardiac rehabilitation or continuous monitoring.

Figure 8. E-Textile ECG Vest with Textile Electrodes and Embedded Wiring. [61]

SensEcho is a vest designed to record single-lead ECG signals, chest and abdominal respiratory signals via respiratory inductive plethysmography (RIP), and triaxial acceleration signals. Its battery allows uninterrupted monitoring for more than 24 hours.[62]

Figure 9. SensEcho ECG Monitoring Vest Design. [62]

Additionally, there exists a wireless, wearable, and flexible ECG sensor inspired by Kirigami, developed by Kinuharu Takei and fellow researchers at Osaka Prefecture University. [63] This sensor, illustrated in Figure 10, allows for the monitoring of ECG data via a mobile phone application, which receives the data wirelessly.

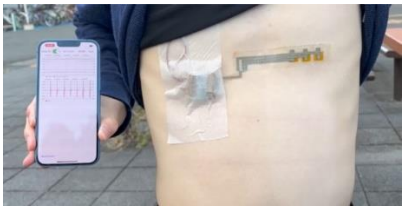


Figure 10. Wireless ECG sensor inspired by Kirigami. [63]

[64] suggested a wearable configuration utilizing Einthoven's triangle, shown in Figure 11. To power the device, two Li-ion batteries and a power management integrated circuit (IC) are utilized. A Bluetooth module facilitates the transmission of ECG data to smart devices.

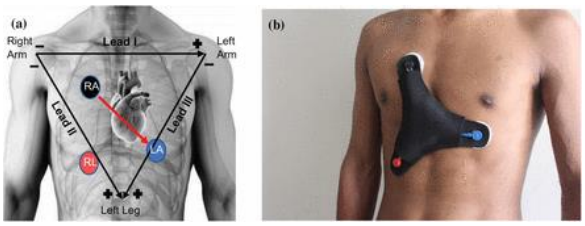


Figure 11. Wireless ECG Wearable based on Einthoven's triangle. [64]

3. Methodology

3.1 Overview of Methodology

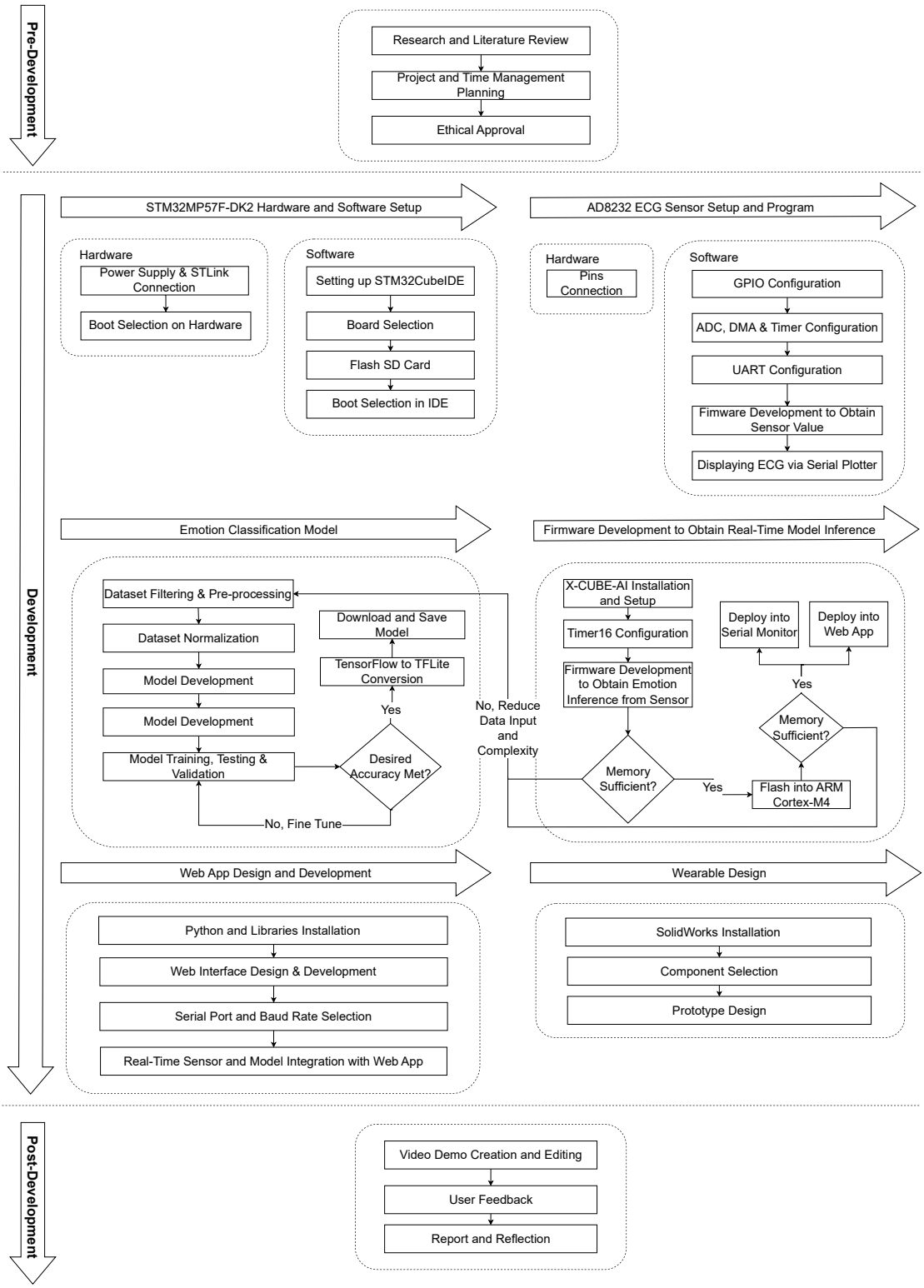


Figure 12. Detailed Overview Methodology.

This section presents a comprehensive overview of the development stages of a real-time ECG-based emotion classification device. During the pre-development stage, efforts were put into researching and learning about the project, planning the project and obtaining ethical approval. During the development stage, 6 primary modules are broken down to be tackled, which are STM32MP157F-DK2 hardware and software setup, AD8232 ECG sensor setup and program, emotion classification model, firmware development to obtain real-time model inference, web app design and development, and wearable design. The details of each module and the methods performed during the development of this project will be discussed in this section. Finally, during the post-development of the project, efforts were put into creating and editing a demo video, obtaining user feedback and writing a report. Figure 12 illustrates the overview of the project.

3.2 STM32MP157F-DK2 and AD8232 ECG Sensor Setup and Connection

3.2.1 Hardware

3.2.1.1 STM32MP1157F-DK2

2 primary connections are needed to be connected on the STM32MP157F-DK2 which is the 5V/3A power supply and STLink/V2-1 that is used to connect the board to the computer as shown in Figure 13. [65]

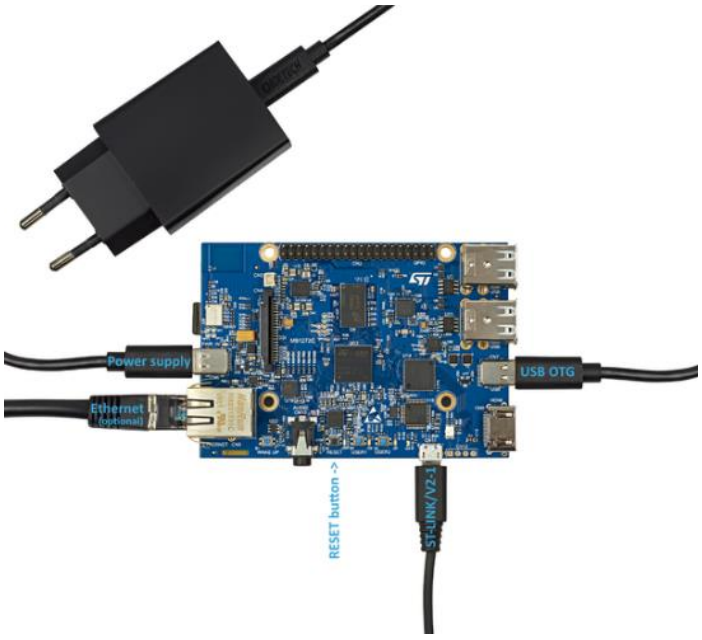


Figure 13. Connection of STM32MP157F-DK2. [65]

Engineering boot is selected through the configuration of buttons as shown in Figure 14 which allows the program to be loaded into ARM Cortex-M4 without booting into the Linux Operating system.

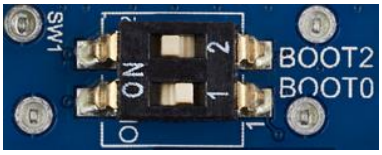


Figure 14. Engineering boot selection.[66]

3.2.1.2 AD8232 ECG Sensor

By referring to the documentation [67], the sensor can be connected to the Arduino UNO microcontroller based on the pinout shown in Figure 15. With the presence of Arduino connectors on the STM32MP157F-DK2, the ECG sensor's corresponding pins were connected to the board pins that match the configurations of Arduino on the STM32.

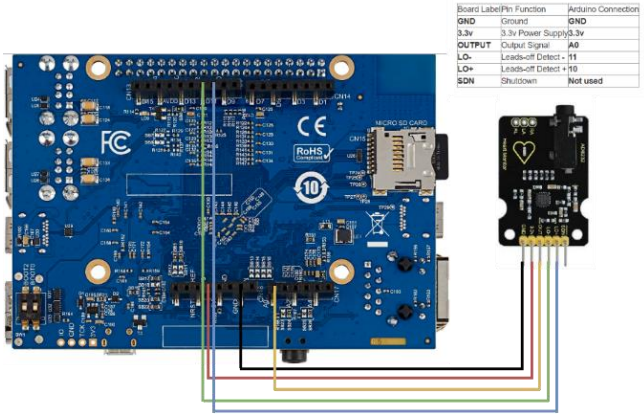


Figure 15. ECG sensor connections.

3 electrode pads which are labelled as L, R and COM can be connected to the AD8232 ECG monitor board using a one-off-three-charge wire with a 3.5mm earphone jack. The electrodes are attached to the body using the standard method, as illustrated in Figure 16.

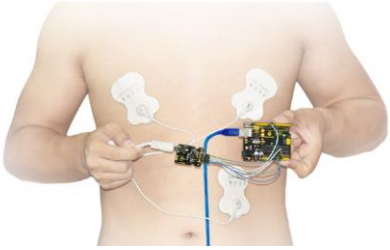


Figure 16. ECG electrodes connection.

3.2.2 Software

3.2.2.1 STM32CubeIDE Setup

STM32CubeIDE version 1.14.1 was installed on Windows 11. A new STM32 project is created and the board is selected as shown in Figure 17.

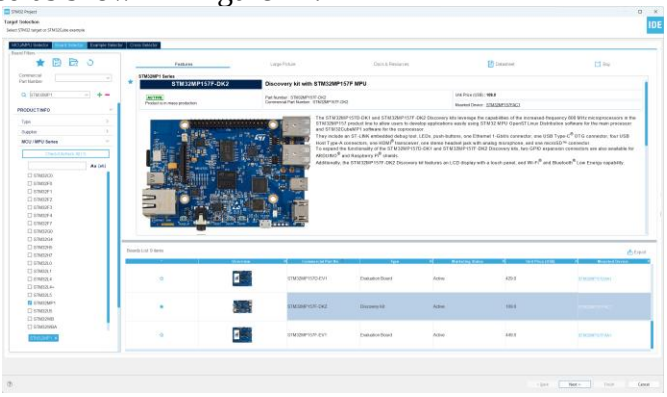


Figure 17. Board Selection.

Once the project is created, engineering mode is configured as shown in Figure 18. Engineering mode disables the Linux boot loads the firmware in RAM and resets the ARM Cortex-M4. The connection setup frequency is configured as 4MHz. By default, the production mode is selected where the program will load into the Linux flashed in the Micro SD card.

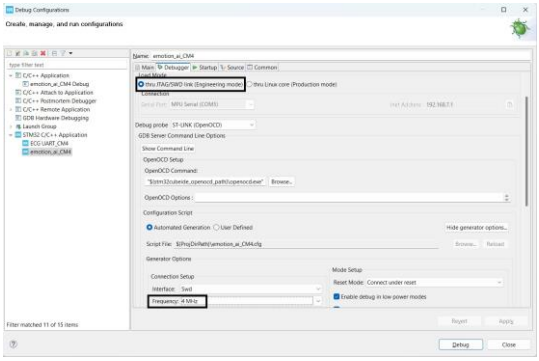


Figure 18. Debug configurations.

3.2.2.2 Configuring GPIO Input Pins

To determine if the sensor has input or not, GPIO input pins can be read from the microcontroller. As shown in Figure 19, LO+ and LO- on the AD8232 ECG sensor module are connected on pins 10 and 11 respectively on the Arduino UNO board. Based on the STM32MP157-DK2 documentation [68], Arduino connector pin 10 on the STM32 board is located at PE11 while Arduino connector pin 11 on the STM32 board is located at PE14. Hence, GPIO inputs are configured to the corresponding pins as shown in Figure 19

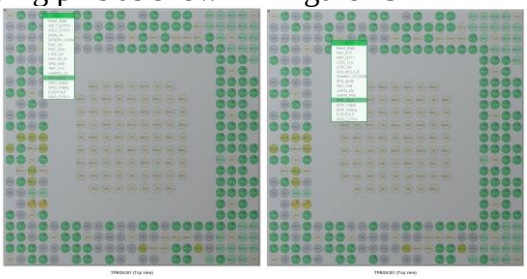


Figure 19. GPIO input pinout configurations.

3.2.2.3 Configuring ADC Pins and Setup DMA

To obtain the sensor ADC readings, OUTPUT pins need to be read. As shown in Figure 15, OUTPUT on the AD8232 ECG sensor module is connected to A0 on the Arduino UNO board. Based on [68], Arduino connector A0 on the STM32 board is located at PF14 and can be configured as ADC2_IN6. Hence, PF14 is configured as GPIO_Analog and ADC2_INP2 as illustrated in Figure 20.

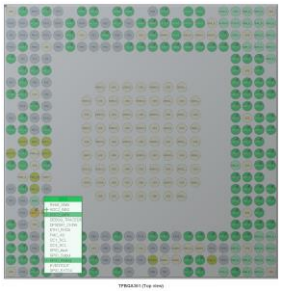


Figure 20. ADC pinout configuration.

ADC2 is configured to M4 and IN6 single-ended is enabled as illustrated in Figure 21. This configuration configures PF14 as Arduino connectors A0 pin. Next, the clock prescaler is set as asynchronous clock mode divided by 256 to slow down the ADC clock. The ADC clock is set as 24MHz as shown in Figure 22. The sampling frequency of ADC can be calculated as shown:

$$sampling\ frequency = \frac{ADC\ clock\ frequency}{t_{CONV} + clock\ prescaler} \quad \textit{Equ 1}$$

$$t_{CONV} = t_s + N\_bit\ resolution/2 \quad \textit{Equ 2}$$

Where $t_s = 1.5$
Hence, the sampling frequency of the ADC is approximately $90kHz$.

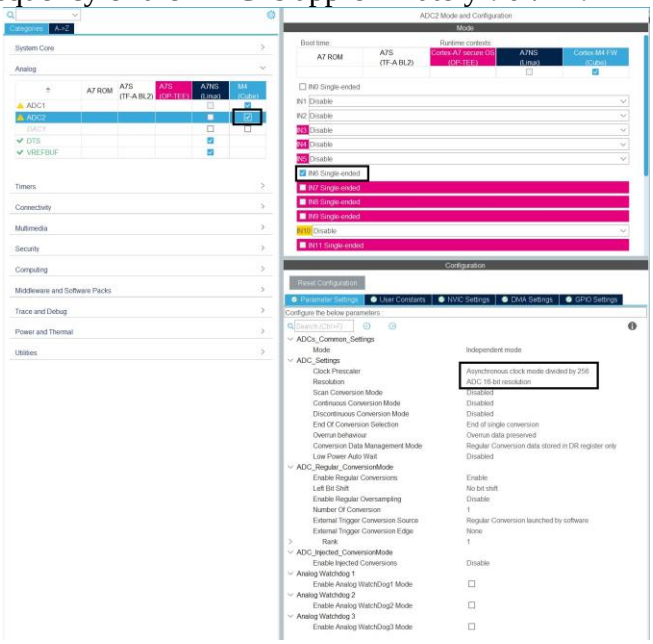


Figure 21. ADC mode and configuration.

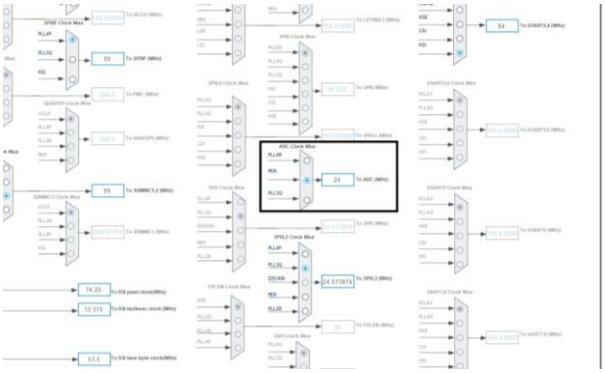


Figure 22. ADC clock mux frequency.

To allow ADC values read from the sensor to be directly stored in a memory buffer, without passing through the CPU, DMA is setup as shown in Figure 23. This allows the data to be sent from one peripheral to another so that the CPU can work on other things while the DMA moves data. To configure the frequency of ADC at $500Hz$, a timer is being set up to trigger the ADC conversions. An interrupt will be triggered when the timer counts to 180 as $\frac{90kHz}{500Hz} = 180$. Hence, the 500Hz ADC data input can be produced.

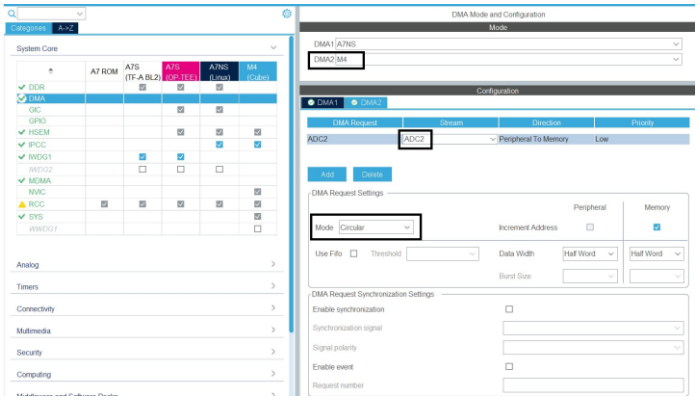


Figure 23. DMA configuration.

3.2.2.4 Configuring UART Pins for STLink

In order to send values read from the ADC to the computer, a serial communication protocol needs to be established. STLink-V2 port on the STM32 board enables data to be sent via the board to the computer through UART protocol. Based on the documentation [68], the STLink UART4 TX is located at pin PG11 while the STLink UART4 RX is located at pin PB2. Hence, UART4 is enabled at the M4 microprocessor and asynchronous modes are selected as shown in Figure 24. The baud rate is selected as 115200Bits/s.

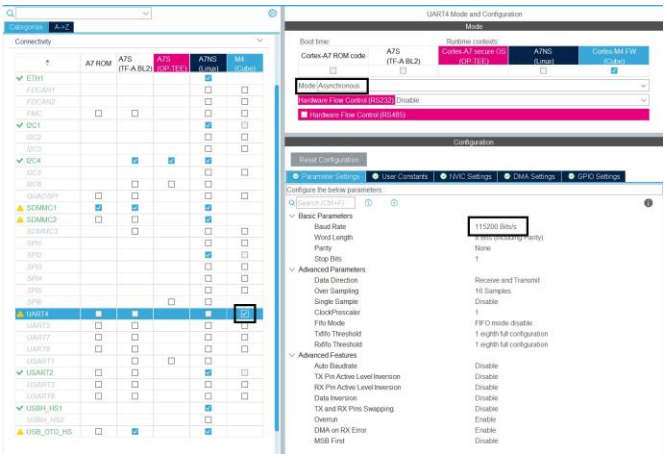


Figure 24. UART configurations.

3.2.2.5 X-CUBE-AI Installation and Setup

Prior to integrating the trained model with ECG sensors into the firmware, it is essential to install and configure X-CUBE-AI. Ensuring that the STM32Cube IDE is updated to the latest version is essential to prevent compiler errors during code execution. For this investigation, STMCubeIDE version 1.14.1 is utilized.

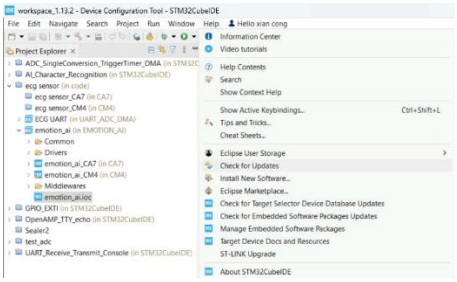


Figure 25. Update to the latest STM32Cube IDE Version.

After the update of STM32CubeIDE is completed, X-CUBE-AI is installed through the software package manager. For this study, X-CUBE-AI version 8.1.0 is being utilized.

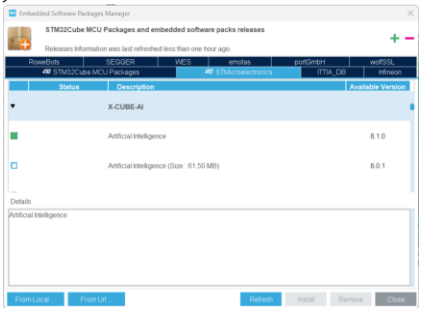


Figure 26. X-CUBE-AI Installation.

After the installation of X-CUBE-AI, various configurations are required to integrate the model into STM32CubeIDE. Initially, M4 is selected and enabled for the X-CUBE-AI software packs. Subsequently, a network is added, and the TF Lite model is specified. A name is assigned to the model, and the TensorFlow Lite model trained is uploaded, as shown in Figure 27. The compression type for the model can be chosen from options including "None", "Lossless", "Low", "Medium", or "High", while optimization options comprise "Balanced", "Time", or "RAM". As real-time sensor readings will be provided to the model, validation inputs and outputs can be disregarded. Following this, the "analyze" button is utilized to assess whether the model possesses adequate Flash Memory and RAM for execution on hardware.

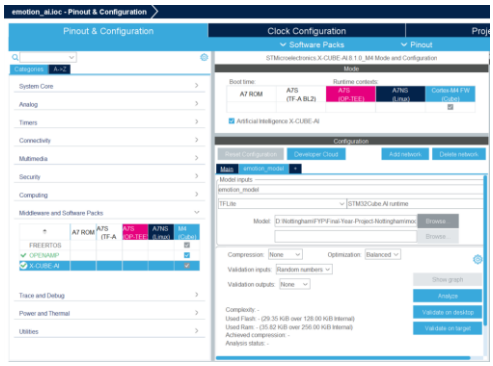


Figure 27. Model Implementation in STM32CubeIDE.

3.2.2.6 Configuring Timer16 for Model Inference Time Calculation

To determine how long it takes for the DNN inference to be carried out on the STM32MP157F-DK2, Timer16 is being set up as illustrated in Figure 28.

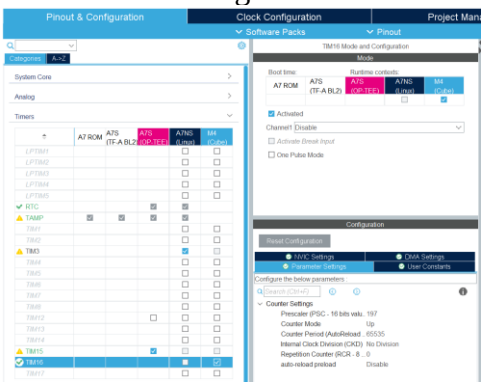


Figure 28. Timer16 Setup Configuration.

3.3 Emotion Classification Model

3.3.1 Data Pre-Processing, Filtering, Segmentation and Normalisation

WESAD dataset [69] is downloaded and several files are presented in the folder. SX.pkl contains synchronized data and labels for each test subject. However, there is a significant amount of data inside SX.pkl and hence dataset preprocessing is carried out to filter only the ECG and labels. ECG from SX.pkl is given in mV values and hence conversion to 16-bit is needed using the following equation:

$$16\_bit\_raw\_sensor\_value = \left(\frac{ECG\_millivolt}{Vcc} + 0.5\right) \times chan\_bit \qquad \textit{Equ 3}$$

Whereby $Vcc = 3V$,

$chan_{bit} = 2^{16}$.

In this study, the ECG signals were initially downsampled to 500Hz and categorized and filtered into two distinct groups based on the given labels which are stress and non-stress (neutral). A Python script is written to preprocess the data using Pickle, Pandas and Numpy libraries.

```python
import pickle
import pandas as pd
import numpy as np

# Load data from the pickle file
with open("data_preprocessing\\pkl_data\\S2.pkl", "rb") as file:
    obj = pickle.load(file, encoding="latin1")

# Extract 'label' and 'ECG' arrays
label_data = obj['label']
ecg_data = obj['signal']['chest']['ECG']

# Create a DataFrame with 'label' and 'ECG' columns
df = pd.DataFrame({'label': label_data.flatten(), 'ECG': ecg_data.flatten()})

# Filter out rows where the label is 1, 5, 6, or 7
filtered_df = df[~df['label'].isin([0, 3, 4, 5, 6, 7])].copy()  # Make a copy to avoid SettingWithCopyWarning

# Create a '16-bit' column based on the modified formula using .loc
filtered_df.loc[:, '16-bit'] = ((filtered_df['ECG'] / 3) + 0.5) * (2 ** 16)

# Reshape 'label' and '16-bit' arrays to have the same number of rows
num_rows = len(filtered_df) // 500 * 500
label_data = filtered_df['label'][:num_rows].values.reshape((-1, 500))
bit16_data = filtered_df['16-bit'][:num_rows].values.reshape((-1, 500))

# Identify rows where labels differ within the 7000 readings and discard those rows
valid_rows_mask = (label_data == label_data[:, [0]]).all(axis=1)
label_data = label_data[valid_rows_mask]
bit16_data = bit16_data[valid_rows_mask]

# Create a DataFrame with transposed '16-bit' data
final_df = pd.DataFrame(np.hstack((bit16_data, label_data[:, 0].reshape(-1, 1))), columns=[f'data{i+1}' for i in range(500)] + ['label'])

# Save the final DataFrame to CSV
final_df.to_csv("data_preprocessing\\output_filter_label\\output_S2_2class_500in.csv", index=False)
```

Figure 29. Dataset preprocessing code.

The data is structured so that each row begins with 500 consecutive heartbeats, followed by a label indicating the type of heartbeat. Subsequently, the next row contains another set of 500 consecutive heartbeats, with the label appearing in column 501. The example can be visualized as shown in Figure 30.

	0	1	2	3	4	5	6	7	8	9	...	491	492	493	494	495	496	497	498	499	Label
0	31603	31609	31615	31639	31612	31627	31607	31611	31590	31582	...	31505	31555	31556	31553	31579	31557	31548	31559	31593	1
1	29577	29521	29507	29500	29451	29506	29611	29743	29835	29893	...	29657	29700	29796	29882	30012	30116	30253	30327	30356	1
2	33605	33690	33833	34036	34193	34291	34311	34324	34369	34449	...	50722	54654	58833	62485	64125	64763	64975	64975	64905	1
3	30831	30933	31065	31159	31171	31228	31350	31455	31468	31495	...	33099	33215	33346	33383	33393	33505	33621	33749	33764	2
4	35477	35637	35669	35513	35317	35189	35051	34863	34669	34511	...	30339	30404	30452	30425	30401	30407	30389	30403	30429	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
38592	31992	31980	31974	31944	31915	31897	31919	31960	31925	31889	...	33804	33849	33883	33907	33987	34021	34033	34001	34021	1
38593	33034	33055	33105	33105	32981	32891	32934	32980	33036	32931	...	33102	33087	33215	33175	33079	33077	33109	33188	33179	2
38594	32492	32553	32541	32523	32593	32699	32740	32721	32647	32623	...	31215	31225	31212	31124	31114	31077	31020	31020	31015	1
38595	32633	32633	32607	32601	32627	32630	32620	32684	32676	32632	...	29790	29809	29847	29914	29987	30034	30070	30111	30195	2
38596	33289	33253	33162	33033	33028	33073	33071	33101	33193	33198	...	27852	27870	27805	27762	27756	27769	27817	27815	27813	2

Figure 30. Preprocessed data tabulated.

The data of all subjects are combined under one CSV file to be used for training. “1” in the dataset represents non-stress while “2” represents stress. After comparing the distribution of data across classes, it is noted that the “non-stress” class is significantly higher than the “stress” class, which is shown in Figure 31.

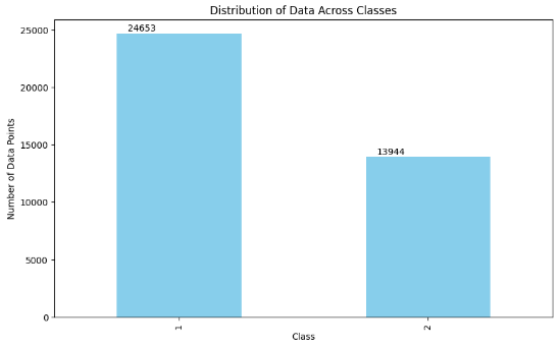


Figure 31. Distribution of Data Across Classes. (Before Normalisation)

Hence, the dataset is further processed to normalise the number of classes so that there is no bias when training the model. The normalised distribution of classes is illustrated in Figure 32. The additional dataset from the “non-stress” class is randomly removed so that both classes have an equal amount of dataset.

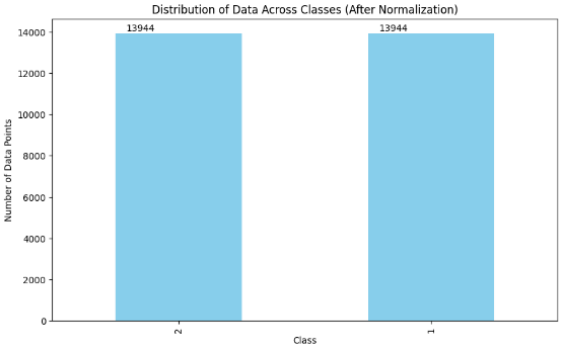


Figure 32. Distribution of Data Across Classes. (After Normalisation)

The sample of preprocessed data is shown in Figure 33.

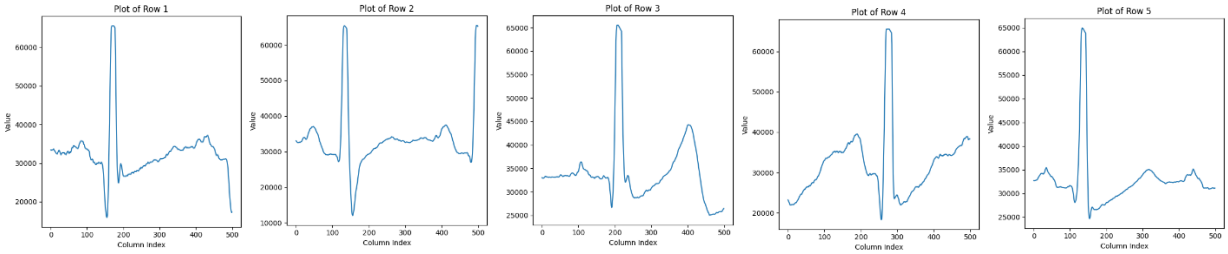


Figure 33. Sample of preprocessed 500Hz data.

Various sampling frequencies (500Hz, 1000Hz, 1500Hz, …, 5000Hz) are tested to ensure the model is accurate and lightweight. Experiments were carried out to determine the limitations of the ARM Cortex-M4.

3.3.2 Model Training and Validation

The preprocessed data were divided into two subsets: 85% for training and 15% for validation. This ensures a significant amount of data is used for training, while a small amount is being validated to monitor the performance of unseen data.

Validation of the model is conducted using accuracy and loss metrics. Throughout training epochs, the validation accuracy is tracked and compared. If an improvement is observed, the model is saved as the best-performing one, and this process iterates until training ends. Ultimately, the model with the highest validation accuracy is saved as the best model, which is then evaluated using test data to determine its accuracy.

Two different models, ANN and CNN are proposed and the performance of model size, inference time and accuracy are compared through experiments which will be discussed in Section 3.3.3. Through iterative experimentation and parameter tuning, this model configuration is optimized to achieve improved performance in stress classification tasks. The primary source guiding the development of the ANN and CNN models is cited in [70] and [71]. Due to limitations in hardware, adjustments and refinements are performed on the original model proposed.

3.3.2.1 ANN

The sequential model architecture comprises five dense layers, progressively reducing the output shape from 32 to 2. The rationale behind the selection of this architecture will be elaborated upon in Section 4.

The key components integrated into the ANN include:

- Batch Normalisation: Applied to each layer to normalise the inputs. This normalisation speeds up the training process and improves the overall stability of the network.
- Dropout Layers: Incorporated within the network to prevent overfitting. These layers randomly deactivate a subset of neurons during each training iteration, which helps the model to generalize better.
- Activation Functions: Leaky Rectified Linear Unit (ReLU) activation functions are used to introduce non-linearity, allowing the model to learn complex patterns. The final layer uses a sigmoid activation function to perform binary classification, effectively distinguishing between stress and non-stress states.
- Optimizer and Loss Function: The Adam optimizer is utilized for its efficient computation of individual adaptive learning rates for different parameters. It minimizes a binary cross-entropy loss function that quantifies the difference between the predicted probabilities and the actual class output, which is ideal for binary classification tasks.
- Batch Size and Iterative Refinement: A batch size of 32 is selected to balance the computational efficiency and the effectiveness of the gradient descent. Through iterative experimentation, the model configuration is continuously refined to optimize accuracy, efficiency, and model size for the task of stress classification.

3.3.2.2 CNN

Next, another sequential model is created and integrates a 1D-CNN alongside three fully connected layers for analysis. The incorporation of 1D-CNN enables the model to capture sequential patterns in the input data, potentially enhancing accuracy compared to conventional fully connected

architectures. The initial layer, a 1D convolutional layer, convolves input data with a kernel to produce feature maps that highlight important features over time.
The key components integrated into the CNN include:

- Max-Pooling Layer: Follows the convolutional layer to reduce the spatial dimensionality of the feature maps, thus reducing the number of parameters and computations in the network.
- Regularization and Non-linearity: Similar to the ANN, batch normalisation and dropout are used for regularization, and Leaky ReLU functions are used to introduce non-linearity.
- Early Stopping: To further combat overfitting, early stopping is implemented to halt training if the validation loss does not improve for 15 consecutive epochs.
- Learning Rate: A learning rate of 0.000008 is chosen to ensure that the model converges to a good solution without overshooting.
- Batch normalisation: Applied to normalise inputs between layers, enhancing training stability and convergence
- Dropout regularization: Mitigate overfitting by randomly deactivating some neurons during training.
- Activation Functions: Leaky ReLU activation functions are utilized to introduce non-linearity and aid in learning complex patterns.

The output of the max-pooling layer is flattened into a one-dimensional vector to be fed into the fully connected layers. Following this, three fully connected layers are sequentially stacked to combine the features extracted by the previous layer. The first dense layer consists of 8 neurons, serving as the initial step in extracting higher-level features from the flattened input. The final dense layer comprises two neurons, serving as the output layer for binary classification tasks. The choice of this architecture will be discussed through the experiment performed in Section 4.3.3.

3.3.3 Experiments

3.3.3.1 Experiment 1 (ANN): Reduction of Hidden Layers and Nodes

Multiple iterations of the ANN model were trained and evaluated with varying numbers of hidden layers and nodes. Initially, there are 8 hidden layers with decreasing numbers of nodes per layer: 1024, 512, 256, 128, 64, 32, 16, and 4. Then, the number of hidden layers is reduced to 7, with node counts of 512, 256, 128, 64, 32, 16, and 4 respectively. This process continues, reducing the number of hidden layers by one each time and removing the layer with the highest number of nodes while the performance and model size are noted. The objective was to find the optimal balance between the model complexity, performance and model size, ensuring compatibility with the hardware's constraints. The design of the experiments is as follows:

Table 4. Experiment 1 Design Specifications.

Design	**Parameters**
Architecture	ANN
Number of Classes	4 (Baseline, Stress, Amusement, Meditation)
Duration of Input Signal	10 seconds
Frequency of Input Signal	7000Hz
Number of Hidden Layers	*Independent Variable*
Number of Hidden Nodes in each Layer	*Independent Variable*

The design of hidden nodes in each layer involves selecting values that are powers of 2, such as 4, 8, 16, 32, 64, 128, etc. This choice is made because it leads to higher computational efficiency, as computing systems are optimized for operations involving powers of 2. Consequently, this results in more efficient memory usage and faster computation times, since certain hardware architectures can handle these sizes more effectively. Furthermore, using powers of 2 can enable optimizations in algorithms and data structures, thereby improving the overall performance of the neural network. [73]

3.3.3.2 Experiment 2 (ANN): Testing the Hardware Limitation using the ANN Model

From the previous experiment, insufficient SRAM and model overfitting are noted which will be discussed in Section 4.3.3.1. These can be tackled with complexity reduction in the number of inputs and outputs of the model. The objective of this experiment is to gauge the impact of varying input parameters on the model's accuracy and its compatibility for compilation and deployment. The binary classification method will be tackled with the specifications as shown:

Table 5. Experiment 2 Design Specifications.

Design	**Parameters**
Architecture	ANN
Number of Classes	2 (Non-Stress, Stress)
Duration of Input Signal	*Independent Variable*
Frequency of Input Signal	$500Hz$
Number of Hidden Layers	5
Number of Hidden Nodes in each Layer	32, 16, 8, 4, 2

The original dataset's baseline label is altered to the Non-Stress label. The foundational architecture utilized is derived from Experiment 1, featuring only 5 hidden layers, as shown in Figure 34. To maintain consistency, the dataset is downsampled to $500Hz$. Consequently, testing with 500 data points corresponds to a duration of 1 second, while utilizing 1000 data points extends the ECG signal to 2 seconds. 10 iterations are carried out, ranging from 500 to 5000 number of input data points. Fixed compression and optimization techniques are implemented using the STM32CubeIDE functionality, with emphasis on monitoring the FLASH memory and RAM utilization.

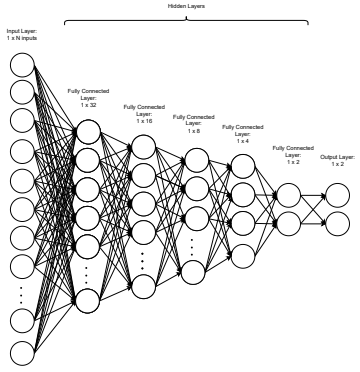


Figure 34. Model Architecture used for Experiment 2.

The model's architecture remains uniform, comprising 5 hidden layers with hidden nodes distributed as 32, 16, 8, 4, and 2. Furthermore, metrics including model accuracy and loss are monitored using Google Colab throughout the training phase.

The test accuracy is determined by selecting the highest validation accuracy achieved during model training, which is subsequently used to validate the model's performance on unseen data. Several hyperparameters as discussed in Section 3.3.2.1 are applied in the model. Ultimately, the model undergoes compilation and deployment onto hardware to identify the maximum capabilities of the hardware.

3.3.3.3 Experiment 3 (CNN): Testing the Hardware Limitation using the CNN model

In this experiment, a 1D-CNN model was developed and optimized for the emotion classification task. The objective was to compare the performance of the CNN model to the ANN model from Experiment 2 when deployed on the ARM Cortex-M4 hardware. Overfitting issues still exist in Experiment 2 results which is shown in Section 4.3.3.2. Hence, the goal of this experiment is to mitigate overfitting issues to obtain maximum performance while deployable in the hardware. The CNN model's accuracy and inference time were evaluated to determine its suitability for the emotion monitoring application. The design specifications of this experiment are as follows:

Table 6. Experiment 3 Design Specifications.

Design	**Parameters**
Architecture	1D-CNN
Number of Classes	2 (Non-Stress, Stress)
Duration of Input Signal	*Independent Variable*
Frequency of Input Signal	$500Hz$
Number of Hidden Layers	*Independent Variable*
Number of Hidden Nodes in each Layer	*Independent Variable*

Similarly, the original dataset's baseline label is altered to the Non-Stress label. The foundational architecture utilized consists of 1D-CNN in the first 1 to 2 layers, followed by 2 to 4 fully connected layers. Hence, the architecture in this experiment is not fixed. Similarly, to maintain consistency, the dataset is downsampled to $500Hz$. Consequently, testing with 500 data points corresponds to a duration of 1 second, while utilizing 1000 data points extends the ECG signal to 2 seconds. Fixed compression and optimization techniques are implemented using the STM32CubeIDE functionality, with a specific emphasis on monitoring the FLASH memory and RAM utilization.

The test accuracy is determined by selecting the highest validation accuracy achieved during model training, which is subsequently utilised to validate the model's performance on unseen data. Several hyperparameters as discussed in Section 3.3.2.2 are applied in the model.

3.3.4 Saving Model and Model Conversion

After saving the best model in .h5 format, TensorFlow Lite format is being converted to enable its deployment on hardware. This conversion process includes size optimization aimed at reducing the model's overall size to enhance efficiency and compatibility with hardware constraints.

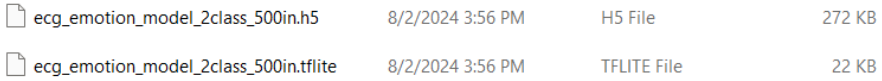


Figure 35. Conversion from TensorFlow to TensorFlow Lite Model.

As shown in Figure 35, the TFLite format is converted from H5 format and there is a significant reduction in file size. The TFLite format is specifically catered for deploying AI models in hardware.

3.4 Firmware Development to Obtain Real-Time Model Inference

3.4.1 Obtaining Real-Time Sensor Value

Once the configuration is set up as shown in Section 3.2.2, code can be generated automatically where the configuration that had been set was created in main.c file. Initially, standard C libraries with `ai_platform.h`, which houses the built-in functions for X-CUBE-AI are imported. Additionally, the library for the emotion recognition model is also imported.

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h> // to use sprintf
#include "math.h"
#include "string.h"

#include "ai_platform.h"
#include "emotion_model.h"
#include "emotion_model_data.h"
```

Figure 36. Import Libraries.

Next, buf array and buf_len variables are created to store the message to be printed onto the serial console on the computer. A timestamp variable is created to determine how long the DNN inference takes place.

```
/* USER CODE BEGIN 1 */
  char buf[50]; // buffer for serial output string
  int buf_len = 0;
  uint32_t timestamp;
```

Figure 37. Message variables declaration.

The main operation of the code works under the while loop as shown in Figure 38. In each interaction, the ADC is called and waits until the ADC conversion is completed. Once the ADC value is obtained, it is stored inside the “raw” variable that had been created. Next, the ADC values are printed in the “msg” array and the array is transmitted via UART4 to the computer and it will keep on looping until the code is terminated.

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_ADC_Start(&hadc2);
      HAL_ADC_PollForConversion(&hadc2, HAL_MAX_DELAY);
      raw = HAL_ADC_GetValue(&hadc2);
      HAL_Delay(1);
      sprintf(msg, "%d\r\n", raw);
      HAL_UART_Transmit(&huart4, msg, strlen(msg), HAL_MAX_DELAY);
    /* USER CODE END WHILE */
```

Figure 38. Main operation while loop.

To be able to get a usable value by reading the ADC, a reference voltage “VREFBUF” needs to be set. “VREFBUF” needs to be enabled after the ADC_Init() but before starting it. [72] Then, calibration for ADC is needed before using it. If the code in Figure 39 is not implemented, the output will keep on printing the maximum number of bits in the ADC.

```
/* USER CODE BEGIN ADC2_Init 2 */
__HAL_RCC_VREF_CLK_ENABLE(); // Enable the VREF clock
  HAL_SYSCFG_VREFBUF_HighImpedanceConfig(SYSCFG_VREFBUF_HIGH_IMPEDANCE_DISABLE);
  HAL_SYSCFG_VREFBUF_VoltageScalingConfig(SYSCFG_VREFBUF_VOLTAGE_SCALE1);
  HAL_SYSCFG_EnableVREFBUF(); // To enable VREFBUF

  if(HAL_ADCEx_Calibration_Start(&hadc2, ADC_CALIB_OFFSET_LINEARITY, ADC_SINGLE_ENDED) != HAL_OK)
  {
    /* Calibration Error */
    Error_Handler();
  }
/* USER CODE END ADC2_Init 2 */
```

Figure 39. ADC setup and calibration.

3.4.2 Integration of Sensor Value with AI Model

Variables are initialized by creating an array for input and output data. In this case, an input array of 500 elements and an output array with 2 elements are created. The emotions array holds labels for the model's output categories, and pointers of ai_input and ai_output are created to manage input and output buffers.

```
/* USER CODE BEGIN PV */
ai_handle emotion_model;
float aiInData[AI_EMOTION_MODEL_IN_1_SIZE];
float aiOutData[AI_EMOTION_MODEL_OUT_1_SIZE];
ai_u8 activations[AI_EMOTION_MODEL_DATA_ACTIVATIONS_SIZE];
const char* emotions[AI_EMOTION_MODEL_OUT_1_SIZE] = {
        "non-stress", "stress"
};

ai_buffer * ai_input;
ai_buffer * ai_output;
```

Figure 40. Model variables declaration.

AI_Init function is run which initializes the AI model, input and output array. A message will be printed once the initialization is successfully performed.

```
/* USER CODE BEGIN 2 */
 AI_Init();
// start timer/counter
// HAL_TIM_Base_Start(&htim16);
buf_len = sprintf(buf, "\r\n\r\nSTM32 X-Cube-AI test\r\n");
HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
/* USER CODE END 2 */
```

Figure 41. AI Initialization.

Subsequently, the code will perform the main operation of the code. Firstly, the ADC will start to obtain the 16-bit ADC values from the ECG sensor. Once the ADC conversion is successful, the data will be stored in the input array until the buffer is filled up. Next, the code will obtain the current timestamp followed by running the AI model inference. The input and output array buffers are passed into the AI_Run function in which model inference will be carried out. The output array buffer will return the confidence of each class which is then printed out in the for loop. A simple validation is performed to verify if the output is at an extreme, which helps identify whether the sensor is disconnected or not. If the sensor is connected, the prediction classes and timestamps will be printed out.

```
/* USER CODE BEGIN WHILE */
while (1)
{
    for (int i = 0; i < AI_EMOTION_MODEL_IN_1_SIZE+1; ++i) {
        HAL_ADC_Start(&hadc2);
        HAL_ADC_PollForConversion(&hadc2, HAL_MAX_DELAY);
        aiInData[i] = HAL_ADC_GetValue(&hadc2);
        HAL_Delay(1);

        if (i == AI_EMOTION_MODEL_IN_1_SIZE) {
            timestamp = htim16.Instance->CNT;
                buf_len = sprintf(buf, "Running inference\r\n");
                HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);

                AI_Run(aiInData, aiOutData);
                /* Output results */
                for (uint32_t i = 0; i < AI_EMOTION_MODEL_OUT_1_SIZE; i++) {
                    buf_len = sprintf(buf, "%8.6f ", aiOutData[i]);
                    HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
                }
              if (aiOutData[0] != 0.0 || aiOutData[1] != 1.0) {
                    uint32_t class = argmax(aiOutData, AI_EMOTION_MODEL_OUT_1_SIZE);
                    buf_len = sprintf(buf, "Prediction : %d - %s\r\n Duration : %lu\r\n", (int)class, emotions[class], htim16.Instance->CNT - timestamp);
                    AL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
                }
              else {
                    buf_len = sprintf(buf, "Sensor Detached\r\n");
                    HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
                }
        }

    }
    // Wait before doing it again
    HAL_Delay(500);
```

Figure 42. Main Operation Loop.

An AI_Init function is created to initialize the AI model. An error message will be prompted if the model fails to load or create.

```
static void AI_Init(void)
{
  ai_error err;

  char buf[50]; // buffer for serial output string
  int buf_len = 0;

  /* Create a local array with the addresses of the activations buffers */
  const ai_handle act_addr[] = { activations };
  /* Create an instance of the model */
  err = ai_emotion_model_create_and_init(&emotion_model, act_addr, NULL);
  if (err.type != AI_ERROR_NONE) {
      buf_len = sprintf(buf, "ai_emotion_model_create error - type=%d code=%d\r\n", err.type, err.code);
      HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
//    printf("ai_emotion_model_create error - type=%d code=%d\r\n", err.type, err.code);
      Error_Handler();
  }
  ai_input = ai_emotion_model_inputs_get(emotion_model, NULL);
  ai_output = ai_emotion_model_outputs_get(emotion_model, NULL);
}
```

Figure 43. AI Init Function.

An AI_Run function is created to perform inference by passing the model, input array which contains the ECG values, and output array which contains the classes into the ai_emotion_model function. An error message will be prompted if the model fails to run.

```
static void AI_Run(float *pIn, float *pOut)
{

  ai_i32 batch;
  ai_error err;

  char buf[50]; // buffer for serial output string
  int buf_len = 0;

  /* Update IO handlers with the data payload */
  ai_input[0].data = AI_HANDLE_PTR(pIn);
  ai_output[0].data = AI_HANDLE_PTR(pOut);

  batch = ai_emotion_model_run(emotion_model, ai_input, ai_output);
  if (batch != 1) {
    err = ai_emotion_model_get_error(emotion_model);
    buf_len = sprintf(buf, "AI ai_emotion_model_run error - type=%d code=%d\r\n", err.type, err.code);
    HAL_UART_Transmit(&huart4, (uint8_t *)buf, buf_len, 100);
    //printf("AI ai_emotion_model_run error - type=%d code=%d\r\n", err.type, err.code);
    Error_Handler();
  }
}
```

Figure 44. AI Run Function.

An argmax function is created to return the index of the highest confidence-scored output.

```
static uint32_t argmax(const float * values, uint32_t len)
{
  float max_value = values[0];
  uint32_t max_index = 0;
  for (uint32_t i = 1; i < len; i++) {
    if (values[i] > max_value) {
      max_value = values[i];
      max_index = i;
    }
  }
  return max_index;
}
```

Figure 45. argmax Function.

3.4.3 Displaying Result via Serial Monitor/Serial Plotter

The software developed is built and compiled onto the hardware. An .elf file will be generated automatically which can be executed by the STM32 board. Once the program is uploaded, a shutdown command will be invoked at the STM32CubeIDE Console and the .elf file will be loaded into the ARM Cortex-M4 microprocessor as shown in Figure 46.

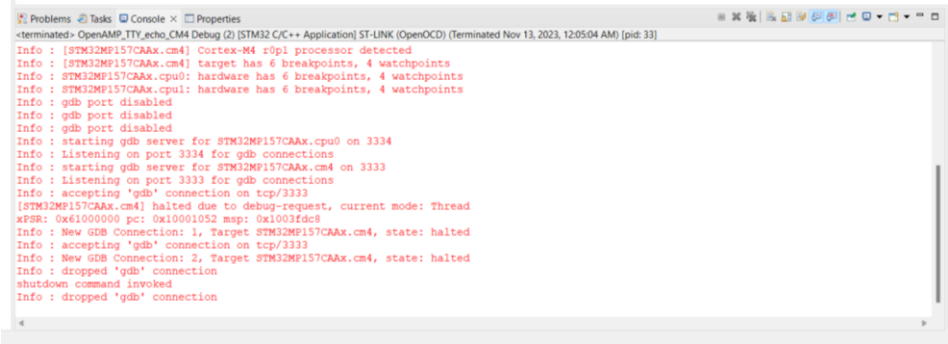


Figure 46. Program uploaded onto ARM Cortex-M4.

To view the sensor values of the computer, several software can be used as long as they can access the serial port on the computer (i.e. Arduino IDE, Putty, HyperTerminal, etc). For simplicity, a serial monitor on Arduino IDE is being used as there are serial plotter features which can display the ECG signals in graph format. Firstly, the port of the STLink needs to be determined by the Device Manager on the computer as shown in Figure 47.

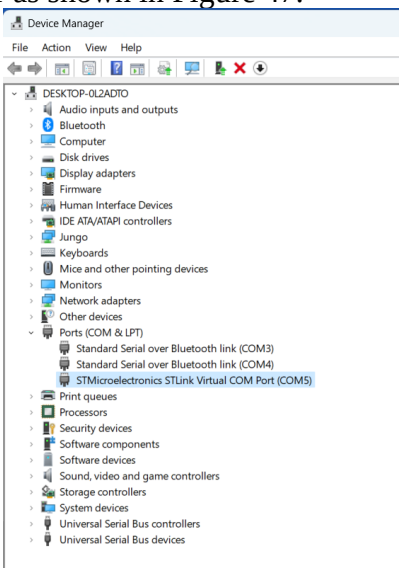


Figure 47. Device manager showing STLink COM port.

Once the STLink COM port is determined, the correct COM is selected on the Arduino IDE and the serial monitor is opened. In the serial monitor, the correct baud rate which is 115200 is selected and the ADC values will be printed out. Additionally, a serial plotter can be used to visualize the ADC values which are equivalent to the ECG signals in 16 bits.

3.5 Web App Design and Development

A dashboard had been developed with Python libraries to visualize the real-time ECG input and model inference output. The application starts by setting up serial communication with an ECG sensor. Users can configure the serial connection directly through the web interface, specifying parameters such as the baud rate and serial port. This flexibility allows the application to be adaptable to different hardware setups. This user-friendly approach ensures that non-technical users can easily configure the necessary settings without delving into the code.

```python
import serial
import streamlit as st
import plotly.graph_objects as go

st.sidebar.title('Settings')

# User input for baud rate and serial port in the sidebar
baud_rate = st.sidebar.number_input('Enter Baud Rate', min_value=9600, max_value=115200, value=115200, step=9600)
serial_port = st.sidebar.selectbox('Select Serial Port', options=['COM5', 'COM1', 'COM2', 'COM3', 'COM4'])

# Open the serial port
ser = serial.Serial(serial_port, baud_rate)

st.title('Real-Time ECG Emotion Classification')

main_container = st.container()

chart = st.plotly_chart(go.Figure(), use_container_width=True)

ecg_values = []  # Define ecg_values as a global variable
prediction_text = st.metric(label="Prediction", value="", delta=None)
confidence_0_text = st.metric(label="Model Confidence 0", value="", delta=None)
confidence_1_text = st.metric(label="Model Confidence 1", value="", delta=None)
duration_text = st.metric(label="Duration (ms)", value="", delta=None)

def parse_serial_data(data):
    global ecg_values
    data = data.decode().strip()
    if data.startswith("Running inference"):
        confidence_line_0 = ser.readline().decode().strip()
        confidence_line_1 = ser.readline().decode().strip()
        prediction_line = ser.readline().decode().strip()
        duration_line = ser.readline().decode().strip()

        confidence_0 = float(confidence_line_0.split(':')[-1].strip())
        confidence_1 = float(confidence_line_1.split(':')[-1].strip())
        prediction_info = prediction_line.split(':')[-1].split('-')
        prediction = int(prediction_info[0].strip()) if prediction_info[0].strip() != "Sensor Detached" else "Sensor Detached"
        duration = float(duration_line.split(':')[-1].strip())

        prediction_text.metric(label="Prediction", value="stress" if prediction == 1 else "non-stress" if prediction == 0 else prediction, delta=None)
        confidence_0_text.metric(label="Model Confidence (Non-stress)", value=confidence_0, delta=None)
        confidence_1_text.metric(label="Model Confidence (Stress)", value=confidence_1, delta=None)
        duration_text.metric(label="Duration (ms)", value=duration, delta=None)

    elif '.' in data:  # Check if the data contains a decimal point
        try:
            ecg_values.append(float(data))
            if len(ecg_values) >= 500:
                chart.plotly_chart(go.Figure(), use_container_width=True)
                ecg_values.clear()
            else:
                # Update the chart with new ECG data
                chart.plotly_chart(go.Figure(data=[go.Scatter(x=list(range(len(ecg_values))), y=ecg_values, mode='lines', name='ECG Sensor ADC Value')]), use_container_width=True)
        except ValueError:
            pass  # Ignore if the data cannot be converted to float
    else:
        pass  # Ignore other cases

# Function to continuously read data from serial and update the graph
def update_graph():
    try:
        while True:
            line = ser.readline()
            parse_serial_data(line)
    except KeyboardInterrupt:
        ser.close()

update_graph()
```

Figure 48. Dashboard script.

The main functionality of the web app revolves around the real-time acquisition and visualization of ECG data. The application continuously reads data from the serial port within a loop. For real-time visualization, ECG signal values are plotted on a Plotly graph embedded within the Streamlit interface. The graph updates as new data points are received and processed.

3.6 Wearable Design

SolidWorks is installed and components are selected to design the wearable. The components selected include a Li-ion battery for powering the microcontroller, a voltage regulator for recharging the battery, an AD8232 ECG sensor module for capturing ECG values, a microcontroller with ARM Cortex-M4 (STM32MP157F-DK2), and a clipper to clip the wearable device on the waist of the user. The case of the device can be 3D-printed using environmentally friendly material.

4. Results and Discussion

4.1 Overview of Results and Discussion

This comprehensive section presents all of the results covered in each module from methodology. These include the results of STM32MP157F-DK2 hardware and software setup, AD8232 ECG sensor setup and program, emotion classification model, firmware development to obtain real-time model inference, web app design and development, and wearable design. In-depth analysis will be focused on the emotion classification model. A comparative analysis between the ANN and CNN models is also provided, highlighting their respective strengths and trade-offs.

4.2 STM32MP157F-DK2 and AD8232 ECG Sensor Setup and Connection

The STM32MP157F-DK2 and AD8232 ECG sensor are set up as shown in Figure 49 below.

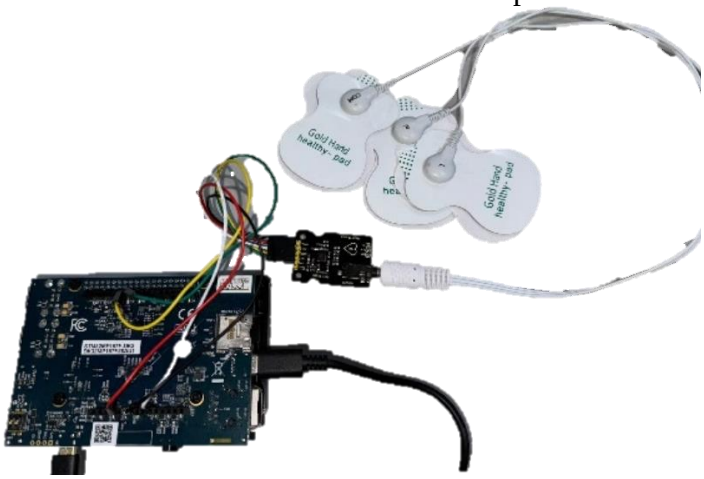


Figure 49. STM32MP157F-DK2 and AD8232 Setup.

The software is set up successfully codes can be compiled onto the STM32MP157F-DK2. The ECG values can be successfully obtained as shown in Figure 80.

4.3 Emotion Classification Model

4.3.1 Challenges Faced

During the development of the emotion monitoring system, several challenges related to the hardware limitations of the ARM Cortex-M4 microprocessors were encountered. These include insufficient FLASH and SRAM memories.

4.3.1.1 Insufficient FLASH Memory

WESAD contains 4 classes of emotions, which are baseline, stress, amusement and meditation. As mentioned in Section 3.3.2, an initial model with 4-class ANN has been developed with the reference of [70] and [71]. Hence, the design of the initial model is as follows:

Table 7. Initial Model Design.

Design	**Parameters**
Architecture	ANN
Number of Classes	4 (Baseline, Stress, Amusement, Meditation)
Duration of Input Signal	10 seconds
Frequency of Input Signal	7000Hz
Number of Hidden Layers	8
Number of Hidden Nodes in each Layer	1024, 512, 256, 128, 64, 32, 16, 4

When the DNN model is converted to TensorFlow Lite and uploaded onto the STM32CubeIDE, an error was prompted saying that the FLASH Memory is insufficient which is shown in Figure 50.

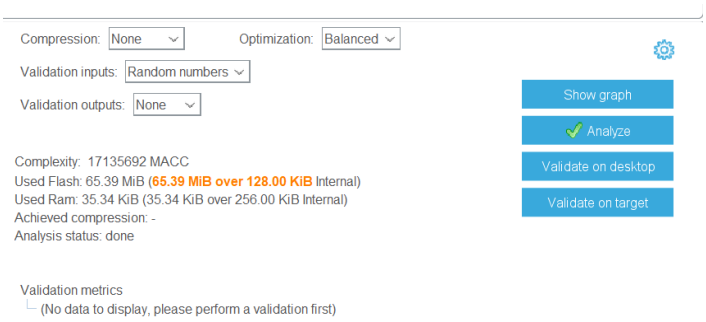


Figure 50. Flash Memory Insufficient.

The initial ANN model exceeded the available FLASH memory capacity of the microcontroller, preventing the successful deployment of the model. The FLASH memory is a non-volatile storage medium used to store the program code and the model parameters. The limited capacity of the FLASH memory posed a significant challenge, as the model's size exceeded the available space, making it impossible to load and execute the model on the STM32MPF157-DK2.

4.3.1.2 Insufficient SRAM Memory

After ensuring that the FLASH Memory and RAM are sufficient to accommodate the model, firmware is developed to deploy the DNN model using real-time ECG sensor data, as previously explained.

```
./../../../arm-none-eabi/bin/ld.exe: C:/ST/STM32CubeIDE_1.13.2/STM32CubeIDE/plugins/com.st.stm32cube.ide.mcu.externalto

./../../../arm-none-eabi/bin/ld.exe: C:/ST/STM32CubeIDE_1.13.2/STM32CubeIDE/plugins/com.st.stm32cube.ide.mcu.externalto

./../../../arm-none-eabi/bin/ld.exe: emotion_ai_CM4.elf section `.rodata' will not fit in region `SRAM1_text'
./../../../arm-none-eabi/bin/ld.exe: region SRAM1_text overflowed with text and data
./../../../arm-none-eabi/bin/ld.exe: section .data VMA [0000000010020000,0000000010020c93] overlaps section .rodata VMA
./../../../arm-none-eabi/bin/ld.exe: section .init_array VMA [0000000010026f00,0000000010026f03] overlaps section .bss
./../../../arm-none-eabi/bin/ld.exe: region `SRAM1_text' overflowed by 28424 bytes
```

Figure 51. Insufficient SRAM when building the project.

When building the project, the compiler prompted an error saying that .rodata would not fit in region SRAM1_text and SRAM1_text overflowed with text and data. In addition to the FLASH memory constraint, the ANN model required more Static Random-Access Memory (SRAM) than what was available on the hardware. The SRAM is a volatile memory used for storing intermediate computations and activations during the model inference. The limited capacity of the SRAM led to building failures when attempting to deploy the model, as illustrated in Figure 51.

Upon further research and debugging, it was determined that the 7000 input array is too large for the hardware and needs to be reduced. Hence, experiments in Section 3.3.3.2 are used to determine the optimum input size array that maximizes the hardware capability while having the highest possible accuracy for the model. Hence, various sizes of the input array are experimented with, which will be further elaborated.

4.3.1.3 Model Overfitting

As noted earlier, the WESAD dataset comprises four classes. Following the discussion, the initial ANN model is trained, with results illustrated in Figure 52. While the model achieves a high training accuracy of 97%, the validation accuracy is significantly lower at 31%. This discrepancy indicates substantial overfitting.

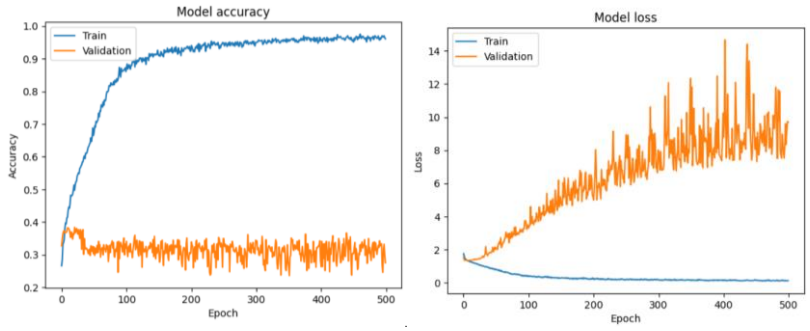


Figure 52. Model Accuracy and Model Loss of Initial ANN.

One primary cause is the imbalanced distribution of class sizes, a phenomenon demonstrated in Figure 53. Overfitting occurs when a model learns to perform well on the training data but fails to generalize unseen data, often due to its complexity or insufficient data representation from certain classes. In this case, the overfitting is caused by the uneven distribution of classes within the dataset.

Figure 53. Uneven class distribution.

4.3.2 Solutions

To address the challenges faced, the following solutions were implemented:

4.3.2.1 Reduction of Hidden Layers and Nodes

The complexity of the ANN model was reduced by decreasing the number of hidden layers and nodes in each layer. This approach aimed to reduce the model's size and memory requirements while maintaining acceptable performance. By reducing the number of parameters and computations, the model's size was reduced, making it more compatible with the hardware's limited resources. Therefore, several experiments aimed at identifying the optimal model for deployment on the hardware are undertaken which is elaborated in Section 3.3.3.

4.3.2.2 Model Compression

Apart from simplifying the model's complexity, the compression function within STM32CubeIDE reduces the required FLASH memory. Figure 54 illustrates the application of high compression to the ANN model with 6 hidden layers.

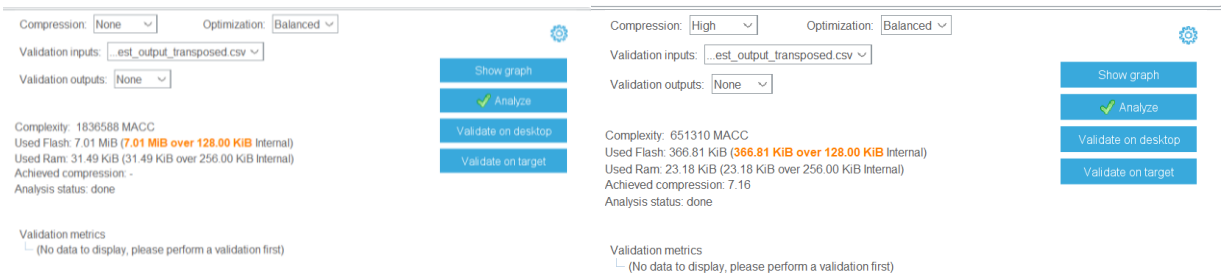


Figure 54. FLASH Memory before and after Compression.

However, instead of applying high compression, only medium compression was utilized at the end. This decision was made to balance between reducing memory requirements and maintaining model accuracy. High compression might lead to a significant loss of model performance, whereas medium compression strikes a better compromise, ensuring adequate memory reduction while preserving model effectiveness.

4.3.2.3 Normalisation of Dataset

Unequal dataset sizes among classes are a significant contributor to overfitting problems. Therefore, data normalisation is conducted to address this issue by evenly distributing data across classes, mitigating the impact of class imbalances. This process ensures that each class contributes proportionally to the model's learning, thereby reducing the risk of overfitting and improving generalization performance.

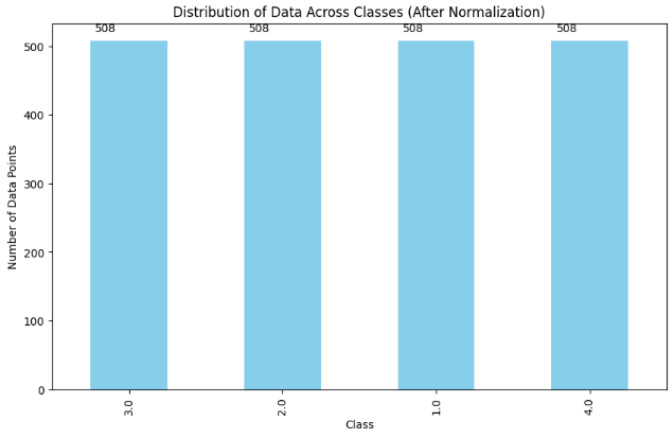


Figure 55. Data Normalisation.

Once the normalised data is trained with the initial ANN model proposed, the validation accuracy is able to improve by roughly 10% as illustrated in Figure 56. However, the overfitting problem still exists and needs to be addressed.

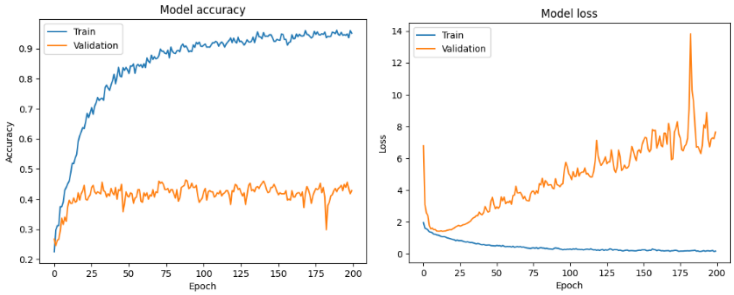


Figure 56. Performance of 5 layers ANN after Normalisation.

4.3.2.4 Complexity Reduction to Binary Classification

To further reduce the overfitting issue, the initial model's 4-class classification setup was simplified to binary classification, specifically targeting "stress" and "non-stress" categories. Consequently, a new dataset with $500Hz$ and 2 classes was generated and trained were created and trained. This adjustment effectively addressed the overfitting challenges, as demonstrated in Figure 57. By focusing solely on distinguishing between stress and non-stress states, the model's complexity was reduced, leading to improved generalization and performance.

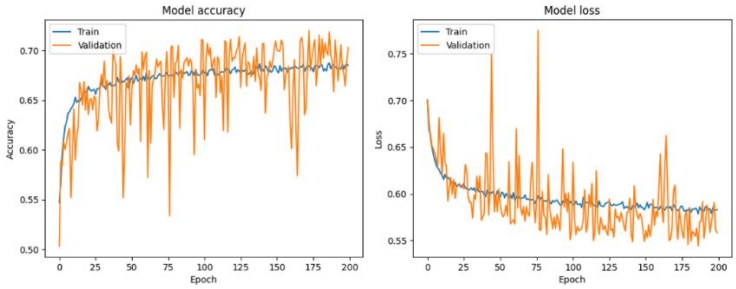


Figure 57. Model Performance after Reduce to Binary Classification.

4.3.3 Experiments Result

4.3.3.1 Experiment 1 (ANN): Reduction of Hidden Layers and Nodes

Architecture 1. Initially, the ANN architecture is shown in Figure 58:

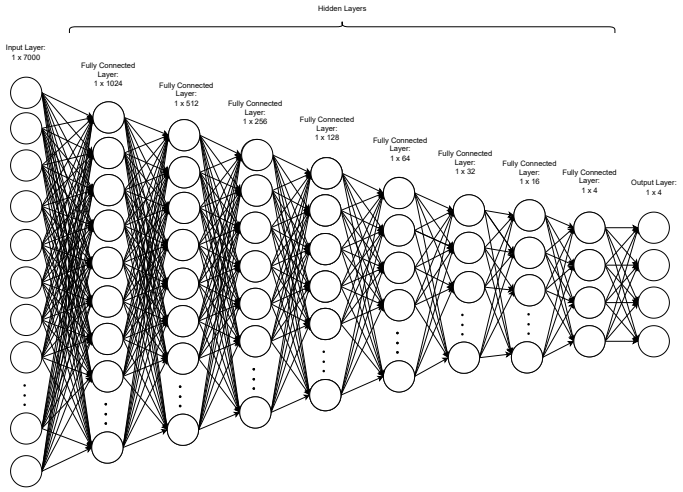


Figure 58. Initial ANN Architecture.

The design architecture of the initial ANN is as follows:

Table 8. Design Architecture of the Initial ANN.

Design	**Parameters**
Number of Hidden Layers	*8*
Number of Hidden Nodes in each Layer	*1024, 512, 256, 128, 64, 32, 16, 4*

The initial design utilizes a neural network model with a substantial size, which is up to $30.03MiB$. This model poses challenges for hardware-constrained STM32MP157F-DK2. This is because the maximum FLASH memory available is only $128KiB$. Hence, the model size needs to be reduced and compressed significantly to be deployed into the hardware.

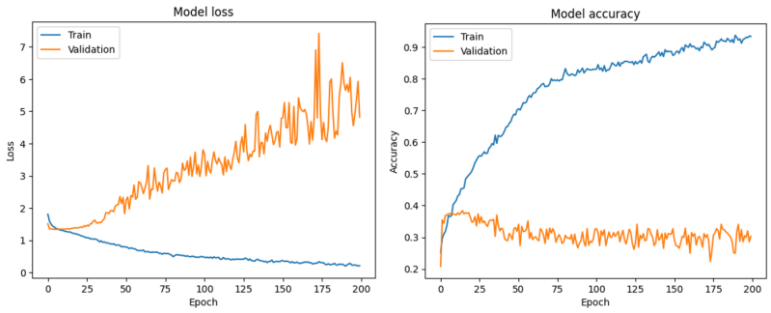


Figure 59. Performance of initial ANN Model.

As illustrated in Figure 59, overfitting occurs when the model learns the training data too well, capturing noise and specific patterns that do not generalize the unseen data. Overfitting can be seen when the training accuracy reaches approximately 100% while the validation accuracy remains around 30%.

Architecture 2. Subsequently, the ANN architecture is shown in Figure 60:

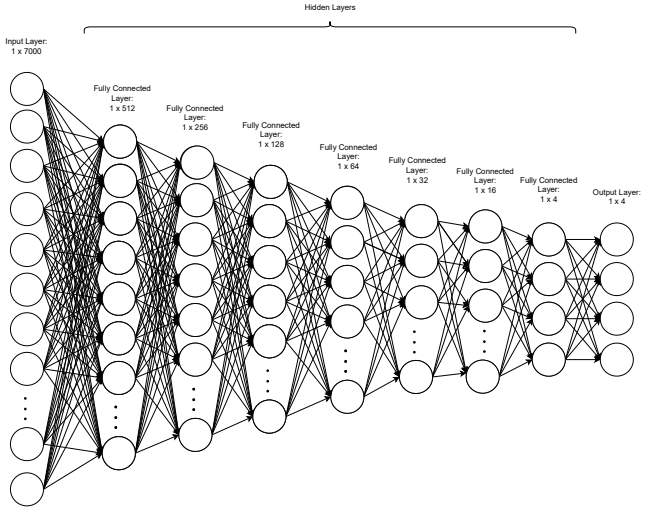


Figure 60. ANN Architecture Design 2.

The design architecture of the initial ANN is as follows:

Table 9. Design 2 for Architecture of the ANN.

Design	**Parameters**
Number of Hidden Layers	*7*
Number of Hidden Nodes in each Layer	*512, 256, 128, 64, 32, 16, 4*

The subsequent design utilizes a neural network model with a smaller size, which is $14.35MiB$. This model still poses challenges for hardware-constrained as the model size is still too large after compression.

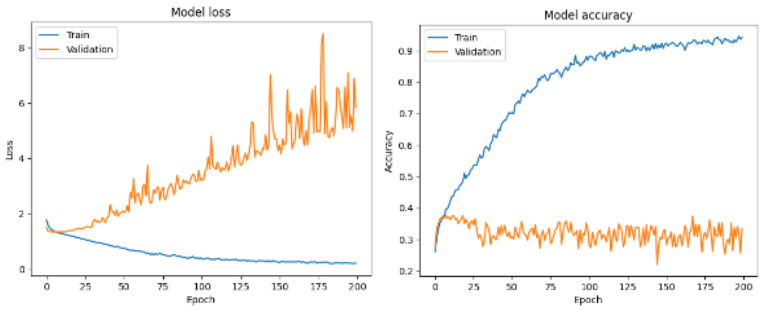


Figure 61. Performance of ANN Model for Architecture 2.

As illustrated in Figure 61, not only does overfitting occur, but the performance of the model in terms of accuracy is lower and the loss is higher. This happens as the model architecture complexity has been reduced and hence the model could not learn as detailed as the previous model.

Architecture 3. Next, the ANN architecture is further reduced as shown in Figure 62:

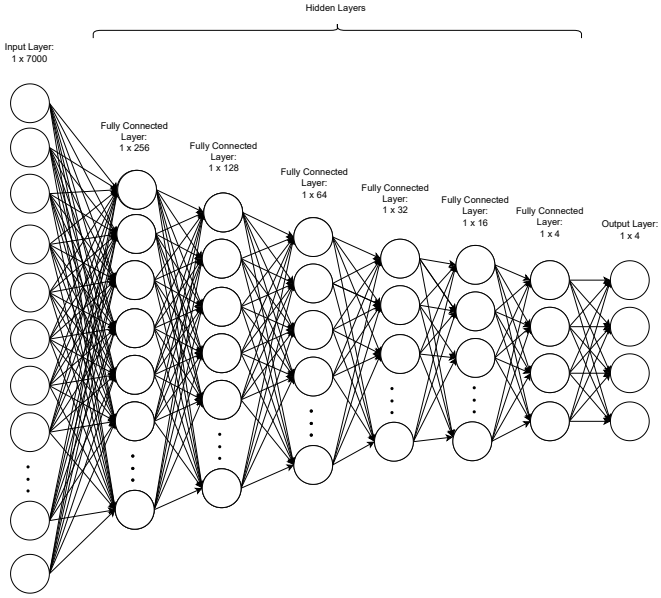


Figure 62. ANN Architecture Design 3.

The design architecture of the initial ANN is as follows:

Table 10. Design 3 for Architecture of the ANN.

Design	**Parameters**
Number of Hidden Layers	*6*
Number of Hidden Nodes in each Layer	*256, 128, 64, 32, 16, 4*

This design utilizes a neural network model with a smaller size, which is $7.01MiB$. When the model is uploaded onto STM32CubeIDE, it remains too large for the hardware even after compression.

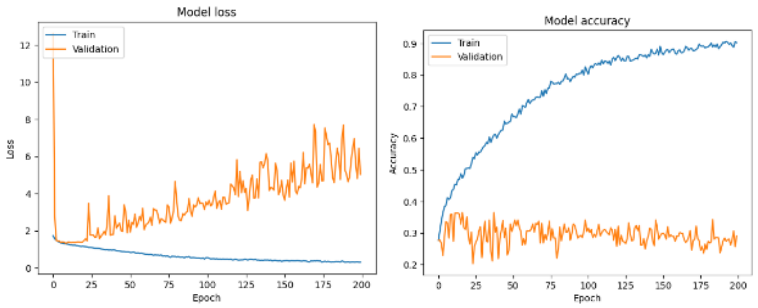


Figure 63. Performance of ANN Model for Architecture 3.

As illustrated in Figure 63, the performance of the model in terms of accuracy kept on decreasing and the loss was even higher. This proves that compromises between model performance and model size need to be made.

Architecture 4. Following on, the ANN architecture is shown in Figure 64:

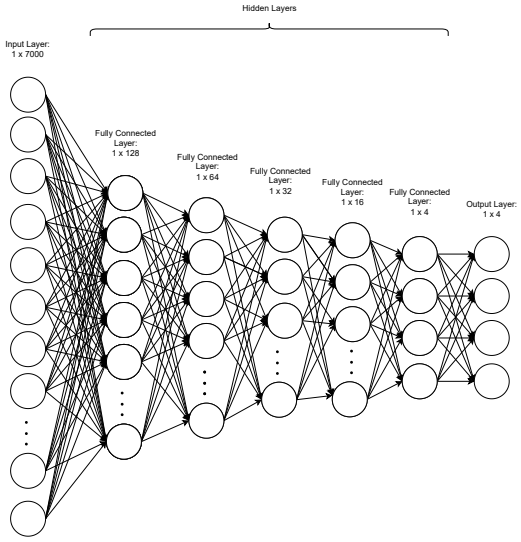


Figure 64. ANN Architecture Design 4.

The design architecture of the initial ANN is as follows:

Table 11. Design 4 for Architecture of the ANN.

Design	**Parameters**
Number of Hidden Layers	*5*
Number of Hidden Nodes in each Layer	*128, 64, 32, 16, 4*

This model achieves a size of $7.01MiB$ and it is able to fit into the FLASH and RAM after medium compression is applied as shown in Figure 65.

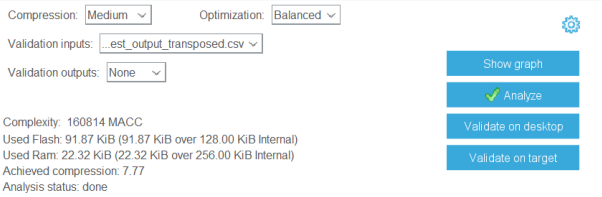


Figure 65. Model fit into FLASH and RAM after compression.

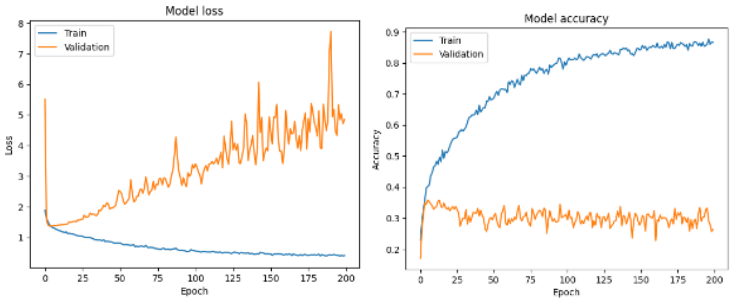


Figure 66. Performance of initial ANN Model for Architecture 4.

As illustrated in Figure 66, although this model can be fitted onto the hardware, the performance in terms of accuracy is the lowest among all of the previous models, with training accuracy reaching a maximum of less than 90%.

The reduction in model complexity proves the ability to deploy ANN in ARM Cortex-M4, though with a slight decrease in accuracy compared to the initial, more complex model. This trade-off was necessary to ensure the model's compatibility with the hardware limitations and enable real-time inference capabilities on the resource-constrained device. However, a significant challenge arises when attempting to flash the model onto the hardware, as an error message is displayed in STMCubeIDE indicating insufficient SRAM availability, which was discussed in Section 4.3.1.2. Hence, complexity reduction in terms of input data and model needs to be made to overcome the two main challenges faced – insufficient SRAM and model overfitting.

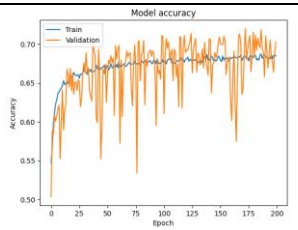
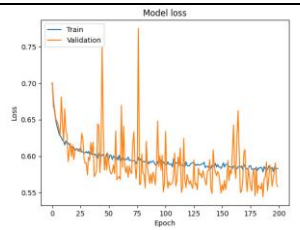
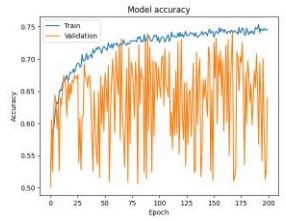
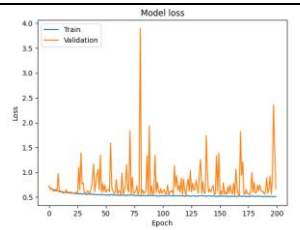
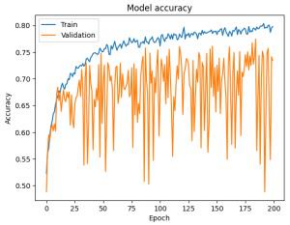
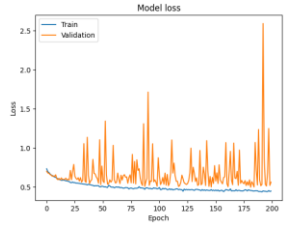
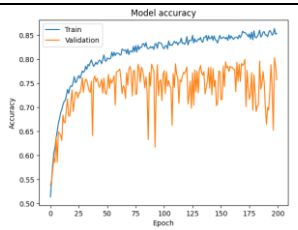
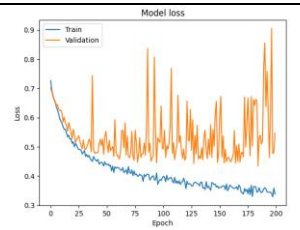
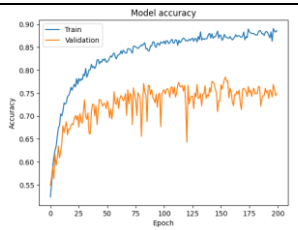
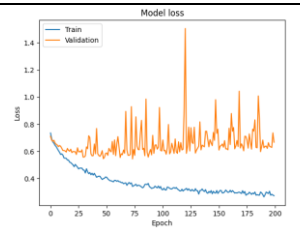
4.3.3.2 Experiment 2 (ANN): Testing the Hardware Limitation using the ANN Model

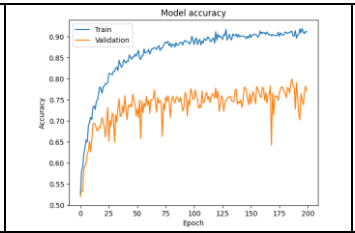
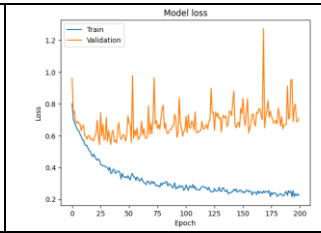
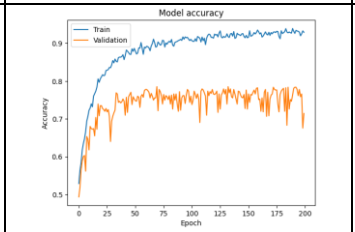
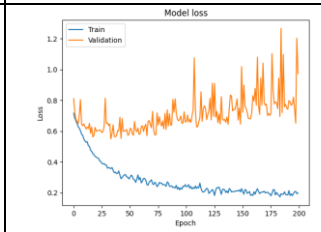
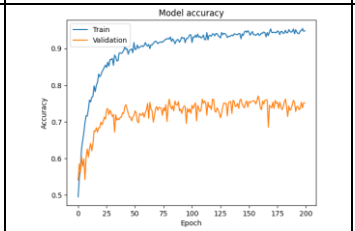
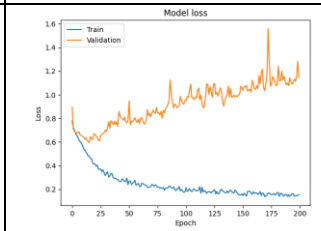
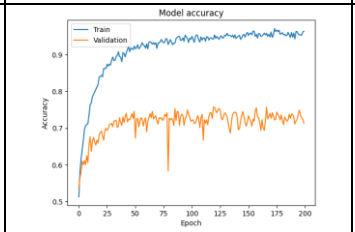
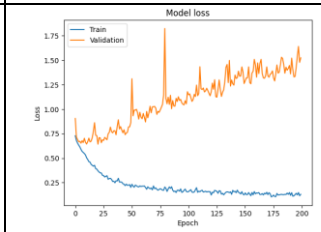
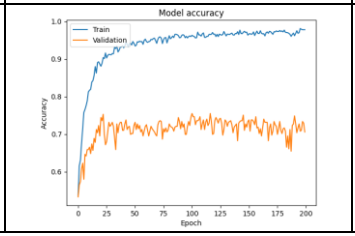
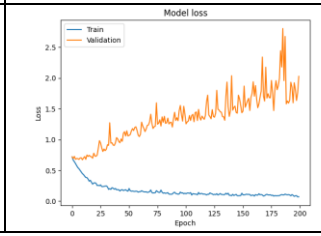
Table 12 shows that the model achieved its peak performance of 78.74% accuracy with 1500 input data points at 3 seconds. However, as the input data size increased beyond this point, the model started exhibiting signs of overfitting, suggesting that further fine-tuning techniques need to be applied before deploying the model in production.

While all model configurations remained within the FLASH memory and RAM limits, a hardware limitation was observed when handling datasets larger than 3000 input data points (at 7 seconds and beyond). Due to constraints, the hardware could not compile and execute arrays exceeding 3000 elements. This highlights the need to either optimize the hardware resources or simplify the model architecture to overcome this limitation before deployment.

The experiment highlights the trade-off between model performance and hardware limitations when working with ANNs. While increasing the input data size can potentially improve model accuracy up to a certain point, it also increases the hardware resource requirements. The results indicate that there is a hardware limitation beyond which the ANN model cannot be compiled and deployed successfully on the STM32MP157F-DK2. It is important to determine a balance between model performance and hardware limitations, as pushing the hardware beyond its capabilities can lead to failures or instability. To overcome the limitation, architecture optimization can be explored to further improve the result which is experimented with in Section 3.3.3.3.

Table 12. Experiment Results to test the Hardware Limitation using the ANN model.

Number of Data Input	Test Duration (s)	Compression & Optimization	Flash Memory /128 (KiB)	RAM /256 (KiB)	Model Accuracy	Model Loss	Test Accuracy (%)	Compile and Build
500	1	Medium Balanced	19.87	4.74			71.08	✓
1000	2	Medium Balanced	27.68	6.69			74.18	✓
1500	3	Medium Balanced	37.18	8.64			78.74	✓
2000	4	Medium Balanced	44.99	10.6			76.13	✓
2500	5	Medium Balanced	52.8	12.55			76.95	✓

3000	6	Medium Balanced	60.62	14.5			75.93	✓
3500	7	Medium Balanced	68.43	16.46			75.34	×
4000	8	Medium Balanced	76.24	18.41			73.22	×
4500	9	Medium Balanced	84.05	20.36			71.58	×
5000	10	Medium Balanced	91.87	22.32			73.19	×

In this experiment, the ANN model achieved a test accuracy of up to 78.74% with 1500 inputs, which is close to ideal performance. Subsequently, the model was converted to TensorFlow Lite and tested on hardware. The model was successfully built and compiled and the FLASH memory and RAM required during runtime is illustrated in Figure 67.

```
arm-none-eabi-size  emotion_ai_CM4.elf
arm-none-eabi-objdump -h -S emotion_ai_CM4.elf  > "emotion_ai_CM4.list"
   text    data     bss     dec     hex filename
 103028    3228   17392  123648   1e300 emotion_ai_CM4.elf
Finished building: default.size.stdout

Finished building: emotion_ai_CM4.list
```

Figure 67. Successful Build.

Based on [74], "text" contains code and constant data, which ends up in the FLASH memory. The "data" is used for initialized data, consuming both FLASH memory and RAM, as the variable will end up in RAM while the initialization of the variable requires FLASH memory. Additionally, "bss" contains uninitialized data and hence will be stored in RAM. Hence, equations can be made such that:

$$FLASH\ memory = text + data \qquad \textit{Equ 4}$$

$$RAM = data + bss \qquad \textit{Equ 5}$$

The "dec" represents the sum of "text", "data" and "bss". Based on Figure 67, the FLASH memory used is $106{,}256\ bytes$ and the RAM used is $20620\ bytes$.

4.3.3.3 Experiment 3 (CNN): Testing the Hardware Limitation using the CNN model

Architecture 1. Initially, the 1D-CNN architecture is shown in Figure 68:

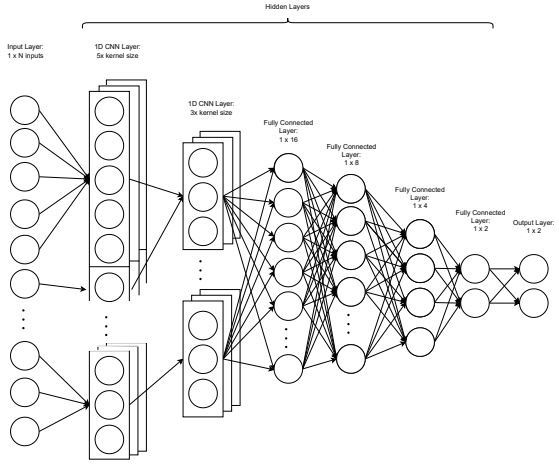


Figure 68. Initial 1D-CNN Architecture.

The design specifications of the 1D-CNN are as follows:

Table 13. Design Specifications of the Initial 1D-CNN.

Design	**Parameters**
Number of Data Inputs	3000
Number of Hidden Layers	*6 (2x 1D-CNN, 4x FC)*
Number of Hidden Nodes in each Layer	*5x kernel size, 3x kernel size, 16, 8, 4, 2*

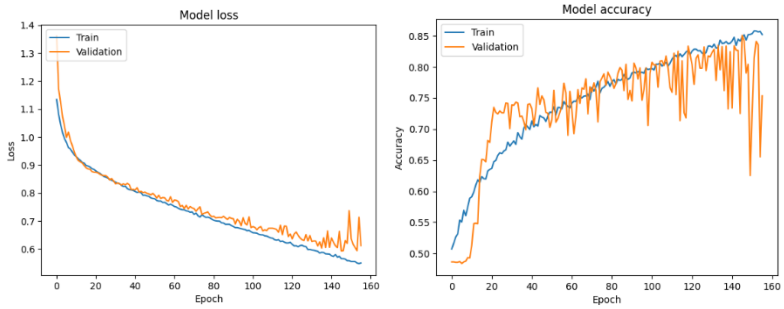


Figure 69. Performance of initial 1D-CNN Model.

As illustrated in Figure 69, the overfitting issue is resolved when the validation accuracy is approximately close to the training accuracy. During testing, the model is able to achieve up to 83.91%. However, this model poses challenges for hardware-constrained STM32MP157F-DK2 as it cannot be compiled and built on STM32CubeIDE after compression due to limited FLASH memory and RAM. Hence, a reduction of number of inputs is needed and performed in the next design.

Architecture 2. Next, the same 1D-CNN architecture is used.

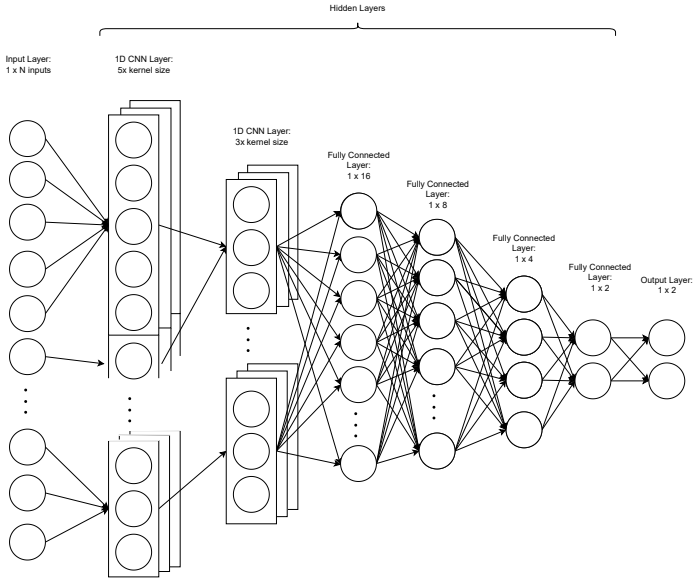


Figure 70. 1D-CNN Architecture for the 2nd iteration.

The design specifications of the initial 1D-CNN are as follows:

Table 14. Design Specifications of the 1D-CNN for the 2nd iteration.

Design	**Parameters**
Number of Data Inputs	1500
Number of Hidden Layers	*6 (2x 1D-CNN, 4x FC)*
Number of Hidden Nodes in each Layer	*5x kernel size, 3x kernel size, 16, 8, 4, 2*

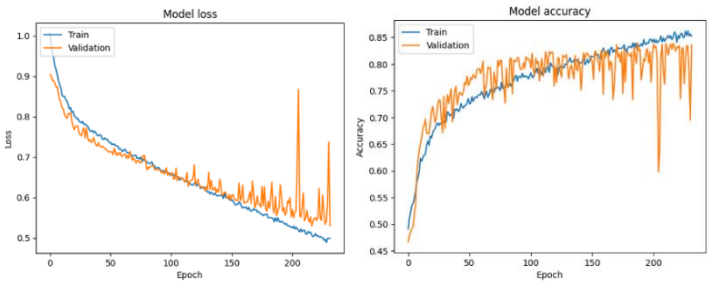


Figure 71. Performance of 1D-CNN Model for the 2nd iteration.

Compared to Architecture 1, the performance of the model in terms of accuracy decreases slightly. During testing, the model is able to achieve up to 83.47%. However, this model still cannot be compiled and built on STM32CubeIDE after compression due to limited FLASH memory and RAM. Thus, a further reduction of number of inputs is needed and performed in the next design.

Architecture 3. Subsequently, the same 1D-CNN architecture is used.

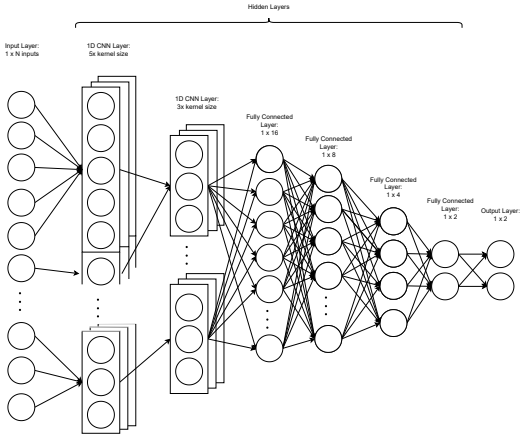


Figure 72. 1D-CNN Architecture for the 3rd iteration.

The design specifications of the 1D-CNN are as follows:

Table 15. Design Specifications of the 1D-CNN for the 3rd iteration.

Design	**Parameters**
Number of Data Inputs	1000
Number of Hidden Layers	*6 (2x 1D-CNN, 4x FC)*
Number of Hidden Nodes in each Layer	*5x kernel size, 3x kernel size, 16, 8, 4, 2*

The performance of this model is illustrated in Figure 73.

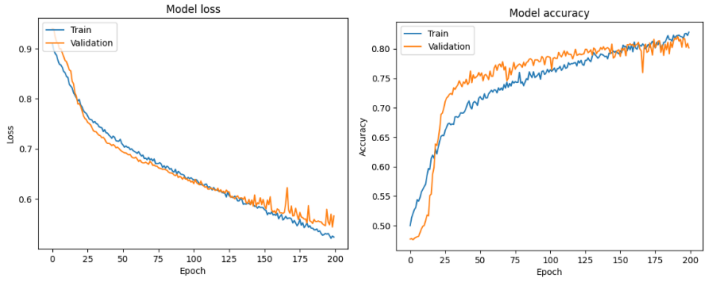


Figure 73. Performance of 1D-CNN Model for the 3rd iteration.

Compared to Architecture 2, the performance of the model in terms of accuracy decreases even more. During testing, the model is able to achieve up to 81.44%. As illustrated in Figure 74, the validation accuracy at certain points is greater than the training accuracy. This is caused due to the dropout and regularization hyperparameters used to prevent the model from overfitting. Finally, the model is able to converge at approximately 170 epochs. However, this model still cannot be compiled and built on STM32CubeIDE after compression due to limited FLASH memory and RAM. Thus, a further reduction of number of inputs and model architecture is needed and performed in the next design.

Architecture 4. Subsequently, a new and smaller 1D-CNN architecture is proposed.

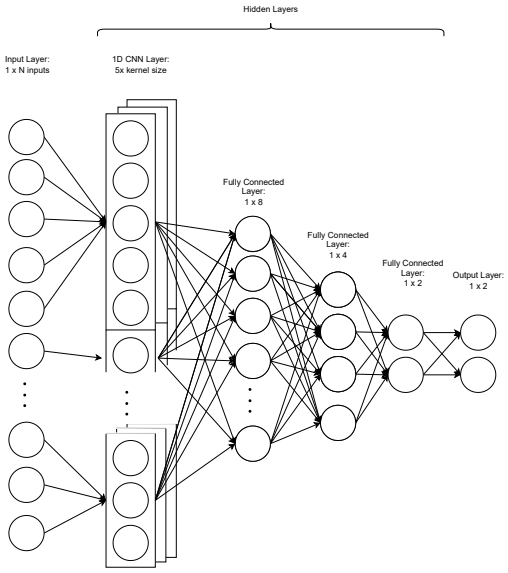


Figure 74. 1D-CNN Architecture for the 4th iteration.

The design specifications of the 1D-CNN are as follows:

Table 16. Design Specifications of the 1D-CNN for the 4th iteration.

Design	**Parameters**
Number of Data Inputs	500
Number of Hidden Layers	*4 (1x 1D-CNN, 3x FC)*
Number of Hidden Nodes in each Layer	*5x kernel size, 8, 4, 2*

The performance of this model is illustrated in Figure 75.

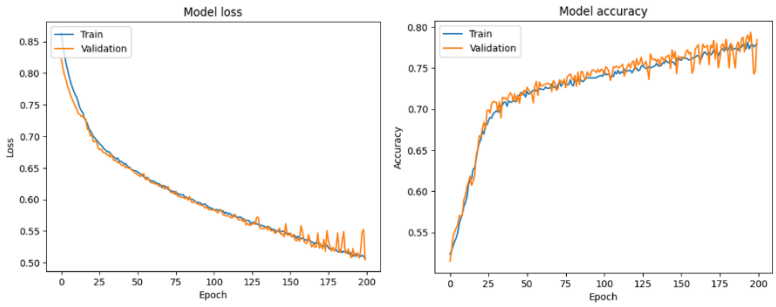


Figure 75. Performance of 1D-CNN Model for the 4th iteration.

Compared to Architecture 3, the performance of the model in terms of accuracy decreases even more. During testing, the model is able to achieve up to 79.34%. However, the learning curve for both training and validation accuracies are identical, which is ideal as the model is learning well. Moreover, this model is able to compile and build on STM32CubeIDE after compression.

Architecture 5. Following this, a new 1D-CNN architecture is proposed to determine the model performance in terms of accuracy.

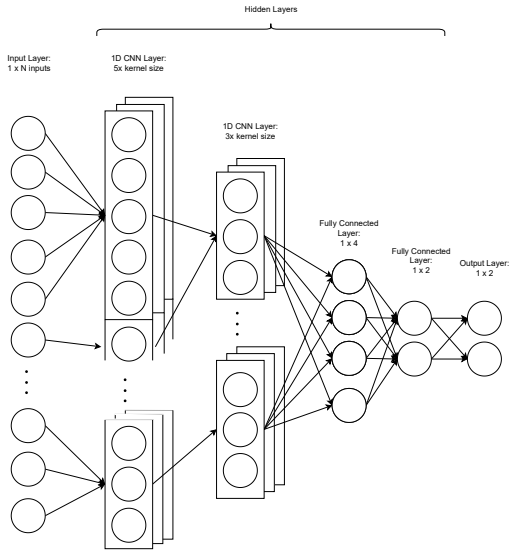


Figure 76. 1D-CNN Architecture for the 4th iteration.

The design specifications of the 1D-CNN are as follows:

Table 17. Design Specifications of the 1D-CNN for the 4th iteration.

Design	**Parameters**
Number of Data Inputs	1000
Number of Hidden Layers	*4 (2x 1D-CNN, 2x FC)*
Number of Hidden Nodes in each Layer	*5x kernel size, 3x kernel size, 4, 2*

The performance of this model is illustrated in Figure 77.

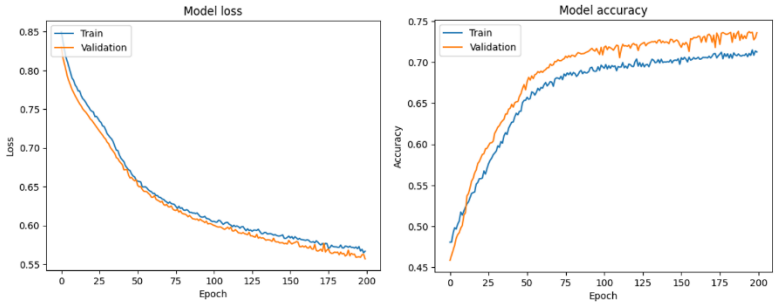


Figure 77. Performance of 1D-CNN Model for the 4th iteration.

Compared to Architecture 4, the performance of the model in terms of accuracy decreases even more. During testing, the model can achieve up to 74.31%. Although the model is learning well and can be compiled and built into the hardware, the model performance in terms of accuracy is lower than in Architecture 4.

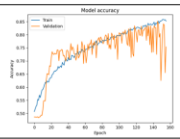
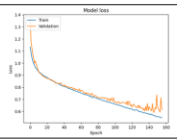
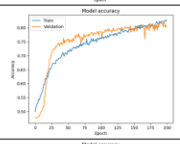
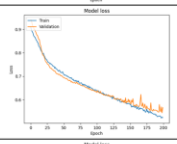
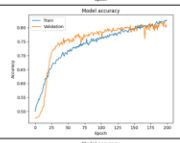
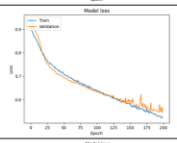
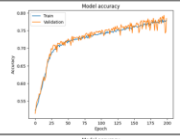
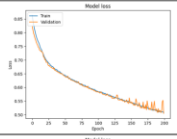
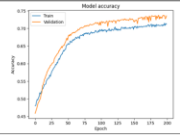
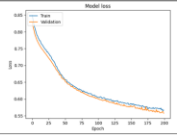
Initially, despite achieving an impressive accuracy of 83.91% with a CNN model comprising 2 layers of 1D-CNN and 4 fully connected layers, hardware limitations posed a significant barrier. The model's inability to be built and compiled due to constraints in FLASH memory and RAM highlighted the critical importance of considering hardware constraints during model design.

Subsequent efforts to reduce the input size while retaining the architecture proved insufficient in overcoming these hardware limitations. However, the decision to further reduce the input size to 500 and simplify the architecture to a single layer of 1D-CNN and 3 fully connected layers produced promising results. Despite a slight decrease in test accuracy to 79.34% compared to the initial model, this configuration successfully addressed the hardware constraints while avoiding overfitting issues.

Further experimentation explored the feasibility of accommodating 1000 input data with a modified architecture of 2 layers of 1D-CNN and 2 fully connected layers. This adjustment required reducing the number of hidden nodes in the CNN layer but led to compromised test accuracy, peaking at 74.3%. This trade-off between model complexity and accuracy highlights the necessity of fine-tuning model architectures to strike an optimal balance between performance and hardware compatibility.

Overall, these findings emphasize the importance of iterative experimentation and adaptation in optimizing CNN models for deployment on resource-constrained hardware. By carefully adjusting model architectures and input sizes, hardware limitations can be mitigated while maintaining satisfactory performance levels. The result of Experiment 3 is summarized and tabulated in Table 18.

Table 18. Experiment Results to test the Hardware Limitation using the CNN Model.

Number of Data Input	Test Duration (s)	Architecture	Model Accuracy	Model Loss	Test Accuracy (%)	Compile and Build
3000	6	2x 1D CNN 4x FC			83.91	×
1500	3	2x 1D CNN 4x FC			83.47	×
1000	2	2x 1D CNN 4x FC			81.44	×
500	1	1x 1D CNN 3x FC			79.34	✓
1000	2	2x 1D CNN 2x FC			74.3	✓

4.3.4 Comparison between ANN and CNN

Based on the Experiment 2 and Experiment 3 tested, the highest accuracy models that can be built and compiled onto the hardware are selected. Figure 78 illustrates the architecture of the highest accuracy ANN model based on experiment results.

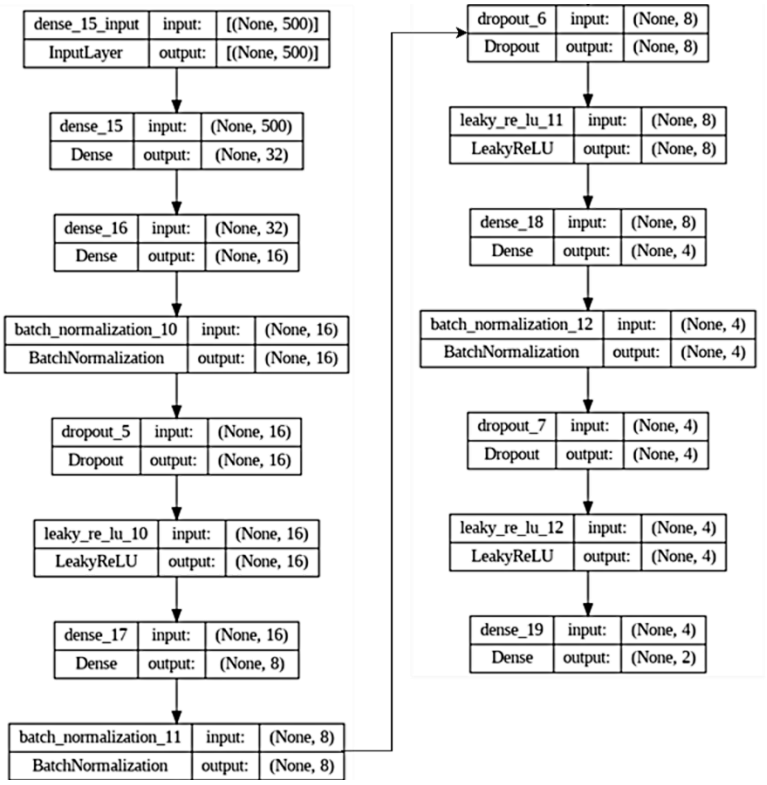


Figure 78. The architecture of the Best ANN Model.

Figure 79 illustrates the architecture of the highest 1D-CNN model based on experiment results.

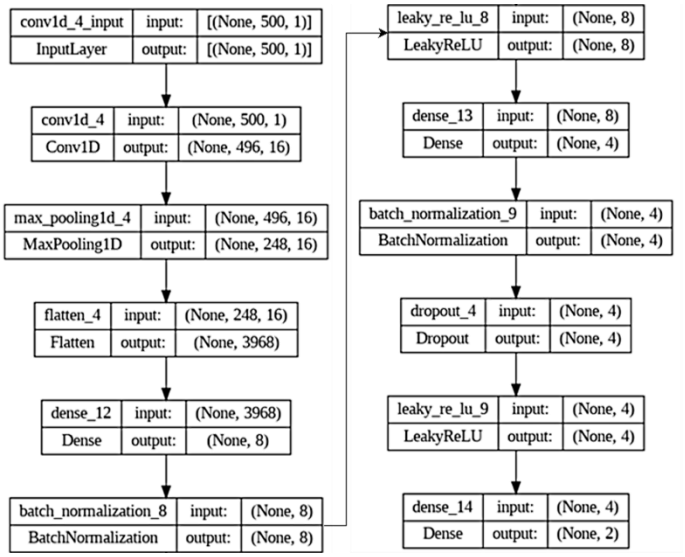


Figure 79. The architecture of the Best 1D-CNN Model.

The performance of each architecture that maximizes the hardware capabilities and test performance during production mode is summarized in Table 19.

Table 19. Comparison between best performance ANN and CNN that can be compiled on hardware..

	ANN	**1D-CNN**
Number of Input Data	1500	500
Number of Hidden Layers	5	4
FLASH memory ($bytes$)	131040	115108
RAM ($bytes$)	20620	44444
Inference Time (μs)	6293	13427
Test Accuracy (%)	78.74	79.34
Overfitting	Slightly	None

In terms of the number of input data, the ANN utilizes 1500 data points, while the CNN operates with a reduced input size of 500 data points. Hence, this contributes to the higher amount of FLASH memory required during compilation. Furthermore, the number of hidden layers for ANN is greater than CNN and hence this also contributes to the higher amount of FLASH memory consumed.

While both architectures leverage multiple layers for learning complex patterns, the CNN achieves comparable performance with one fewer hidden layer, indicating its effectiveness in feature extraction and classification tasks. CNN requires higher RAM during compilation due to the more complex computation methods.

In terms of inference time, the ANN demonstrates faster processing, with an inference time of $6293\mu s$ compared to the CNN's $13427\mu s$. This result suggests that the ANN architecture may be more suitable for real-time applications requiring rapid decision-making.

In terms of evaluating test accuracy, the CNN outperforms the ANN, achieving a slightly higher accuracy of 79.34% compared to the ANN's 78.74%. However, it is also crucial to take note that there is slight overfitting in the ANN model compared to the CNN model. This improvement in accuracy highlights the effectiveness of the CNN architecture, particularly in tasks that involve sequential and time series data.

The choice between the ANN and CNN models depends on the specific requirements of the application, such as the priority given to accuracy or inference time. If higher accuracy is the primary concern, the CNN model may be preferred, but if real-time performance is crucial, the ANN model may be a better choice due to its shorter inference time. Additionally, the CNN model's longer inference time may impact the overall responsiveness of the emotion monitoring system, which could be a consideration in certain applications.

4.4 Firmware Development to Obtain Real-Time Model Inference

Figure 80 shows the successful visualization of real-time ECG signals printed from serial COM on the computer.

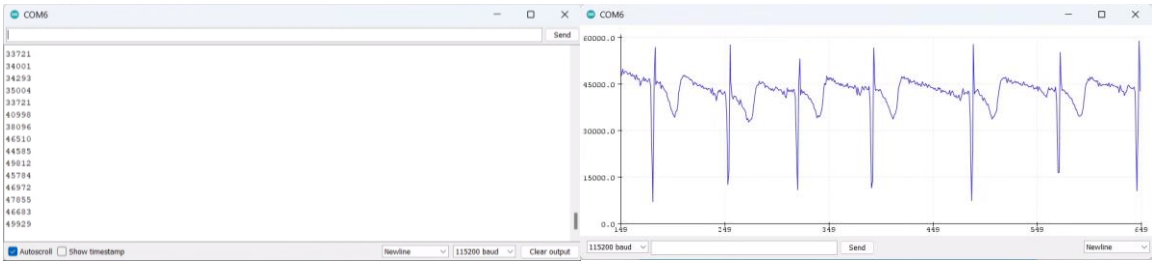


Figure 80. Serial monitor and serial plotter displaying 16-bit ADC values.

Similarly, the real-time inference of the model using the sensor value can be displayed via serial monitor as shown in Figure 81.

Figure 81. Serial monitor displaying AI Inference using real-time ECG data.

4.5 Web App Design and Development

The application ensures that the graph remains responsive and informative by updating the plot with every new batch of ECG data points received, maintaining a clear and continuous visual representation of the ECG signal. The dashboard is shown in Figure 82.

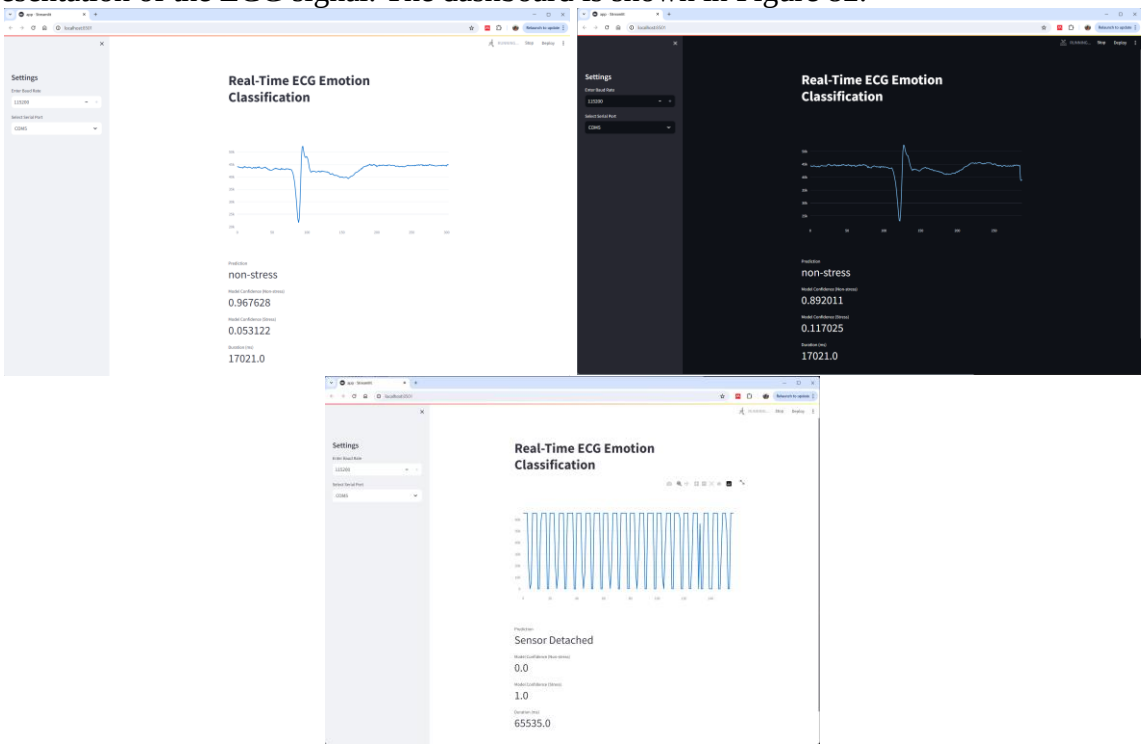


Figure 82. Real-time Dashboard.

On the left side of the dashboard, users have the flexibility to alter the baud rate and COM ports. Hence, this dashboard works on any computer running Windows Operating System. The dashboard is available in either light or dark mode, which improves visualization.

4.6 Wearable Design

Based on several references of wearable designs and sensor placement, an idea of the wearable is drawn and shown in Figure 83.

Figure 83. Wearable Prototype Design.

The wearable device can be conveniently attached to the side of the trousers, facilitating effortless accessibility and wearability. This also ensures comfort and user-friendliness when using the device.

5. Conclusion

In summary, the findings underscore the promising potential of merging ECG technology with AI algorithms for real-time emotion monitoring in resource-constrained applications. The development and deployment of an ECG-based emotion classification model on an STM32MP157F-DK2 using ARM Cortex-M architecture highlights its lightweight and swift performance, facilitating real-time inference. Throughout the project, two distinct model architectures were developed and tested. The ANN model attained a commendable accuracy of 78.74% with an inference time of $6293\mu s$, while the CNN model achieved a slightly higher accuracy of 79.34% with an inference time of $13427\mu s$. The selection between these models should be based on a comprehensive assessment of application requirements, hardware limitations, and the desired trade-off between accuracy, real-time functionality, and computational complexity.

Prior to the project, a HIRARC assessment had been performed to ensure all of the risks for the project. Health and safety measurements were also taken care of by applying alcohol before putting electrodes on the body for measurements and replacing electrodes after each use. Furthermore, ethical approval had been obtained in the early stages to ensure the project was done ethically.

The project has effectively addressed all its primary objectives and additional "stretch" goals, resulting in a comprehensive and functional system. These include:

- Objective 1 – To integrate the AD8232 sensor with STM32MP157F-DK2 and obtain 16-bit data that can be printed on a computer. This objective has been achieved fully and it is able to come out as a working model. The details are discussed in Section 3.2 and the result is shown in Figure 80. The ECG values can be printed out in real time on the serial monitor.

- Objective 2 – To collect and preprocess the dataset obtained. This objective has been successfully achieved where the details are discussed in Section 3.3.1 and the result is shown in Figure 33. A $500Hz$ normalized dataset has been created and filtered.
- Objective 3 – To train various DNN models that perform emotion classifications. This objective has been fully achieved as shown in Section 3.3.2 and the result is shown in Figures 78 and 79. ANN and 1D-CNN models are trained and a comparison between both of the models is discussed in Section 4.3.4.
- Objective 4 – To convert the model into TensorFlow Lite format so that it is optimized for the hardware. This objective has been fully achieved as discussed in Section 3.3.4 and shown in Figure 35. The TFLite format can be deployed into the hardware successfully as shown in Figure 65.
- Objective 5 – To investigate and optimize the performance of DNN models. This goal was achieved fully as discussed in Section 3.3.3 and the results can be found in Section 4.3.3. Experiments were performed to compare the limitations of the hardware, as well as determine the optimal performance in terms of accuracy and inference time. The result can be visualized in Figure 81.
- Objective 6 – To design a wearable for the real-time emotion classification system. This was part of the "stretch" goal planned during the pre-development stage and it was achieved partially. The design of the wearable was discussed in Section 3.6 and the 3D model is shown in Figure 83. However, the overall wearable is not printed out.
- Objective 7 – To design a real-time dashboard for ECG and model inference visualization. This objective is also part of the "stretch" goal planned. The objective has been achieved fully as discussed in Section 3.5. The dashboard design is shown in Figure 82 where it updates the data in real-time. Users have the flexibility to change the baud rate and COM port through the UI.

This project represents substantial progress in the field of emotion monitoring, laying the groundwork for further advancements. As hardware capabilities progress and optimization techniques evolve, it is foreseeable that even more precise and efficient models can be deployed on resource-constrained devices, facilitating diverse applications in healthcare, wellness, and beyond.

5.1 Reflection on Time Plan

The project's primary objectives were successfully achieved within the planned timeframe, with "stretch" goals also completed. The project resulted in the development of an optimized real-time ECG-based emotion classification wearable device that is accurate, fast, and lightweight. Several iterative experiments were conducted to ensure the project's capabilities and functionalities.

However, there were some delays encountered during the search for an available dataset, initially intending to utilize 2D-CNN on ECG signal images. Due to the high computational power requirement, a decision was made to use raw ECG signals as input for the model instead. Subsequent additional steps were carried out including preprocessing the dataset from WESAD, normalisation of datasets, and experimentation with various model architectures.

5.2 Future Work and Improvements

In future work, several enhancements and expansions of the current project can be explored. Firstly, the development of an arrhythmia detection model deployable in the cloud, utilizing the same ECG signals, holds the potential for extending the utility of the system in healthcare settings.

Efforts to reduce the cost of the microcontroller should be prioritized, making the technology more accessible to a wider range of users. Integrating the hardware onto a Printed Circuit Board (PCB) offers benefits such as the reduction in system size, aligning with the project's goal of creating a portable and wearable system. The power consumption of the STM32MP157F-DK2 running real-time AI inference can be determined and experimented with in future works. Low-power consumption devices can also be explored so that the wearable can last for a certain amount of time.

Moreover, as technology continues to advance, the adoption of wireless ECG electrodes and the integration of wearable devices into everyday clothing, such as shirts or vests, can greatly enhance wearability and comfort. This evolution in design would facilitate seamless integration into users' daily routines, further promoting the adoption of the technology.

Lastly, improvements in ECG sensor technology are essential to ensuring accurate readings even under conditions of rapid movement in various directions. By enhancing sensor capabilities, the system can maintain reliability and accuracy in diverse real-world scenarios, thereby increasing its utility and effectiveness in practical applications.

References

[1] P. Prajod and E. André, "On the Generalizability of ECG-based Stress Detection Models," in *Proceedings - 21st IEEE International Conference on Machine Learning and Applications, ICMLA 2022*, 2022. doi: 10.1109/ICMLA55696.2022.00090.

[2] P. Schmidt, A. Reiss, R. Duerichen, and K. Van Laerhoven, "Introducing WeSAD, a multimodal dataset for wearable stress and affect detection," *ICMI 2018 - Proceedings of the 2018 International Conference on Multimodal Interaction*, pp. 400–408, Oct. 2018, doi: 10.1145/3242969.3242985.

[3] S. Koldijk, M. A. Neerincx, and W. Kraaij, "Detecting Work Stress in Offices by Combining Unobtrusive Sensors," *IEEE Trans Affect Comput*, vol. 9, no. 2, pp. 227–239, Apr. 2018, doi: 10.1109/TAFFC.2016.2610975.

[4] B. Behinaein, A. Bhatti, D. Rodenburg, P. Hungler, and A. Etemad, "A Transformer Architecture for Stress Detection from ECG," *Proceedings - International Symposium on Wearable Computers, ISWC*, pp. 132–134, Aug. 2021, doi: 10.1145/3460421.3480427.

[5] I. Lucan Orășan, C. Seiculescu, and C. D. Căleanu, "A Brief Review of Deep Neural Network Implementations for ARM Cortex-M Processor," *Electronics (Switzerland)*, vol. 11, no. 16. 2022. doi: 10.3390/electronics11162545.

[6] U. Lundberg, "Stress and (Public) Health," *International Encyclopedia of Public Health*, pp. 241–250, Jan. 2008, doi: 10.1016/B978-012373960-5.00103-9.

[7] S. Reisman, "Measurement of physiological stress," *Bioengineering, Proceedings of the Northeast Conference*, pp. 21–23, 1997, doi: 10.1109/NEBC.1997.594939.

[8] G. Jun and K. G. Smitha, “EEG based stress level identification,” *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pp. 3270–3274, Feb. 2017, doi: 10.1109/SMC.2016.7844738.

[9] F. Agrafioti, D. Hatzinakos, and A. K. Anderson, “ECG pattern analysis for emotion detection,” *IEEE Trans Affect Comput*, vol. 3, no. 1, pp. 102–115, Jan. 2012, doi: 10.1109/T-AFFC.2011.28.

[10] P. Sarkar and A. Etemad, “Self-Supervised ECG Representation Learning for Emotion Recognition,” *IEEE Trans Affect Comput*, vol. 13, no. 3, 2022, doi: 10.1109/TAFFC.2020.3014842.

[11] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of Edge Computing and Deep Learning: A Comprehensive Survey,” *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2. 2020. doi: 10.1109/COMST.2020.2970550.

[12] A. L. Golande and T. Pavankumar, “Optical electrocardiogram based heart disease prediction using hybrid deep learning,” *J Big Data*, vol. 10, no. 1, 2023, doi: 10.1186/s40537-023-00820-6.

[13] A. Tjolleng *et al.*, “Classification of a Driver’s cognitive workload levels using artificial neural network on ECG signals,” *Appl Ergon*, vol. 59, 2017, doi: 10.1016/j.apergo.2016.09.013.

[14] J. Gordon Betts *et al.*, *Anatomy and Physiology 2e*. Houston, Texas: OpenStax, 2022.

[15] L. B. T. Yugar *et al.*, “The Role of Heart Rate Variability (HRV) in Different Hypertensive Syndromes,” *Diagnostics*, vol. 13, no. 4, Feb. 2023, doi: 10.3390/DIAGNOSTICS13040785.

[16] C. Clyburn, J. J. Sepe, and B. A. Habecker, “What gets on the nerves of cardiac patients? Pathophysiological changes in cardiac innervation,” *J Physiol*, vol. 600, no. 3, p. 451, Feb. 2022, doi: 10.1113/JP281118.

[17] P. J. Bota, C. Wang, A. L. N. Fred, and H. Placido Da Silva, “A Review, Current Challenges, and Future Possibilities on Emotion Recognition Using Machine Learning and Physiological Signals,” *IEEE access*, vol. 7, pp. 140990–141020, 2019, doi: 10.1109/ACCESS.2019.2944001.

[18] R. W. Levenson, “The autonomic nervous system and emotion,” *Emotion Review*, vol. 6, no. 2, pp. 100–112, Mar. 2014, doi: 10.1177/1754073913512003/ASSET/IMAGES/LARGE/10.1177_1754073913512003-FIG2.JPEG.

[19] B. Macukow, “Neural networks-state of art, brief history, basic models and architecture,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9842 LNCS, pp. 3–14, 2016, doi: 10.1007/978-3-319-45378-1_1/FIGURES/10.

[20] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-January, pp. 1–6, Jul. 2017, doi: 10.1109/ICENGTECHNOL.2017.8308186.

[21] Y. Wu, F. Yang, Y. Liu, X. Zha, and S. Yuan, “A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification,” Oct. 2018, Accessed: Mar. 31, 2024. [Online]. Available: https://arxiv.org/abs/1810.07088v1

[22] V. T. Ninh *et al.*, “An Improved Subject-Independent Stress Detection Model Applied to Consumer-grade Wearable Devices,” in *Lecture Notes in Computer Science (including*

subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2022. doi: 10.1007/978-3-031-08530-7_77.

[23] A. Liapis, E. Faliagka, C. Katsanos, C. Antonopoulos, and N. Voros, "Detection of Subtle Stress Episodes During UX Evaluation: Assessing the Performance of the WESAD Bio-Signals Dataset," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12934 LNCS, pp. 238–247, 2021, doi: 10.1007/978-3-030-85613-7_17.

[24] P. Sarkar and A. Etemad, "Self-Supervised ECG Representation Learning for Emotion Recognition," *IEEE Trans Affect Comput*, vol. 13, no. 3, pp. 1541–1554, 2022, doi: 10.1109/TAFFC.2020.3014842.

[25] E. Zhou, M. Soleymani, and M. J. Mataric, "Investigating the Generalizability of Physiological Characteristics of Anxiety," *Proceedings - 2023 2023 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2023*, pp. 4848–4855, 2023, doi: 10.1109/BIBM58861.2023.10385292.

[26] "(PDF) Emotion Classification Using Electrocardiogram and Machine Learning: A Study on the Effect of Windowing Techniques." Accessed: Apr. 01, 2024. [Online]. Available: https://www.researchgate.net/publication/371750271_Emotion_Classification_Using_Electrocardiogram_and_Machine_Learning_A_Study_on_the_Effect_of_Windowing_Techniques

[27] E. Zhou, M. Soleymani, and M. J. Mataric, "Investigating the Generalizability of Physiological Characteristics of Anxiety," *Proceedings - 2023 2023 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2023*, pp. 4848–4855, 2023, doi: 10.1109/BIBM58861.2023.10385292.

[28] M. Amin *et al.*, "ECG-Based Driver's Stress Detection Using Deep Transfer Learning and Fuzzy Logic Approaches," *IEEE Access*, vol. 10, pp. 29788–29809, 2022, doi: 10.1109/ACCESS.2022.3158658.

[29] R. Jegan, S. Mathuranjani, and P. Sherly, "Mental Stress Detection and Classification using SVM Classifier: A Pilot Study," *ICDCS 2022 - 2022 6th International Conference on Devices, Circuits and Systems*, pp. 139–143, 2022, doi: 10.1109/ICDCS54290.2022.9780795.

[30] M. Kang, S. Shin, J. Jung, and Y. T. Kim, "Classification of Mental Stress Using CNN-LSTM Algorithms with Electrocardiogram Signals.," *J Healthc Eng*, vol. 2021, pp. 9951905–9951905, Jun. 2021, doi: 10.1155/2021/9951905.

[31] X. Wu *et al.*, "Research on emotion recognition based on ECG signal," *J Phys Conf Ser*, vol. 1678, no. 1, p. 012091, Nov. 2020, doi: 10.1088/1742-6596/1678/1/012091.

[32] T. Dissanayake, Y. Rajapaksha, R. Ragel, and I. Nawinne, "An Ensemble Learning Approach for Electrocardiogram Sensor Based Human Emotion Recognition," *Sensors 2019, Vol. 19, Page 4495*, vol. 19, no. 20, p. 4495, Oct. 2019, doi: 10.3390/S19204495.

[33] C. Xiefeng, Y. Wang, S. Dai, P. Zhao, and Q. Liu, "Heart sound signals can be used for emotion recognition," *Sci Rep*, vol. 9, no. 1, Dec. 2019, doi: 10.1038/S41598-019-42826-2.

[34] G. Chen, Y. Zhu, Z. Yang, and Z. Hong, "Emotionalgan: Generating ECG to enhance emotion state classification," *ACM International Conference Proceeding Series*, pp. 309–313, Jul. 2019, doi: 10.1145/3349341.3349422.

[35] Siddharth, T. P. Jung, and T. J. Sejnowski, "Utilizing Deep Learning Towards Multi-Modal Bio-Sensing and Vision-Based Affective Computing," *IEEE Trans Affect Comput*, vol. 13, no. 1, pp. 96–107, 2022, doi: 10.1109/TAFFC.2019.2916015.

[36] J. A. Miranda-Correa, M. K. Abadi, N. Sebe, and I. Patras, "AMIGOS: A Dataset for Affect, Personality and Mood Research on Individuals and Groups," *IEEE Trans Affect Comput*, vol. 12, no. 2, pp. 479–493, Apr. 2021, doi: 10.1109/TAFFC.2018.2884461.

[37] S. Katsigiannis and N. Ramzan, "DREAMER: A Database for Emotion Recognition Through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices," *IEEE J Biomed Health Inform*, vol. 22, no. 1, pp. 98–107, Jan. 2018, doi: 10.1109/JBHI.2017.2688239.

[38] A. Goshvarpour, A. Abbasi, and A. Goshvarpour, "An accurate emotion recognition system using ECG and GSR signals and matching pursuit method," *Biomed J*, vol. 40, no. 6, pp. 355–368, Dec. 2017, doi: 10.1016/J.BJ.2017.11.001.

[39] R. Subramanian, J. Wache, M. K. Abadi, R. L. Vieriu, S. Winkler, and N. Sebe, "Ascertain: Emotion and personality recognition using commercial sensors," *IEEE Trans Affect Comput*, vol. 9, no. 2, pp. 147–160, Apr. 2018, doi: 10.1109/TAFFC.2016.2625250.

[40] M. Naji, M. Firoozabadi, and P. Azadfallah, "Classification of Music-Induced Emotions Based on Information Fusion of Forehead Biosignals and Electrocardiogram," *Cognit Comput*, vol. 6, no. 2, pp. 241–252, Nov. 2014, doi: 10.1007/S12559-013-9239-7/TABLES/5.

[41] F. Agrafioti, D. Hatzinakos, and A. K. Anderson, "ECG pattern analysis for emotion detection," *IEEE Trans Affect Comput*, vol. 3, no. 1, pp. 102–115, Jan. 2012, doi: 10.1109/T-AFFC.2011.28.

[42] Silicon Labs, "Which ARM Cortex Core Is Right for Your Application: A, R or M?" Accessed: Mar. 20, 2024. [Online]. Available: https://www.silabs.com/documents/public/white-papers/Which-ARM-Cortex-Core-Is-Right-for-Your-Application.pdf

[43] H. Han and J. Siebert, "TinyML: A Systematic Review and Synthesis of Existing Research," *4th International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2022 - Proceedings*, pp. 269–274, 2022, doi: 10.1109/ICAIIC54071.2022.9722636.

[44] "STM32MP157F - MPU with Arm Dual Cortex-A7 800 MHz, Arm Cortex-M4 real-time coprocessor, 3D GPU, TFT/MIPI DSI displays, FD-CAN, Secure boot and Cryptography - STMicroelectronics." Accessed: Mar. 24, 2024. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32mp157f.html

[45] S. M. Noor and E. John, "Performance and Energy Evaluation of ARM Cortex Variants for Smart Cardiac Pacemaker Application," 2016.

[46] L. Wulfert *et al.*, "AIfES: A Next-Generation Edge AI Framework," *IEEE Trans Pattern Anal Mach Intell*, 2024, doi: 10.1109/TPAMI.2024.3355495.

[47] L. Lamberti, L. Bompani, V. J. Kartsch, M. Rusci, D. Palossi, and L. Benini, "Bio-inspired Autonomous Exploration Policies with CNN-based Object Detection on Nano-drones," *Proceedings -Design, Automation and Test in Europe, DATE*, vol. 2023-April, 2023, doi: 10.23919/DATE56975.2023.10137154.

[48] J. Gava *et al.*, "A Lightweight Mitigation Technique for Resource- Constrained Devices Executing DNN Inference Models Under Neutron Radiation," *IEEE Trans Nucl Sci*, vol. 70, no. 8, pp. 1625–1633, Aug. 2023, doi: 10.1109/TNS.2023.3262448.

[49] I. L. Orasan, A. I. Bublea, and C. D. Caleanu, "Deep Learning-Based Eye Gaze Estimation for Automotive Applications Using Knowledge Distillation," *IEEE Access*, vol. 11, pp. 120741–120753, 2023, doi: 10.1109/ACCESS.2023.3325134.

[50] S. Kim, S. Chon, J. K. Kim, J. Kim, Y. Gil, and S. Jung, “Lightweight Convolutional Neural Network for Real-Time Arrhythmia Classification on Low-Power Wearable Electrocardiograph,” *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2022-July, pp. 1915–1918, 2022, doi: 10.1109/EMBC48229.2022.9871156.

[51] S. Qin, J. Zhang, H. Shen, and Y. Wang, “Arm Movements Recognition by Implementing CNN on Microcontrollers,” *2021 9th International Conference on Control, Mechatronics and Automation, ICCMA 2021*, pp. 171–176, 2021, doi: 10.1109/ICCMA54375.2021.9646200.

[52] F. Alongi, N. Ghielmetti, D. Pau, F. Terraneo, and W. Fornaciari, “Tiny Neural Networks for Environmental Predictions: An Integrated Approach with Miosix,” *Proceedings - 2020 IEEE International Conference on Smart Computing, SMARTCOMP 2020*, pp. 350–355, Sep. 2020, doi: 10.1109/SMARTCOMP50058.2020.00076.

[53] A. Faraone and R. Delgado-Gonzalo, “Convolutional-Recurrent Neural Networks on Low-Power Wearable Platforms for Cardiac Arrhythmia Detection,” *Proceedings - 2020 IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2020*, pp. 153–157, Aug. 2020, doi: 10.1109/AICAS48895.2020.9073950.

[54] A. A. Jordan, A. Pegatoquet, A. Castagnetti, J. Raybaut, and P. Le Coz, “Deep Learning for Eye Blink Detection Implemented at the Edge,” *IEEE Embed Syst Lett*, 2020, doi: 10.1109/LES.2020.3029313.

[55] F. De Vita, G. Nocera, D. Bruneo, V. Tomaselli, D. Giacalone, and S. K. Das, “Quantitative Analysis of Deep Leaf: A Plant Disease Detector on the Smart Edge,” *Proceedings - 2020 IEEE International Conference on Smart Computing, SMARTCOMP 2020*, pp. 49–56, Sep. 2020, doi: 10.1109/SMARTCOMP50058.2020.00027.

[56] S. Akhtari, F. Pickhardt, D. Pau, A. Di Pietro, and G. Tomarchio, “Intelligent Embedded Load Detection at the Edge on Industry 4.0 Powertrains Applications,” *5th International Forum on Research and Technologies for Society and Industry: Innovation to Shape the Future, RTSI 2019 - Proceedings*, pp. 427–430, Sep. 2019, doi: 10.1109/RTSI.2019.8895598.

[57] E. Torti *et al.*, “Embedded real-time fall detection with deep learning on wearable devices,” *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018*, pp. 405–412, Oct. 2018, doi: 10.1109/DSD.2018.00075.

[58] Mike Cadogan, “ECG Lead positioning • LITFL • ECG Library Basics.” Accessed: Mar. 24, 2024. [Online]. Available: https://litfl.com/ecg-lead-positioning/

[59] “The ECG leads: Electrodes, limb leads, chest (precordial) leads and the 12-Lead ECG – Cardiovascular Education.” Accessed: Mar. 24, 2024. [Online]. Available: https://ecgwaves.com/topic/ekg-ecg-leads-electrodes-systems-limb-chest-precordial/

[60] M. Chhabra and M. Kalsi, “Real Time ECG monitoring system based on Internet of Things (IoT),” *International Journal of Scientific and Research Publications*, vol. 7, no. 8, 2017.

[61] M. N. Teferra, D. A. Hobbs, R. A. Clark, and K. J. Reynolds, “Preliminary Analysis of a Wireless and Wearable Electronic-Textile EASI-Based Electrocardiogram,” *Front Cardiovasc Med*, vol. 8, p. 806726, Dec. 2021, doi: 10.3389/FCVM.2021.806726/BIBTEX.

[62] H. Xu *et al.*, “Assessing Electrocardiogram and Respiratory Signal Quality of a Wearable Device (SensEcho): Semisupervised Machine Learning-Based Validation Study.,” *JMIR Mhealth Uhealth*, vol. 9, no. 8, pp. e25415–e25415, Aug. 2021, doi: 10.2196/25415.

[63] Y. Xuan *et al.*, “Wireless, minimized, stretchable, and breathable electrocardiogram sensor system,” *Appl Phys Rev*, vol. 9, no. 1, Mar. 2022, doi: 10.1063/5.0082863.
[64] N. Huda, S. Khan, R. Abid, S. B. Shuvo, M. M. Labib, and T. Hasan, “A Low-cost, Low-energy Wearable ECG System with Cloud-Based Arrhythmia Detection,” *medRxiv*, p. 2020.08.30.20184770, Sep. 2020, doi: 10.1101/2020.08.30.20184770.
[65] “Unpack the STM32MP157x-DK2 board - stm32mpu.” Accessed: Mar. 02, 2024. [Online]. Available: https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Let%27s_start/Unpack_the_STM32MP157x-DK2_board
[66] “STM32MP157x-DKx - hardware description - stm32mpu.” Accessed: Mar. 02, 2024. [Online]. Available: https://wiki.stmicroelectronics.cn/stm32mpu/wiki/STM32MP157x-DKx_-_hardware_description#Boot_related_switches
[67] “Ks0261 keyestudio AD8232 ECG Measurement Heart Monitor Sensor Module - Keyestudio Wiki.” Accessed: Mar. 02, 2024. [Online]. Available: https://wiki.keyestudio.com/Ks0261_keyestudio_AD8232_ECG_Measurement_Heart_Monitor_Sensor_Module
[68] “um2637-discovery-kits-with-increasedfrequency-800-mhz-stm32mp157-mpus-stmicroelectronics”, Accessed: Mar. 02, 2024. [Online]. Available: https://www.st.com/resource/en/user_manual/um2637-discovery-kits-with-increasedfrequency-800-mhz-stm32mp157-mpus-stmicroelectronics.pdf
[69] “WESAD (Wearable Stress and Affect Detection) - UCI Machine Learning Repository.” Accessed: Mar. 05, 2024. [Online]. Available: https://archive.ics.uci.edu/dataset/465/wesad+wearable+stress+and+affect+detection
[70] V. Dham, K. Rai, and U. Soni, “Mental Stress Detection Using Artificial Intelligence Models,” in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1950/1/012047.
[71] Edouard Castets, “Stress Detection ECG.” Accessed: Apr. 21, 2024. [Online]. Available: https://github.com/Edouard99/Stress_Detection_ECG?tab=readme-ov-file
[72] “STM32MP15 ADC internal peripheral - stm32mpu.” Accessed: Mar. 04, 2024. [Online]. Available: https://wiki.st.com/stm32mpu/wiki/STM32MP15_ADC_internal_peripheral
[73] B. Mehlig, *Machine Learning with Neural Networks*. 2021. doi: 10.1017/9781108860604.
[74] “text, data and bss: Code and Data Size Explained | MCU on Eclipse.” Accessed: Mar. 30, 2024. [Online]. Available: https://mcuoneclipse.com/2013/04/14/text-data-and-bss-code-and-data-size-explained/

Appendices

List of Abbreviations

Abbreviations	Definition
1D-CNN	1-Dimension Convolutional Neural Network
ACC	Acceleration
Ach	Acetylcholine
ADC	Analog-to-Digital Converters
ADM	Affective Dimensional Models
AI	Artificial Intelligence
ANN	Artificial Neural Network
ANS	Autonomic Nervous System
BVP	Blood Volume Pulse
CMSIS	Common Microcontroller Software Interface Standard
CNN	Convolutional Neural Network
COM	Communication
CPU	Central Processing Unit
C-RNN	Convolution-Recurrent Neural Network
CVD	Cardiovascular Disease
DEM	Discrete Emotional Models
DMA	Direct Memory Address
DNN	Deep Neural Network
DSP	Digital Signal Processing
DT	Decision Tree
ECG	Electrocardiogram
EDA	Exploratory Data Analysis
EEG	Electroencephalogram
FC	Fully Connected
FCN	Fully Convolution Network
FCNN	Fully Connected Neural Network
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GSR	Galvanic Skin Response
HIRARC	Hazard Identification, Risk Assessment and Risk Control
IBI	Inter-Beat Interval
IC	Integrated Circuit
IDE	Integrated Development Environment
KNN	K-Nearest Neighbors
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
MCU	Microcontroller Units
MEG	Magnetoencephalography
MPU	Memory Protection Unit

NB	Naive Bayes
PANAS	Positive and Negative Affect Schedule
PCB	Printed Circuit Board
PNS	Parasympathetic Nervous System
Q-RNN	Quantum-Recurrent Neural Network
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RF	Random Forest
RIP	Respiratory Inductive Plethysmography
RNN	Recurrent Neural Network
RSP	Respiration
RX	Receive
SA	Sinoatrial
SAM	Self-Assessment Manikin
SNS	Sympathetic Nervous System
SOTA	State-Of-The-Art
SRAM	Static Random-Access Memory
SSSQ	Short Stress State Questionnaire
STAI	State-Trait Anxiety Inventory
SVM	Support Vector Machine
TEMP	Temperature
TSST	Trier Social Stress Test
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter
WESAD	Wearable Stress and Affect Detection

Project Proforma

Project Review Proforma – Review number: 1 Date: 30/11/2023

Summarised Planned State of Project:	**Actual Progress Since Last Review**
Literature Review: 1. Research covers emotion recognition and heart disease prediction with ECG. 2. Research covers software tools, models, and architecture for edge computing. Setting up Yocto OS and loading the starter pack: 1. Research and learning function of STM32MP157F DK2 libraries and software. 2. Installing the starter pack and testing the capability of the edge computer. Setting up and coding ECG sensor: 1. Connecting ECG sensor on edge computer. 2. Coding software to send serial data onto the serial plotter.	Literature Review: 1. Research was done according to plan. Specific literature reviews were carried out on review papers on AI models. Setting up Yocto OS and loading the starter pack: 1. Basic features of edge computer are tested and demoed to supervisor and moderator. 2. The task was performed according to plan. Setting up and coding ECG sensor: 1. The task was performed according to plan and is up to schedule. 2. UART, GPIOs, and 16-bit ADC are configured, and engineering mode is booted. 3. Data of ECG can be plotted on a serial monitor with a 115200 baud rate.

Next Steps

1. Based on my Gantt chart planning, I am slightly ahead of my schedule. Hence, an optional task can be performed which is stated in the Gantt Chart – designing wearables for the solution.
2. Focusing on STM32 model zoo and journals to develop a TensorFlow Lite model for emotion classification prediction.
3. Develop a script to display output.

Supervisor Feedback

The supervisor advised simplifying the project's overall complexity by converting several tasks into optional "stretch" goals. Emphasis was suggested to be placed on developing the emotion classification model. Furthermore, attention was directed towards exploring the STM32 model zoo and constructing a model based on literature from reputable journals. This feedback prioritises the development of a robust emotion classification model and leveraging existing resources for model creation.

Figure 84. Project Proforma 1.

Project Review Proforma – Review number: 2 Date: 5/2/2024

Summarised Planned State of Project:	**Actual Progress Since Last Review**
Design Solution/Application: 1. Designing wearables that can be integrated with emotion classification and heart disease prediction models. (Optional) Development of AI Model: 1. Setting up an environment to deploy a heart disease detection and prediction model. 2. Coding program to develop emotion recognition model. 3. Coding program to develop heart disease prediction model. 4. Coding program to connect sensor with AI model. Deployment of Model and Sensor into Edge and Cloud Computer: 1. Deploying emotion recognition model developed onto edge computer.	Design Solution/Application: 1. The wearables design was done according to plan. Development of AI Model: 1. Environment and libraries for edge computer and AI model have been set up. 2. The dataset was downloaded, and a customized dataset was created. 3. TensorFlow model was trained. 4. TensorFlow Lite model was created and uploaded onto STM32Cube IDE. 5. Debug and solve flash memory insufficient issues. 6. A program that connects sensors with an AI model has been created. 7. The task was performed according to plan. Deployment of Model and Sensor into Edge and Cloud Computer: 1. Debugging deployment of AI models and sensor code. 2. The task is still ongoing.
Next Steps 1. Start writing the thesis draft. 2. Complete the first cycle of obtaining real-time inference using values from sensors. 3. Debugging firmware deployment on STM32CubeIDE. **Supervisor Feedback** The supervisor recommended commencing the drafting of the thesis and focusing on completing the initial phase of acquiring real-time emotion classification inference utilizing sensor data.	

Figure 85. Project Proforma 2.

Project Review Proforma – Review number: 3 Date: 6/3/2024

Summarised Planned State of Project:	**Actual Progress Since Last Review**
Thesis Draft: 1. Start planning and write the outline of the thesis. 2. Complete introduction and literature review. Complete the first cycle of obtaining real-time inference using values from sensors: 1. Successfully deploy the model onto hardware. 2. Determine the limitation of hardware. 3. Perform analysis on the model inference time, accuracy and size. Debugging firmware deployment on STM32CubeIDE: 1. Research and debug the problem faced and the error prompted on the STM32CubeIDE.	Thesis Draft: 1. The thesis draft planning was done according to plan. Complete the first cycle of obtaining real-time inference using values from sensors: 1. Real-time model inference was obtained and printed on a serial monitor. 2. Tackle model overfitting issue. 3. The task was performed according to plan. Debugging firmware deployment on STM32CubeIDE: 1. Debugging deployment of AI models and sensor code. 2. The SRAM issue was determined and the model was retrained to reduce the complexity. 3. The task was performed according to plan.

Next Steps

1. Continue writing the thesis draft.
2. Perform experiments to determine the hardware limitations of STM32MP157F-DK2.
3. Research CNN model architecture.

Supervisor Feedback

The supervisor suggested conducting experiments on the ANN architecture using varying numbers of inputs to ascertain the hardware limitations of the STM32MP157F-DK2. Additionally, it was recommended to explore CNN model architectures through research and attempt deployment if feasible. The suggestion to delve into CNN model architectures underscores the potential for enhancing the project's capabilities through the adoption of alternative neural network structures, potentially improving the performance of the model.

Figure 86. Project Proforma 3.

Project Review Proforma – Review number: 4 Date: 20/3/2024

Summarised Planned State of Project:	**Actual Progress Since Last Review**
Thesis Draft: 1. Complete methodology for thesis draft. Perform experiments on ANN to determine the hardware limitations of STM32MP157F-DK2: 1. Test various numbers of inputs and determine if the model can be compiled on the hardware. 2. Record the accuracy, compatibility and size of the model. Research CNN model Architecture: 1. Research and write a 1D-CNN architecture and determine its compatibility with the hardware.	Thesis Draft: 1. The thesis draft methodology was done according to plan. Perform experiments on ANN to determine the hardware limitations of STM32MP157F-DK2: 1. Highest accuracy model was determined. 2. Hardware limitations were determined. 3. The overfitting issue was noted and tackled with a new model architecture. 4. The task was performed according to plan. Research CNN model Architecture: 1. A 1D-CNN script was developed and trained. 2. The model developed is not compatible with the hardware due to hardware constraints. 3. Fine-tuning of the model and experiments needs to be carried out. 4. The task is still ongoing.
Next Steps 1. Continue writing the thesis draft. 2. Perform experiments to deploy 1D-CNN into STM32MP157F-DK2. 3. Fine-tune the model with various hyperparameters to mitigate overfitting. **Supervisor Feedback** The supervisor advised continuing the thesis draft while also conducting experiments aimed at deploying the 1D-CNN model onto the hardware. Furthermore, it was suggested to refine the model by adjusting various hyperparameters to address potential overfitting issues.	

Figure 87. Project Proforma 4.

Project Review Proforma – Review number: 5 Date: 3/4/2024

Summarised Planned State of Project:	**Actual Progress Since Last Review**
Thesis Draft: 1. Submission of thesis draft. Perform experiments to deploy 1D-CNN into STM32MP157F-DK2: 1. Test various architectures and determine if the model can be compiled and built on the hardware. 2. Record the accuracy, compatibility and size of the model. Fine-tune the model with various hyperparameters to mitigate overfitting: 1. Tested various hyperparameters such as batch normalization, dropout, regularization, learning rate, and early stopping to determine if the model improved.	Thesis Draft: 1. The thesis draft was completed and submitted according to plan. Perform experiments to deploy 1D-CNN into STM32MP157F-DK2: 1. Highest accuracy model was determined. 2. Hardware limitations were determined. 3. The 1D-CNN model was deployed into hardware. 4. The task was performed according to plan. Fine-tune the model with various hyperparameters to mitigate overfitting: 1. Fine-tuning of the model was carried out. 2. The overfitting issue is mitigated. 3. The task was performed according to plan.

Next Steps

1. Create a user interface (UI) dashboard to showcase the real-time result of the model deployed into the hardware.
2. Work on the final thesis.
3. Compile all of the documents, scripts, and logbooks to be submitted.

Supervisor Feedback

The supervisor proposed the creation of a UI dashboard to present real-time results generated by the deployed model on the hardware platform. This is to emphasize the importance of developing a user-friendly interface to visualize and interpret the model's outputs in real time. By integrating a UI dashboard into the project, stakeholders and end-users can conveniently access and interpret the emotion classification results, enhancing the project's usability and practicality.

Figure 88. Project Proforma 5.

Gantt Chart

Year 3 Final Year Project
Supervisor: Dr Hermawan Nugroho
Moderator: Prof T. Nandha Kumar
Project Code: HN-BEng-23-01

Student Name: Koay Xian Cong
Student ID: 20418760

Legends: Error Bars for Uncertainty: ↔ Moderator Meeting Week: ▭
Submission Dateline: ✖ Project Milestone: ★

Tasks	Autumn Semester												Study Week		Exam		Sem Break		CNY Break		Spring Semester						HR Break				Study Week
	Sept	Oct					Nov				Dec				Jan				Feb				Mar				Apr				May
	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16	WEEK 17	WEEK 18	WEEK 19	WEEK 20	WEEK 21	WEEK 22	WEEK 23	WEEK 24	WEEK 25	WEEK 26	WEEK 27	WEEK 28	WEEK 29	WEEK 30	WEEK 31	WEEK 32	WEEK 33	WEEK 34	WEEK 35	WEEK 36	WEEK 37
Project Planning																																	
Hazard Identification, Risk Assessment, Risk Control Form (HIRARC) Form																																	
FYP Briefing and Discussion on Project Specification																																	
Hardware Inspection and Sensor Component Research																																	
Software, Tools and Application Research																																	
Finalising Project Objectives and Project Specification																																	
Hardware and Software Research and Purchase Cost Estimation																																	
Purchase Requisition																																	
Lab Space Booking																																	
Obtaining Ethical Approval																																	
Writing Project Outline and Planning Report																																	
Submission of Project Outline and Planning Report																																	
Research and Literature Review																																	
Researching of Common Application of Edge Computing																																	
Literature Review of Emotion Recognition using ECG																																	
Literature Review of Common Heart Disease																																	
Literature Review of Heart Disease Detection using Machine Learning and Deep Learning																																	
Literature Review of Heart Disease Prediction Model																																	
Literature Review of STM32MP157F DK Edge Computing Device and Application																																	
Literature Review of Software Tools, Model and Architecture for Edge Computing																																	
Design Solution/Application																																	
Research and Learning the Function of STM32MP157F DK Edge Computer																																	
Installing Virtual Machine to Set Up STM32 Software to Flash OS into the Edge Computer																																	
Installing STM32 Cube Programmer in Ubuntu OS																																	
Coding Program to Test Functions of Edge Computer																																	
Integrating Heart Monitoring Sensor onto the Edge Computer																																	
Coding Program to Test Sensors and Perform Experiments to Ensure Accurate Measurements of Sensors																																	
Designing Wearables of that can be Integrated with Heart Disease Detection and Prediction Solution (Optional)																																	
Development of AI Model																																	
Setting Up Environment to Deploy Heart Disease Detection and Prediction Model																																	
Coding Program to Develop Emotion Recognition Model																																	
Coding Program to Develop Heart Disease Prediction Model (Optional)																																	
Coding Program to Connect Sensor with AI Model																																	
Coding Program to Display Prediction Output on the LCD Display (Optional)																																	
Developing User Interface to Monitor Heart Disease and Data Collected from Sensor (Optional)																																	
Debugging Code to Solve Bugs Found, Improvement of Code/Model																																	
Deployment of Model and Sensor into Edge and Cloud Computer																																	
Combining both Software and Hardware Developed into Wearables (Optional)																																	
Deploying Heart Disease Prediction Model Developed onto Cloud Computer (Optional)																																	
Deploying Emotion Recognition Model Developed onto Edge Computer																																	
Performing Testing on Real Live User																																	
Obtaining User Experience, Feedback and Model Prediction Result																																	
Performance Improvement and Solving Bugs																																	
Thesis, Documents and Presentation																																	
Writing Sections of Draft Thesis																																	
Draft Typed Thesis Submission																																	
Writing Sections of Final Thesis																																	
Final Thesis, Logbook, Code and Miscellaneous Submission																																	
Preparation of Presentation																																	
Presentation																																	
Return of Project Items																																	

Figure 89. Gantt Chart.