

Development of Embedded AI Applications & Toolchain

Major Project Report
(7EC192)

*Submitted in fulfilment of the requirements
for the degree of*

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Dhruvil Prajapati
(22MECE08)



Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2024

Development of Embedded AI Applications & Toolchain

Major Project Report

(7EC192)

*Submitted in partial fulfilment of the requirements
for the degree of*

Master of Technology in Electronics & Communication Engineering

By

Dhruvil Prajapati
(22 MECE08)

Under the guidance of

External Project Guide:

Mr. Swaraj Badhei

Senior AML Specialist
Infineon Technologies
Bangalore.

Internal Project Guide:

Prof (Dr.) Yogesh Trivedi

Professor, EC Department,
Institute of Technology,
Nirma University, Ahmedabad.



Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382 481

May 2024

Certificate

This is to certify that the major project entitled “**Development of Embedded AI Applications & Toolchain**” submitted by **PRAJAPATI DHRUVIL GAUTAM (Roll No: 22MECE08)**, towards the partial fulfilment of the requirements for the award of degree of Master of Technology in Electronics and Communication Engineering (Embedded Systems) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-II, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

Dr. Yogesh Trivedi
Guide & Professor,
Institute of Technology
Nirma University, Ahmedabad

Dr Nagendra Gajjar
PG Coordinator & Professor
Institute of Technology
Nirma University, Ahmedabad

Dr. Usha Mehta
Professor & HOD,
ECE Department
Institute of Technology
Nirma University, Ahmedabad

Director School of Technology,
Institute of Technology
Nirma University Ahmedabad

Statement of Originality

I, Prajapati Dhruvil Gautam, Roll. No. 22MECE08, give undertaking that the Major Project entitled “Development of Embedded AI Applications & Toolchain” submitted by me, towards the partial fulfilment of the requirements for the degree of Master of Technology in Electronics & Communication Engineering (Embedded Systems) of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:22/05/2024

Place: Ahmedabad

Endorsed by
Dr. Yogesh Trivedi

(Signature of Guide)

Acknowledgements

I would like to express my gratitude and sincere thanks to **Prof (Dr.) N.P. Gajjar**, PG Coordinator of M. Tech Embedded Systems and **Prof (Dr.) Sachin Gajjar** for guidelines during the review process.

I take this opportunity to express my profound gratitude and deep regards to **Prof (Dr.) Yogesh Trivedi**, guide of my internship project for his exemplary guidance, monitoring, and constant encouragement.

I would also like to thank **Mr. Swaraj Badhei**, external guide of my internship project from **Infineon Technologies Semiconductor Manufacturing Company**, for guidance, mentoring, and encouragement regarding the project.

- Dhruvil Prajapati
22MECE08

Infineon Technologies Company Profile



Infineon Technologies is a globally renowned semiconductor company at the forefront of technological innovation. Established in 1999, Infineon has emerged as a key player in the semiconductor industry, specializing in the development and manufacturing of cutting-edge solutions for a wide range of applications. The company's commitment to excellence and forward-thinking has positioned it as a leader in automotive, industrial, and power electronics, as well as security and chip card solutions.

Semiconductors are crucial to solve the energy challenges of our time and shape the digital transformation. This is why Infineon is committed to actively driving decarbonization and digitalization. As a global semiconductor leader in power systems and IoT, we enable game-changing solutions for green and efficient energy, clean and safe mobility, as well as smart and secure IoT. We make life easier, safer, and greener. Together with our customers and partners. For a better tomorrow.

Infineon's semiconductor products play a pivotal role in enabling advancements in smart mobility, renewable energy, and digitalization. With a focus on addressing the world's most pressing challenges, the company's research and development efforts have led to groundbreaking solutions in power management, connectivity, and security.

Decarbonization the world must reduce carbon emissions and use energy much more efficiently to secure quality of life for future generations. With our power system solutions, we are a key enabler in the move to harness renewable energy resources and deliver energy-efficient solutions along the entire electrical energy chain. Together with our customers and partners, we make “more out of less” to actively shape a greener future.

Digitalization digital transformation is changing the way we live, work, produce, and consume. With our smart IoT devices and systems solutions, we link the real and the digital world, and play a key role in unleashing the full potential of digitalization. Our products and solutions enable a net ecological benefit, equal to the average annual CO₂ emissions from electricity consumption of more than 179 million people living in Europe. Our products and solutions cover a broad application spectrum including consumer electronics, IoT devices, cloud security, IT equipment, home appliances, connected cars, credit and debit cards, future payments, electronic passports, ID cards, and more. In close alignment with our ecosystem partners, we create standard-setting solutions that inspire lasting trust among customers.

Infineon's business segments and target applications

Automotive (ATV): The ATV division is shaping the future of mobility by enabling clean, safe, and smart cars. Its product and solution offering is powering the decarbonization and digitalization of vehicles.

Green Industrial Power (GIP): The GIP division delivers leading semiconductor solutions for the smart, green, and efficient conversion of electrical energy, covering all steps in the energy chain from generation through transmission to storage and consumption.

Power & Sensor Systems (PSS): The PSS division powers decarbonization and digitalization with a wide range of energy-efficient and digital solutions.

Connected Secure Systems (CSS): The CSS division is driving robust connections, reliable computing, and seamless security for a digitalized, decarbonized world.

As a hub for innovation, Infineon Technologies has consistently pushed the boundaries of semiconductor technology. Their commitment to sustainability is evident through initiatives that prioritize energy efficiency and eco-friendly practices. Collaborative partnerships with industry leaders and a global presence further underscore Infineon's influence in shaping the future of technology.

Abstract

The relentless advancement of Artificial Intelligence (AI) is revolutionizing numerous industries, notably the embedded systems domain. This report sheds light on the development and deployment of cutting-edge, embedded AI applications and an efficient toolchain, focusing on gesture recognition using radar sensor, developing machine learning models, and r&d on neural architecture search algorithm.

The first major achievement was the development and deployment of a radar gesture classification model on the Infineon XENSIV™ kit. This innovative model could identify four distinct gestures in real-time, serving as a testament to the power of embedded AI in enhancing interaction between humans and machines. With the ability to recognize and interpret specific gestures, the model provided a foundation for creating more intuitive and interactive user interfaces for a wide range of applications. In addition to radar gesture recognition, the internship involved the development of various machine learning models. These models were designed to harness the power of AI for solving complex tasks, demonstrating the capacity of machine learning in improving the efficiency of embedded systems. This work further highlighted the potential of machine learning models to advance data-driven solutions in the field of embedded AI. Lastly, the report delves into the research, exploration, and development of "Neural Architecture Search". This groundbreaking work involved the design of neural networks that are aware of the hardware they run on, potentially revolutionizing the way machine learning models are developed and deployed in real-world settings. By taking into consideration the hardware constraints during the design of neural networks, this research could pave the way for more efficient and practical AI solutions.

In summary, the work chronicled in this report underscores the transformative potential of AI in the realm of embedded systems. It offers valuable insights and contributions in radar gesture recognition, machine learning models, and hardware-aware neural architecture search, showcasing a promising future for the integration of AI into everyday devices.

Index

Declaration		iii
Disclaimer		iv
Certificate		v
Acknowledgements		vii
Abstract		xiv
Abbreviation Notation and Nomenclature		xv
1	Introduction	1
	1.1 Motivation	1
	1.2 Problem Statement	1
	1.3 Scope of work	2
	1.4 Outline of Thesis	2
2	Literature Review	3
	2.1 Overview	3
	2.2 M. L Architecture in Embedded Systems	4
	2.3 Optimization Techniques	4
	2.4 Framework and Tools	5
3	Journey into Embedded AI Development	6
	3.1 Introduction to Embedded AI	6
	3.2 Advantages of Embedded AI	7
	3.3 Hardware and Software Selection	8
	3.4 Software Consideration	10
	3.5 Challenges in Embedded AI	11

	3.6 Future of Embedded AI	12
4	Work done at Infineon	14
	4.1 Research & Exploration	14
	4.2 ML Model Development	15
	4.3 Exploration ML Optimization Techniques	17
	4.4 Voice detection at edge	19
	4.5 Radar Gesture Classification	21
5	Research & development of Neural Architecture search algorithm	24
	5.1 Introduction to Neural Architecture Search	24
	5.2 The Need for HW-Aware NAS	25
	5.3 Neural Network / Hardware Co-optimization	26
	5.4 Architecture of HW-NAS	27
	5.5 Aging Evolution in NAS	28
	5.6 Evaluation Metrics for NAS	28
	5.7 Advantages of NAS	29
	5.8 Future Prospect of NAS	30
6	Conclusion	31
	References	32

List of Figures

Fig 2.1	Modus Technical Flow	5
Fig 3.1	XMC800 Board v2	9
Fig 3.2	CY8CKIT-064BOS2-4343W PSoC 64 kit	9
Fig 4.1	Development Flow	15
Fig 4.2	Speech Recognition Model	16
Fig 4.3	Pruning Network	17
Fig 4.4	Quantized Network	18
Fig 4.5	Weight Sharing Matrix	18
Fig 4.6	CYCPROTO-062-4343W PSoC 6	19
Fig 4.7	Voice recognition flow	19
Fig 4.8	Technical flow diagram	20
Fig 4.9	Results of Voice Recognition model	21
Fig 4.10	FMCW 60 Ghz Working	21
Fig 4.11	Gesture Classification Model	21
Fig 4.12	Design Flow of gesture classification	23
Fig 5.1	The General Framework for NAS	25
Fig 5.2	NAS Architecture	27

List of Tables

Table 3.1	ML tools and Description	11
-----------	--------------------------	----

Abbreviation Notation and Nomenclature

AI	Artificial Intelligence
ML	Machine Learning
IoT	Internet of Things
Edge AI	Edge Artificial Intelligence
RTOS	Real-Time Operating System
CMSIS	Cortex Microcontroller Software Interface Standard
DSP	Digital Signal Processor
CNN	Convolutional Neural Network
MCU	Microcontroller Unit
FPGA	Field-Programmable Gate Array
NN	Neural Network
DNN	Deep Neural Network
FMCW	Frequency Modulated Continuous Wave
MTB	Modus Toolbox
NAS	Neural Architecture Search

Chapter 1

Introduction

1.1 Motivation

Imagine a typical day in a modern society using artificial intelligence technology. People wake up in the morning, they use their smartwatches to check if they slept well. Then I asked the voice assistant how long it took to choose the outfit. They activate their smartphones with Face ID scanners and check their daily schedules. Finally, he arrives at the office in a car on autopilot. Artificial intelligence (AI) has become ubiquitous... An internship as an applied machine learning engineer at Infineon Technologies offers the opportunity to translate theoretical knowledge into tangible real-world applications. The prospect of delving into the dynamic intersection of embedded systems and artificial intelligence, honing my skills and contributing to innovative projects further increased my enthusiasm. The philosophy of continuous learning and problem solving ingrained in the corporate environment aligns perfectly with my own values, motivating me to view challenges as opportunities for growth.

1.2 Problem Statement

The convergence of embedded systems, artificial intelligence (AI), and sensor technologies has introduced unprecedented opportunities for innovation, particularly in the realm of Infineon microcontrollers (MCUs). However, this integration poses a series of challenges that demand meticulous attention. The development of embedded AI applications using Infineon MCUs and sensors necessitates a seamless fusion of hardware and software, demanding a delicate balance between computational efficiency and model accuracy.

Furthermore, the optimization of machine learning (ML) models in this embedded environment requires the establishment of a robust toolchain. Challenges lie in tailoring the ML model to the constraints of Infineon MCUs, maximizing performance while minimizing resource utilization. Ensuring real-time responsiveness, energy efficiency, and effective utilization of limited resources emerges as a critical concern.

The problem statement aims to address the intricate interplay between embedded AI application development on Infineon MCUs, sensor integration, and the establishment of an efficient ML model optimization toolchain. Solving these challenges is imperative for unlocking the full potential of embedded AI systems, ensuring their seamless integration into a variety of applications while meeting the stringent resource constraints imposed by the embedded environment. The report will delve into these challenges, proposing innovative solutions and contributing to the advancement of embedded AI technology.

1.3 Scope of Work

The scope of this project encompasses a comprehensive exploration and development of embedded AI applications utilizing Infineon microcontrollers (MCUs) and sensor technologies. The primary focus is developing a Radar Gesture Classification ML model capable of accurately classifying radar-generated gesture patterns using a NN architecture developed and optimized for real-time performance and accuracy, ensuring reliable gesture recognition across diverse scenarios. Deploying the trained NN model on FMCW 60 GHz RADAR SENSOR addresses challenges related to sensor data synchronization, preprocessing, and real-time model inference on the XENSIV platform.

1.4 Outline of Thesis

A comprehensive exploration of the development and implementation of a machine learning model for radar gesture classification, specifically designed for integration with the XENSIV 60GHz radar sensor suite. With a focus on model efficiency and real-time applicability, the project not only addresses the nuances of radar-based gesture recognition but also delves into the complexities of optimization, modelling and machine learning for deployment in resource-limited environments. The report continues with an introduction, briefly presenting the phase, establishing the context, motivation and overall objectives of the project. From there, the literature review will examine existing research, shedding light on radar gesture classification and the technological landscape surrounding the XENSIV suite. Key insights from previous solutions will guide subsequent development efforts. The core of the report focuses on machine learning model development, providing insight into the chosen architecture, the complexity of the training data, and the validation process with a comprehensive understanding of neural network construction, ensuring transparency in the decision-making process. The actual implementation takes centre stage in the “Integration with the XENSIV Toolkit” section, where the report examines the steps taken to seamlessly integrate the machine learning model with the XENSIV Toolkit.

The important aspect of the report is in the “ML Model Optimization Toolchain” section, which explores the development of flexible toolchains capable of optimizing machine learning models for deployment on platforms with limited resources.

Overall, the report serves as a valuable resource, providing not only technical information but also practical considerations for future efforts in similar fields.

Chapter 2

Literature review

2.1 Overview

The problem of excessive energy consumption and poor compatibility while deploying artificial intelligence models and networks on embedded devices has been manageable due to advancements in artificial intelligence algorithms and models and embedded device support. This report presents three approaches to implementing artificial intelligence technologies on embedded devices in response to these issues: neural network compression, acceleration techniques for embedded devices, artificial intelligence models and algorithms on hardware with limited resources, and existing embedded AI application models. This report finishes with future directions for embedded AI and a summary of the topic, after comparing pertinent literature and highlighting the study's merits and drawbacks.

Embedded Intelligence is currently building a foundation that includes AI support delivery platforms such design of hardware accelerators for neural networks, network structure design, network model optimization including quantization, pruning, and weight reduction methods, and improvement of the underlying hardware algorithms [2]. All the above technologies contribute to the deployment of AI to resource-constrained devices, but further development is needed in the following areas: Efficient algorithms and lightweight models, optimization of hardware acceleration techniques., optimization of deployment methods and compatibility.

Advances in artificial intelligence algorithms and models and support for embedded devices have made it possible to overcome the problems of high-power consumption and poor compatibility when deploying artificial intelligence models and networks to embedded devices. In response to these issues, present in artificial intelligence algorithms and models for embedded devices on resource-constrained hardware, including methods for accelerating embedded devices, neural network compression, and current application models. In this report present aspects of methods and applications for deploying intelligent technologies on Embedded devices [17]. Application such as Voice recognition and Radar Gesture Classification such are developed and deployed on embedded devices are mentioned alongside in report, In order to know the development status of radar gesture recognition and predict the future development trend, the research and development of gesture recognition based on radar technology has been studied in recent years. Classification algorithms for gesture recognition are summarized, focusing on key techniques such as dynamic recognition of gesture information, preprocessing of gesture echo signals, and feature extraction in radar gesture recognition technology. System performance is analysed and evaluated. Clarify the issues to be solved in the direction of research and predict the direction of future research. This result shows that Radar's gesture recognition technology has made a significant advance in human-computer interaction applications. With the deepening of related research, the gesture recognition system based on radar technology will develop towards intelligence.

2.2 ML Architecture used in Embedded Systems:

Below are some pre-trained ML architecture

- **MobileNet:** MobileNet is designed for mobile and embedded image processing applications and provides a lightweight architecture with depth-separable convolutions [4]. It balances accuracy and computational efficiency, making it suitable for use on devices with limited resources.
- **TensorFlow Lite Micro (TFLite Micro) Models:** TensorFlow Lite Micro includes a variety of optimized models suitable for embedded systems.[21] These models are designed to run efficiently on microcontrollers and other edge devices for everything from image classification to keyword detection.
- **TinyML Models:** TinyML refers to the deployment of very lightweight machine learning models on resource-constrained devices [6]. These models are often based on simplified architectures, such as fully connected neural networks or support vector machines and are tailored to minimize storage and computational requirements.
- **Quantized Neural Network (QNN):** A quantized neural network uses low-precision representations for weights and activations to minimize memory usage [3]. This category includes binary neural networks (BNNs) and ternary weighting networks that optimize models for use on devices with limited memory.
- **Lightweight Convolutional Neural Networks (LCNN):** LCNNs are designed with a focus on lightweight and efficient convolution operations, making them suitable for real-time image processing on embedded devices [3].

2.3 Optimization Techniques

Machine learning optimization is the process of adjusting hyperparameters in order to minimize the cost function by using one of the optimization techniques. It is important to minimize the cost function because it describes the discrepancy between the true value of the estimated parameter and what the model has predicted.

- **Compact model:** Design a smaller model from scratch that can achieve acceptable performance for the task at hand. This requires a thoughtful design process and selection of model components based on downstream (embedded or mobile) performance requirements [3].
- **Tensor Decomposition:** Simplify large tensors or matrices into smaller matrices or tensors to reduce model storage space and computational cycles. One way to accomplish this is to cluster the model parameters. Clustering groups the weights of each layer into a predefined number of clusters and provides the centroid of each cluster for calculation.
- **Data quantization:** Reduces the precision of model parameters [19]. This is one of the most common and simple ways to optimize machine learning models for use on edge devices [3].
- **Network sparsification:** reduce the number of connections/neurons in the network to obtain a smaller and more sufficient model. Pruning allows you to make your network thinner. Cleaning removes parameters in the model that have little impact on performance. An important part of pruning is choosing which parameters to remove. A simple heuristic is to remove parameters whose values are close to zero.

Combining these techniques can significantly optimize machine learning models and make them suitable for use on edge devices. But that's easier said than done. Optimizations typically reduce model performance, but the performance degradation varies by model.

In rare cases, optimization may improve model performance. Therefore, it is up to the system developer to decide which optimization technique to use and how much to compromise model performance [19].

2.4 Framework and Tools

The development of embedded AI has led to a variety of frameworks and tools that facilitate the use of machine learning models in resource-constrained embedded systems.

Here are some notable

Frameworks:

- **TensorFlow Lite:** Developed by Google, TensorFlow Lite is a popular software specifically designed for mobile and embedded devices. A lightweight version of the TensorFlow framework. Provides model transformation, quantization, and optimization tools for efficient deployment to edge devices [21].
- **PyTorch Mobile:** PyTorch is known for its flexibility and dynamic computational graphs, and has a mobile version designed for deploying models to devices with limited resources. PyTorch Mobile supports model transformation and optimization techniques to ensure efficient execution on edge devices [22].
- **ONNX (Open Neural Network Exchange):** ONNX is an open-source format for representing machine learning models. This allows models to be trained in one framework and then transferred and deployed to another, making it a versatile tool for cross-framework compatibility in embedded systems [20].
- **CMSIS-NN:** ARM's CMSIS (Cortex Microcontroller Software Interface Standard) includes CMSIS-NN, a library specifically tailored to optimize neural network implementations on ARM Cortex-M processors. It provides the ability to operate on neural network layers and allows efficient execution on microcontrollers [22].

Tools:

- **Modus Toolbox™:** Software is a modern, extensible development environment supporting a wide range of Infineon microcontroller devices, including PSoC™ Arm® Cortex® Microcontrollers, TRAVEO™ T2G Arm® Cortex® Microcontroller, XMCTM Industrial Microcontrollers, AIROC™ Wi-Fi devices, AIROC™ Bluetooth® devices, and USB-C Power Delivery Microcontrollers. The ML solution in Modus Toolbox also provides a configurator for importing pre-trained machine learning models and generating embedded models (as C code or binary). This generated model can be used in your ML library along with your target device's application code. This tool also allows you to customize selected pre-trained models and evaluate their performance [10].

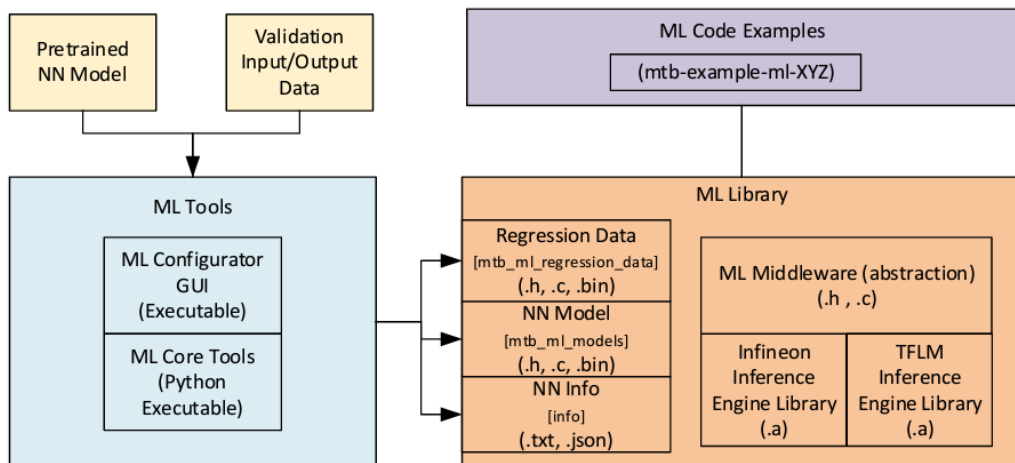


Fig 2.1

- ML Configurator:** The Modu Toolbox™ Machine Learning (ML) tuner is used in ML applications to adapt a pre-trained learning model to the target Infineon platform. The tool accepts a pre-trained ML model and creates an embedded model (like a library) that can be used with application code on the target device. With the Modu Toolbox™ ML Tuner, you can also adapt the selected pre-trained model to the target device with a set of optimization parameters [10].
- Netron:** Netron is a viewer for neural network, deep learning, and machine learning models. Netron supports ONNX, TensorFlow Lite, Core ML, Keras, Caffe, Darknet, MXNet, PaddlePaddle, ncnn, MNN and TensorFlow.

Chapter 3

Journey into Embedded AI Development

3.1 Introduction to Embedded Artificial Intelligence

The advent of artificial intelligence (AI) in recent years has truly revolutionized our industries and our personal lives, offering unprecedented opportunities and capabilities. But while cloud-based processing and cloud AI have been on the rise over the past decade, they have faced issues such as latency, bandwidth limitations, and security and privacy concerns, to name a few. Here, the arrival of the Embedded AI greatly increased the value of his and changed the AI landscape.

This chapter introduces the field of embedded AI and reveals the complexities and innovations that define this emerging field. Embedded AI goes beyond traditional computing paradigms by building intelligence into devices ranging from microcontrollers to specialized hardware the impact will be significant, creating a paradigm shift in the way we perceive and interact with the myriad smart devices around us. We explore the unique challenges and opportunities that characterize the development of AI in embedded systems. From complex hardware considerations to the use of advanced machine learning models, embedded AI requires a delicate balance between computing power and resource limitations. Navigate sensor integration, real-time processing, and security considerations to uncover the complexities that shape the development and deployment of AI at the edge. The real-world applications and case studies that demonstrate the transformative potential of embedded AI across a variety of industries[6]. As we move beyond this, we will also consider optimization strategies beyond simple quantization, such as model compression and pruning, and evaluate the delicate trade-offs between model accuracy and resource efficiency. Essentially, this chapter serves as an entry point into the realm of embedded AI and invites us to witness the convergence of intelligence and embedded systems. There, innovations combine to redefine the functionality of devices that have become an integral part of modern society.

3.2 Advantages of Embedded AI

Following are the advantages of Embedded:

- **Reduced latency:** Embedded AI applications limit the amount of data sent over the wide area network because processing occurs close to the data source rather than in the cloud. Therefore, data processing is faster and Embedded AI application latency is reduced. Additionally, sending and executing instructions from AI applications to the field significantly reduces latency. This is important for several classes of low latency AI applications, such as applications based on industrial robots and automated guided vehicles. Several other video-based applications that need to send data to the cloud can now not only be processed at the edge, but also assess in near real-time what will happen in situations such as: Because it's important, it's now handled at the edge. security case.
- **Real-time performance:** The low latency of Embedded AI applications makes them suitable for implementing features that require real-time performance. For example, machine learning applications that detect events in real time (e.g., fault detection in

production lines, abnormal behaviour detection in security applications) may tolerate delays associated with transmitting and processing connected data.

- **Improved security and data protection:** Embedded AI applications significantly reduce the exposure of data outside the organization that creates or owns the data. This reduces the attack surface and minimizes opportunities for malicious security attacks and data breaches. For this reason, Embedded AI applications tend to be much more secure than those in the cloud [6].

- **Improved privacy controls:** Many AI applications process sensitive data, such as: Data related to security, intellectual property, patients, and other forms of his personal data. Embedded AI deployments create a trusted data management environment. All these applications offer more robust privacy controls than traditional AI applications in the cloud. This is because Edge AI applications limit the amount of data transferred or shared outside of organizations that create or process sensitive data sets.

- **Energy Efficiency:** Cloud Data Transfer and Cloud Data Processing is a highly energy efficient operation. Cloud I/O (input/output) functionality is associated with significant CO2 emissions. Most importantly, Cloud AI is not green overall, as very large amounts of data are typically processed by GPUs (Graphics Processing Units) and TSUs (Tensor Processing Units). Edge AI alleviates environmental performance issues for cloud AI applications. It reduces the number of I/O operations and processes data in an edge device or edge data centre. Therefore, it leads to an improvement in the overall carbon footprint of AI applications.

- **Cost-effective:** Edge AI applications transfer and process much less data than cloud computing applications, saving network bandwidth, and computing resources. Additionally, it consumes less energy than cloud AI applications. This allows edge AI applications to be deployed and operated at significantly lower costs than cloud AI deployments.

- **On-device learning:** Certain Edge AI applications can run within a single device, such as an IoT device or a microcontroller. This enables the development of powerful and intelligent devices such as system-on-chip (SoC) devices. One of the key features of is on-device learning. This is the basis for giving his machine intelligence capabilities that are largely not based on cloud processing.

3.3 Hardware & Software Selection

Hardware Considerations: Hardware Considerations To ensure that the hardware you select meets the requirements of your Embedded AI system, you must consider several factors. A key consideration is processing power, because Embedded AI applications require hardware that can process data quickly and accurately. The processing power and clock speed of hardware play a critical role in achieving the real-time processing that is essential for Embedded AI systems. Another important consideration is the amount and type of memory available on your hardware. Embedded AI applications require sufficient memory to store and process large amounts of data, focusing on the 's fast real-time processing capabilities. There is a race in the hardware industry for the highest TOPS (Terra operations

per second), but this is not a 1-to-1 comparison. Power consumption is also an important consideration, especially for devices with limited power resources, such as IoT devices. Optimizing the hardware for the 's power consumption is important to ensure long-term operation of the device without frequent replacement or recharging of the battery. Technologies such as the use of low-power chips, hardware accelerators, and the and intelligent power management system help achieve the 's low power consumption in Embedded AI systems.

Seamless connectivity is also important for AI systems that require hardware that enables seamless communication with other devices and cloud-based platforms. The hardware must offer various connectivity options such as Wi-Fi, Bluetooth, and mobile networks to enable efficient communication and data processing with your other devices. This connection ensures fast and efficient data processing and analysis, contributing to the overall performance of the Edge AI system.

Some Infineon’s Hardware Compatible to run AI Algorithms

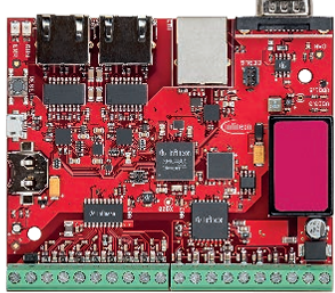
XMC4800 automation board V2	Type	Description	Ordering code
 <p>Ordering code: KIT_XMC48_AUT_BASE_V2</p>	KIT_XMC48_AUT_BASE_V2	The XMC4800 Automation Board V2 utilizes Infineon’s industry leading XMC ARM® Cortex®-M4 microcontroller in combination with Infineon supply, interface/communication and safety products.	KITXMC48AUTBASEV2TOB01
	XMC4800-E196K2048	ARM® Cortex®-M4 microcontroller	XMC4800E196K2048AAXQMA1
	ISO2H823V2.5	24 V 8-channel isolated output	ISO2H823V25XUMA1
	ISO1I813T	24 V 8-channel isolated input	ISO1I813TXUMA1
	SLS 32AIA020A4 USON10	OPTIGA™ Trust E – embedded security solution	SLS32AIA020A4USON10XTMA2
	TLE6250GV33	Infineon CAN transceiver	TLE6250GV33XUMA1
	IFX54441LDV	Infineon voltage regulator	IFX54441LDVXUMA1

Fig 3.1

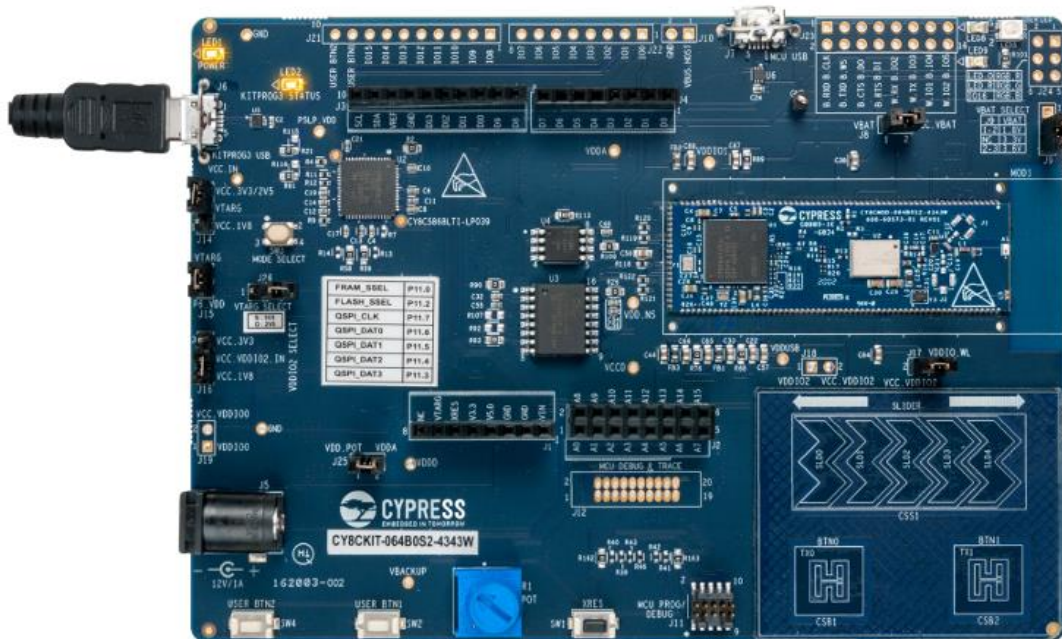


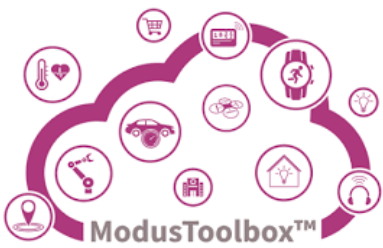
Fig 3.2

CY8CKIT-064B0S2-4343W PSoC 64 “Secure Boot” Wi-Fi BT Pioneer Kit shown in Fig 3.2 a true programmable embedded system-on-chip, integrating a 150-MHz Arm® Cortex®-M4 as the primary application processor, a 100-MHz Arm Cortex-M0+ that supports low-power operations, pre-configured with a root-of-trust and secure processing environment, up to 2 MB Flash and 1 MB SRAM, Secure Digital Host Controller (SDHC) supporting SD/SDIO/eMMC interfaces, CapSense™ touch-sensing, and programmable analog and digital peripherals that allow higher flexibility, in-field tuning of the design, and faster time-to-market[8].

3.4 Software Considerations

When it comes to edge AI systems, it is important to choose software with specific characteristics to make the work optimally. By analysing various factors, verifies that the selected software meets the requirements of the Edge AI system. Compatibility is an important consideration when selecting software. The software must run efficiently on the hardware and enable real-time data processing. Additionally, the ensures compatibility with other software components used on the system, including B. Operating Systems, Libraries, and Frameworks, Seamless Compatibility Integration and Features. Scalability of the is another important aspect that the must consider when designing edge AI systems. Considering that edge AI systems often process large amounts of data, the software must be able to handle the real-time processing and analysis requirements associated with such data Scalable software enables the system to handle increasing data volumes, processing requirements, and user requests without impacting performance. The accuracy of the software used in edge AI systems is critical. These systems rely heavily on accurate data analysis and processing to provide meaningful insights and support decision making. Therefore, the software must have high accuracy and reliability in analysing and processing data. interpretability, or the software’s ability to explain results in an understandable way, plays a key role in the design of the Edge AI system. interpretability allows the user to understand the decision-making process of her system and provides insight into data analysis. This aspect is especially important in applications where decisions made by edge AI systems have a significant impact, such as healthcare and finance [14].

Software used in edge AI systems must prioritize interpretability and present results in a clear and understandable manner.

Tool	Description
	<p>ModusToolbox™ is a set of tools to help you develop applications for Infineon devices. These tools include GUIs, command-line programs, software libraries, and third-party software that you can use in just about any combination you need.. It provides ML tools , libraries, middleware for ML applications[11].</p>

	<p>Imagimob AI is a leading platform for Machine Learning (ML) solutions for edge devices, providing an end-to-end machine learning toolchain that is highly flexible and easy to use with a strong focus on delivering production-grade ML-models for a wide range of use cases building on Infineon’s advanced sensors and comprehensive IoT solutions, such as audio event detection, voice control, predictive maintenance, gesture recognition, signal classification as well as material detection[16].</p>
	<p>Edge Impulse are a cloud-based Machine Learning training platform that is deployable to TinyML devices. With Edge Impulse’s partner ecosystem and expansive training content, deploying machine learning models to embedded microcontrollers is made easier[15].</p>
	<p>SensiML are an AutoML training platform designed to speed up machine learning deployment with their ease-of-use tools[15].</p>

Table 3.1 ML Tools & Description

3.5 Challenges in Embedded AI

Latency Challenges Latency is a critical issue that can significantly affect the performance of AI systems. These systems have three types of latency problems: input delay, processing delay and output delay. Input latency is the delay between the time the edge device stores a data sample and the time the AI model processes it. This may be due to his factors such as slow sensor response time, data transmission delay, and additional data processing costs. Input lag affects the accuracy and timeliness of AI predictions and can lead to lost opportunities to make real-time decisions. Processing latency, on the other hand, refers to the delay between the time an AI model receives a data sample and the time it generates a prediction.

Delays may be caused by factors such as the complexity of the AI model, the size of the input data, and the processing power of the Edge device. This affects the real-time responsiveness of AI predictions and can cause delays in critical applications such as medical diagnostics and

autonomous driving [6]. Output delay is the delay between generating an AI prediction and sending it to the user or downstream system. Several factors can cause this problem, including network congestion, communication protocol overhead, and synchronization between devices. Printing delays can affect the usability and effectiveness of AI predictions and cause delays in decision making and action.

Power Consumption Consumption issues EdgeAI systems can also face challenges related to high power consumption, which can limit deployment and impact, especially in remote and harsh environments such as industrial plants, farmland, and highways. The high energy demand is due to the need for efficient computing resources to process and analyse data in real time. Edge devices often require powerful processors, memory and storage devices that consume a lot of power [9]. This makes it difficult to operate such systems in power-constrained environments. In addition, always-on connectivity and data transfer between peripherals and the cloud can increase energy consumption even more.

Scalability Challenges Edge AI systems can face significant scalability challenges that can affect efficiency, reliability, and flexibility. Scalability is the ability of a system to handle increasing amounts of data, users, or devices without reducing performance. These challenges can be divided into three categories: computational scalability, data scalability, and system scalability [21]. Compute scalability is the ability of edge AI systems to process increasingly large amounts of data without exceeding the processing power and storage capacity of edge devices. Limited processing power, memory and storage space in peripherals limit the size and complexity of AI models, which can hinder accuracy and responsiveness. Data scalability, on the other hand, is the ability of edge AI systems to process increasing amounts. of data without sacrificing performance. Processing large amounts of data in real time on edge devices can be difficult due to limited data transfer capacity and unreliable connections, which can limit the amount and quality of data that can be transferred and processed.

3.6 Future of Embedded AI

Neuromorphic Computing: Improving AI Intelligence by Imitating the Human Brain The future of Embedded AI will also be shaped by new AI paradigms such as neuromorphic computing. This approach mimics the structure and function of the human brain and emulates the neural networks and synaptic connections within our brains. It is based on a new neuromorphic chip that processes information more efficiently while adapting to new situations more quickly and effectively. The neuromorphic chip consists of many artificial neurons and artificial synapses that can mimic the functionality of the brain tip.

Research on neuromorphic computing therefore brings us one step closer to understanding, deciphering, and using the human brain's code in AI applications [6].

The neuromorphic computing chip is well-suited to deliver Embedded AI benefits at scale. This is because the uses less power and is faster than traditional processors. Most importantly, the Embedded AI system is equipped with thinking capabilities like the human brain.

This is very useful in many ubiquitous applications (obstacle avoidance, robust auditory recognition, etc.). As neuromorphic computing matures, this technology will enable a new generation of AI-based edge devices that can learn and adapt in real time[9].

In-Memory Computing for Embedded AI In-memory computing is another technology trend that will impact the future of Embedded AI. This involves storing and processing data directly in the device's memory, rather than relying on traditional storage systems (such as hard drives). This approach aims to significantly reduce data access time and accelerate the computing speed of Embedded AI systems [16]. This further improves the Embedded AI system 's real-time analysis and decision-making capabilities. In the future, Embedded AI applications will need to process larger amounts of data quickly. Therefore, in-memory computing becomes increasingly important to optimize performance and Embedded AI efficiency. Specifically, Embedded AI enables devices to process complex algorithms and extract valuable insights from data at unprecedented speeds [9].

Data-efficient AI: Maximizing the value of in the absence of Ongoing research into data-efficient AI involves expanding and using pre-trained models with domain knowledge, e.g. Many technologies are being considered, ranging from transfer learning (transfer learning) Extensions to the paradigm include humans and characterize processes (such as active learning) as part of human-AI interactions. There are also moderate quality, popular data-efficient techniques to reduce the size of AI models[17]. Data One of the key challenges with AI is that large amounts of data are required to effectively train machine learning algorithms (e.g.deep learning).There is often a lack of sufficient high-quality data to train such algorithms. This issue applies to Embedded AI systems and applications. Additionally, Embedded AI systems face computational and storage limitations that prevent them from taking full advantage of large AI models and large numbers of data points. Data-efficient AI techniques aim to overcome the above limitations by allowing AI models to learn from limited data samples[6].

Chapter 4

Work done at Infineon

At Infineon I am interning in Applied Machine Learning team a one stop solution for AI/ML activities. AML team mission is to provide state of the art AI methods develop AI based applications with focus on R&D in EDA also engage in development of safe and secure low power HW/SW IP for AI. The team operates in four different landscapes which are AI in SoC Development, AI in Software Solutions, AI in Products, AI Infrastructure & Data Mesh in which my work contributes to AI in products to develop hardware compatible AI models by providing support with applications & services. Vision of the team is to continuously drive innovation and business value using cutting-edge machine learning techniques. Our team's goal is to utilize the latest advancements in the field of artificial intelligence to create intelligent systems that can learn from data and make predictions, automate processes, and derive insights that can drive competitive advantage for our company.

4.1 Research and Exploration

Model development is a key area of research in the field of Machine Learning. Machine Learning models are used to learn patterns from data and make predictions or classify new observations. The performance of these models depends on several factors [9], including the quality of the data, the choice of algorithm, and the selection of hyperparameters.

In recent years, there has been significant research in the development of new Machine Learning models and techniques. One of the most exciting areas of research is deep learning, a technique that uses artificial neural networks to learn complex patterns from data. Deep learning has been used to achieve state-of-the-art performance in a variety of tasks, including image recognition, speech recognition, and natural language processing.

Another area of research is reinforcement learning, a technique that involves training an agent to make decisions based on feedback from the environment [7]. Reinforcement learning has been used to develop intelligent systems that can play games, control robots, and optimize processes.

There has also been research in the development of specialized Machine Learning models for specific tasks. For example, there are models designed specifically for time series analysis, text classification, and anomaly detection. These models are optimized for their respective tasks and have been shown to outperform general-purpose models in certain scenarios.

Exploration in model optimization involves finding the best set of hyperparameters that maximize the model's performance [10]. Hyperparameters are parameters that are set before training and affect the learning process. Examples of hyperparameters include learning rate, batch size, and regularization strength. By tuning these hyperparameters, we can improve the accuracy and speed of the model. Optimization can also involve preprocessing and feature engineering to improve the quality of the data. For example, we might remove outliers or missing values, scale the data, or perform feature selection to remove irrelevant or redundant features. These techniques can help to improve the model's accuracy and reduce its computation time.

Deploying models on edge devices can be challenging due to the limited resources available on these devices, such as memory, processing power, and battery life. To overcome these challenges, we need to optimize the model for deployment on edge devices. This may involve

reducing the size of the model, using more efficient algorithms, or using specialized hardware, such as GPUs or MCUs.

In addition to optimization, we also need to consider the security and privacy implications of deploying Machine Learning models. We need to ensure that the model and the data it processes are protected from unauthorized access and misuse. This may involve using encryption, access controls, or anonymization techniques.

In summary, model optimization and deployment are critical components of Machine Learning development. By optimizing our models for performance, reliability, and efficiency, and by deploying them on edge devices or cloud-based environments, we can make Machine Learning applications more accessible and useful to end-users.

4.2 Machine Learning Model Development

A speech command classification model is developed on TensorFlow's speech command dataset which has 8 different voice commands which are pre-processed then fed to CNN model for a classification task. The model is developed to deploy on the PSoC-6 from where the speech commands will be detected in real time. Other than these various models were developed for learning purpose [6].

Flow of the Development:

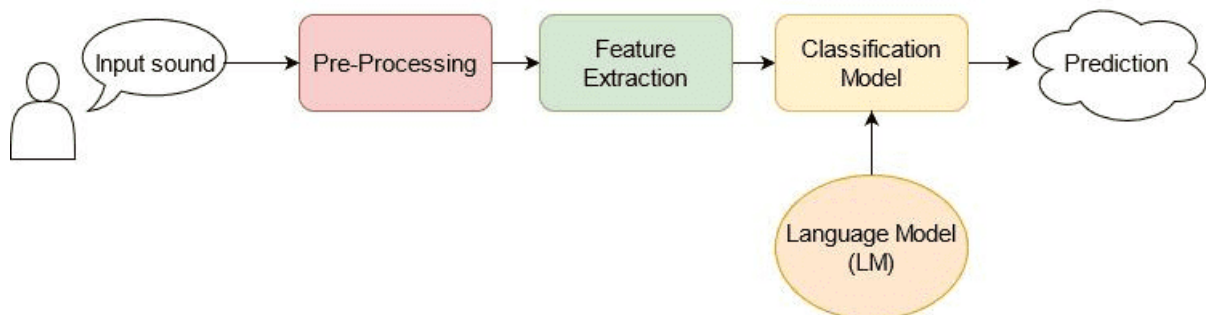


Fig 4.1

1. **Data Preprocessing:** The step involves gathering, cleaning, and preprocessing the data. In your case, you have used TensorFlow's Speech Command Dataset, which contains thousands of audio samples of commands. You need to transform the audio signals into numerical features that can be used by a machine learning algorithm. This process involves techniques such as Fourier transforms and Mel-frequency cepstral coefficients (MFCC).
2. **Model Selection:** In this step, you need to select an appropriate machine learning algorithm for your task. Since you are working with audio data, you might consider using a convolutional neural network (CNN) or a recurrent neural network (RNN). TensorFlow provides pre-built models that can be used as a starting point, or you can build your own model from scratch.
3. **Training:** Once you have selected a model, you need to train it on the data. This involves splitting the data into training, validation, and testing sets. During training, the model learns to recognize patterns in the audio data, and its performance is evaluated on the validation set. You can adjust the hyperparameters such as batch size, learning rate, and optimizer during the training process to achieve better result

4. Deployment: Once you have a trained and evaluated model, you need to deploy it to a production environment. This might involve optimizing the model for deployment, such as reducing its size or using more efficient algorithms.

Design of Developed Model

1. Input Shape: The model takes an input tensor of shape (16, 5, 1), which represents a 1D audio signal with 16 frequency bands and 5 time steps.
2. Conv2D Layer: The first layer of the model is a 2D convolutional layer with a certain number of filters and kernel size. This layer applies a set of learnable filters to the input signal, extracting features that are relevant to the classification task.
3. Batch Normalization: After the convolutional layer, a batch normalization layer is applied to normalize the output of the convolutional layer.
4. Average Pooling 2D: The output of the batch normalization layer is passed through a 2D average pooling layer, which performs down-sampling of the output by taking the average of a small window of features.
5. Conv2D Layer: Another 2D convolutional layer is applied to extract more complex features from the down-sampled output of the previous layer.
6. Batch Normalization: The output of the second convolutional layer is again normalized using a batch normalization layer.
7. Average Pooling 2D: Another 2D average pooling layer is applied to down-sample the output of the second batch normalization layer.
8. Flatten: The output of the second average pooling layer is then flattened into a one-dimensional vector, ready to be passed to a fully connected layer.
9. Dropout: A dropout layer is applied to the flattened output to prevent overfitting. Dropout randomly removes a certain percentage of the neurons in the layer, forcing the network to learn more robust features.
10. Dense Layer: Finally, a fully connected dense layer with a certain number of units is applied to the output of the dropout layer, producing the final output tensor.
11. Output: The output of the dense layer is the model's prediction for the input audio signal.

In summary, speech command classification model uses a sequence of convolutional and pooling layers, followed by a fully connected layer, to learn features from the input audio signal and produce a prediction for the classification task.

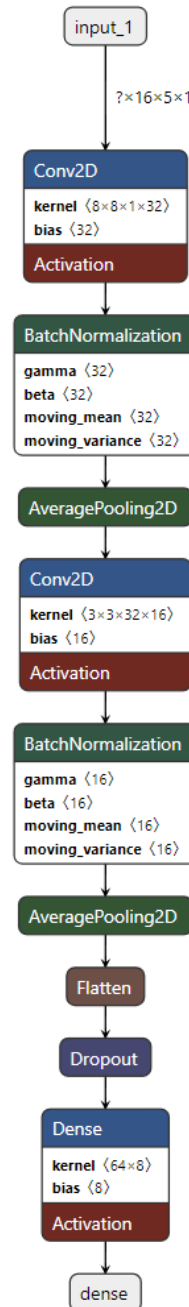
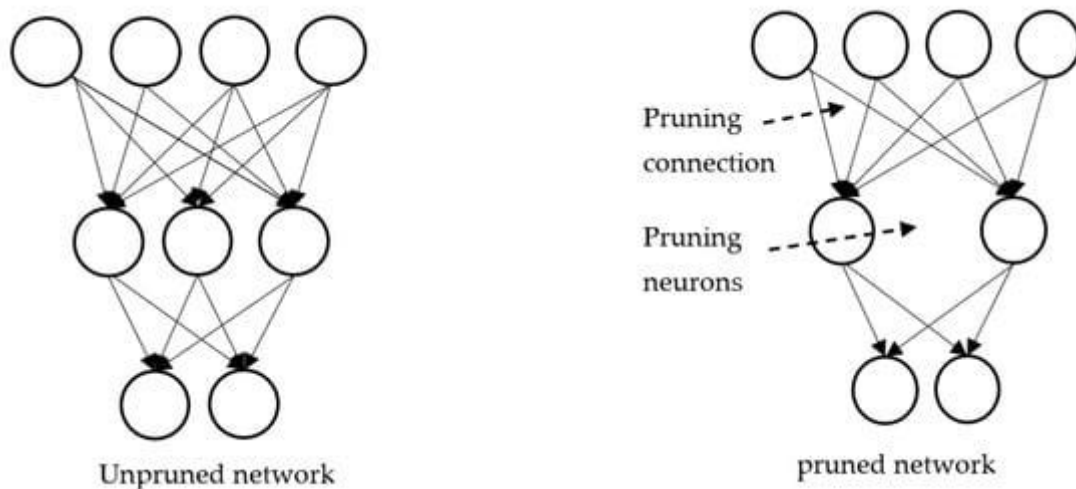


Fig 4.2

4.3 Exploration: ML Optimization Techniques

In the fields of computer vision, natural language processing, video analytics, etc., deep neural networks are being used in innovative and impressive ways. However, the computational resources required to implement neural networks are on the higher side. In addition, the energy consumption of these architectures is also high, while the heat they dissipate into the environment is extremely harmful. The implementation of such feature rich DNNs in Internet of Things (IoT) edge devices is full of technical challenges and concerns due to the limited hardware and power resources of the edge devices. Therefore, DNN models should be optimized to significantly reduce the use of huge computational resources. Clipping: Larger models require more memory and energy, are difficult to partition and consume a lot of computation. Cropping creates models that are smaller in size for inference. Due to its smaller size, the model becomes both memory and energy efficient, and reasoning is faster and with minimal loss [3]. Pruning is done by removing unimportant connections or neurons, as in figure



4.3.

Fig 4.3

Quantization: Basically, quantization means reducing the accuracy of weights, parameters, biases, and activations so that they occupy less memory and reduce the size of the model. In an artificial neural network, the weights are 32-bit floating point values. Consider a neural network with millions of parameters. Here, the memory required to store millions of 32-bit floating point values is too large to accommodate peripherals. Thus, 32-bit floating-point values are usually converted to 8-bit integers. During quantization, the range of parameters or weights must be scaled to an 8-bit integer range (ie -127 to +127). This process is called scale quantization. In addition to scale quantization, quantization data must be grouped, which is called quantization granularity. This means whether quantization is applied per channel (in 3D input) or per row or column (in 2D input). Figure 4.4 shows how PTQ is called weight loss [19].

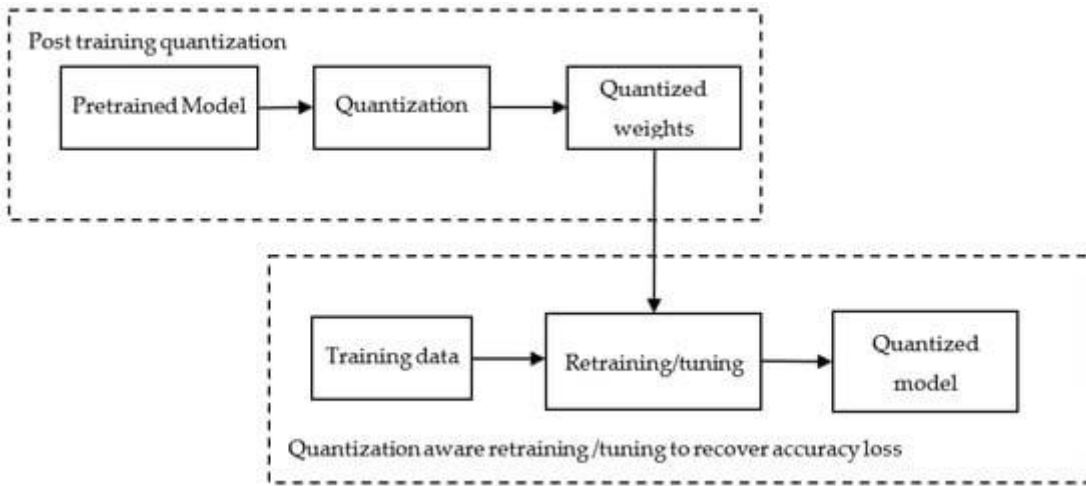


Fig 4.4

Weight Sharing: There is much redundancy among the weights of a neural network, and this proves that a small number of weights are enough to reconstruct a whole network. In the weight-sharing method, the number of effective weights that are required to be stored is reduced by having multiple connections in a neural network share the same weight [9]. The weight sharing may be based on a random method or Hashed Net in which the weights are grouped according to a Hash function [17,16]. The concept of weight sharing is exemplified in Fig 4.5.

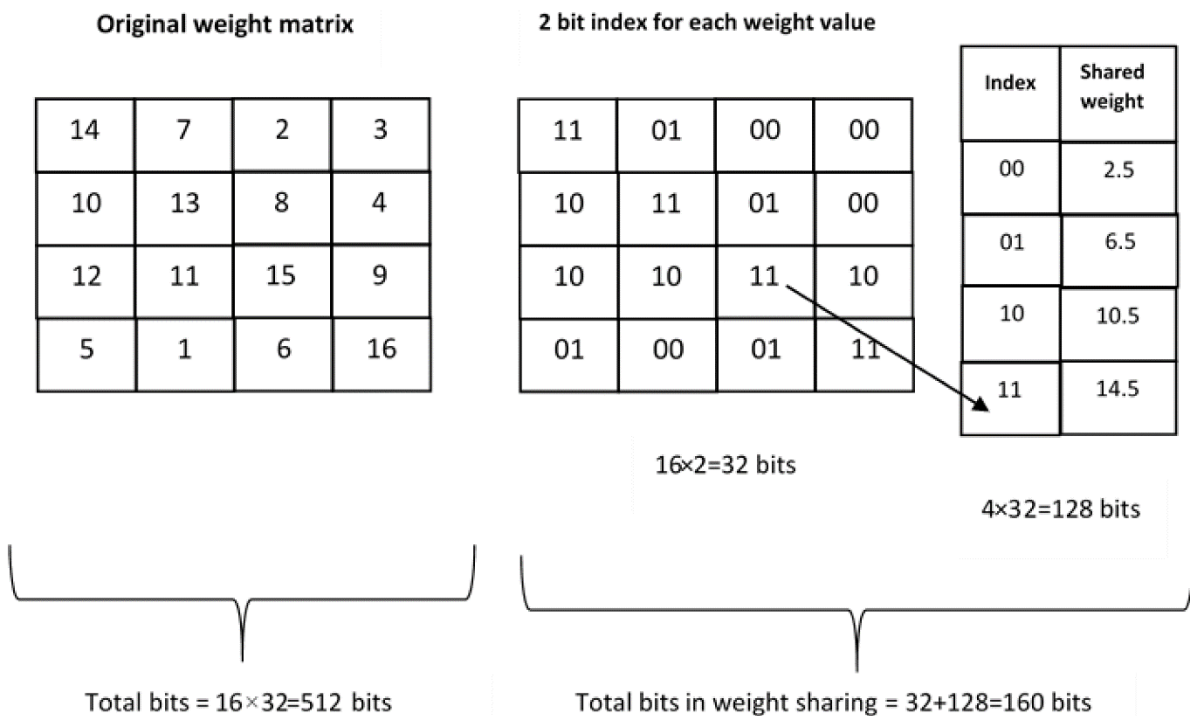


Fig 4.5

4.4 Voice Detection at Edge

As seen earlier a speech command model is being developed which is now been deploy on PSoC-6 MCU for real time voice detection. Before beginning let's have a look at PSoC-6 PSoC™ 6 Wi-Fi BT Prototyping Kit provides [13]:

- Snappable peripherals: Capacitive-sensing CapSense™ slider and buttons, Digilent Pmod interface, 512Mb QSPI NOR flash, uSD card, PDM-PCM microphone, thermistor
- Bread-board compatible form-factor
- Murata LBEE5KL1DX-TEMP Module (CYW4343W) that provides IEEE 802.11a/b/g/n WLAN + Bluetooth [18].

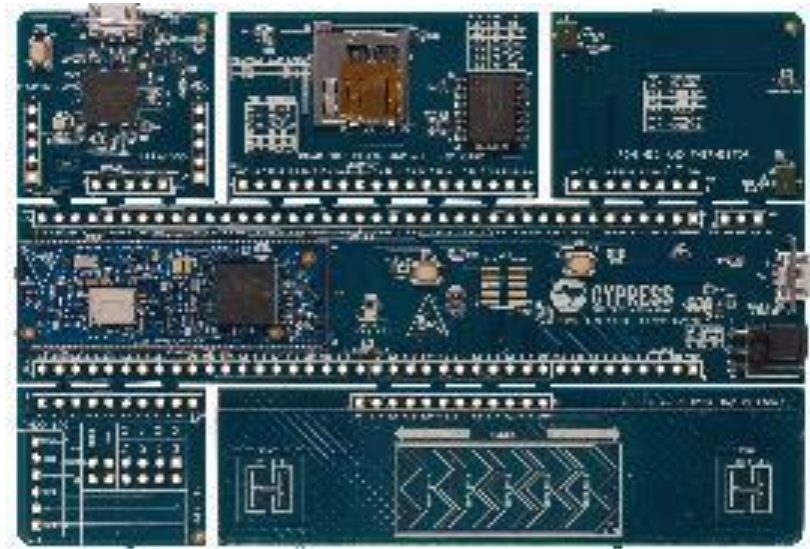


Fig 4.6

Objective:

- Neural Networks (NN) on microcontrollers to run offline speech recognition. End-to-end system development, from audio detection and acquisition, NN model training, deployment of the trained model on to the MCU, integration and testing
- The speech recognition model is aim at understanding and comprehending WHAT is spoken and according to that performing the task.
- In this project it is showed ML algorithms and their implementation on Infineon low power Wifi-BLE prototyping kit.



Fig 4.7

Technical Flow:

1. **Model conversion:** The first step in implementing voice detection on edge using PSoC involves converting a pre-trained machine learning model into a format that can be used on the PSoC. This might involve converting the model to a format such as TensorFlow Lite, ONNX, or Caffe2.
2. **Code Header Generation:** Once the model has been converted, the next step is to generate C header files that describe the model's architecture and parameters. These files will be used to compile the model into the PSoC firmware.
3. **Code Integration:** The next step is to integrate the generated C header files into the PSoC firmware. This might involve writing C code to load the model, allocate memory for input and output tensors, and perform inference on new input data.
4. **Sensor Interfacing:** To perform voice detection, the PSoC needs to be connected to a microphone or other audio sensor. This might involve interfacing with the sensor using protocols such as I2S or PDM, configuring the sensor's sampling rate and resolution, and buffering the input data.
5. **Feature Extraction:** Once the audio data has been acquired, it needs to be pre-processed and converted into features that can be used as input to the machine learning model. This might involve applying a windowing function to the audio data, performing a Fourier transform or other signal processing techniques, and computing features such as Mel-frequency cepstral coefficients (MFCCs).
6. **Inferencing:** Once the input features have been computed, they are passed to the machine learning model for inferencing. The model performs a series of matrix multiplication and activation functions to produce a prediction for the input data. The output of the model might be a binary classification (voice vs. non-voice) or a probability score indicating the likelihood of voice.

In summary, the technical flow for voice detection on edge using PSoC 6 involves converting a pre-trained model, generating C header files, integrating the model into the PSoC firmware, interfacing with an audio sensor, preprocessing the input data, and performing inferencing on the model. This flow can be customized depending on the specific requirements of the voice detection application.

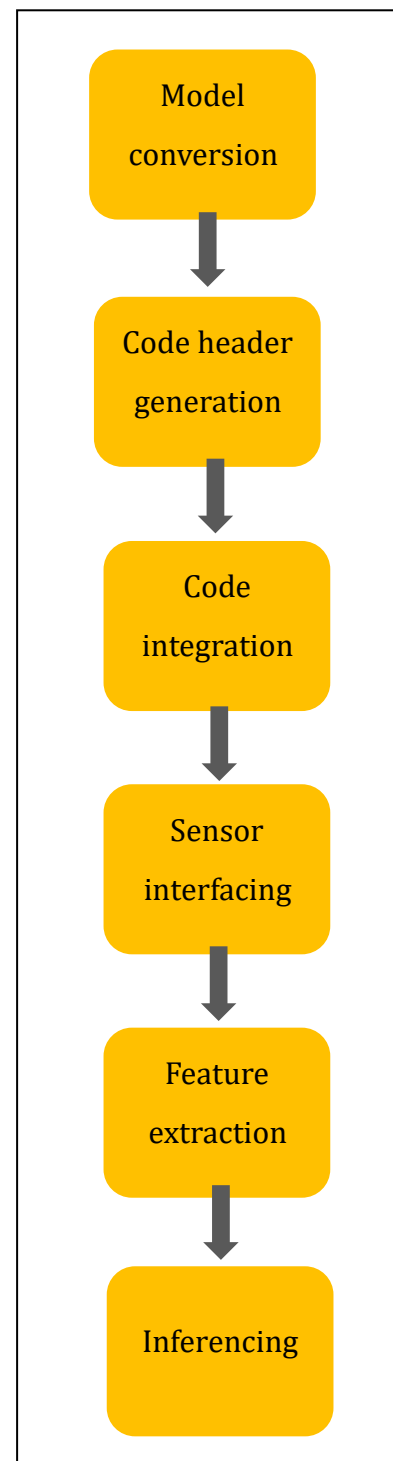


Fig 4.8

Results:

As shown in Fig 4.9 the model is able to detect the speech command the predicted class is being output to terminal with its confidence score.

```
Predicted Class =          on Confidence 0.95
Low Confidence score < 0.8, actual confidence 0.76 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.41 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.35 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.36 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.35 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.30 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.33 - Marked as unknown

Predicted Class =          on Confidence 0.92
Low Confidence score < 0.8, actual confidence 0.28 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.42 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.32 - Marked as unknown
Low Confidence score < 0.8, actual confidence 0.34 - Marked as unknown
```

Fig 4.9

4.5 Radar Gesture Classification

Radar gesture classification is a technique that uses radar signals to detect and classify different hand gestures based on the movement of the hand. This technique is becoming increasingly popular due to its ability to work in low-light or complete darkness, and its ability to operate in harsh environments where cameras or other sensors may not be effective.

One example of a radar gesture classification system is a machine learning model that is deployed on an FMCW RADAR SENSOR. The FMCW 60 GHZ RADAR SENSOR is a hardware platform that includes a radar sensor, a microcontroller, and other components needed to interface with the sensor and perform machine learning tasks [2].

To deploy a machine learning model on the FMCW RADAR SENSOR, the first step is to collect data using the radar sensor. This data includes information about the position, velocity, and other characteristics of the hand gestures. Once the data is collected, it is pre-processed and used to train a machine learning model [5].

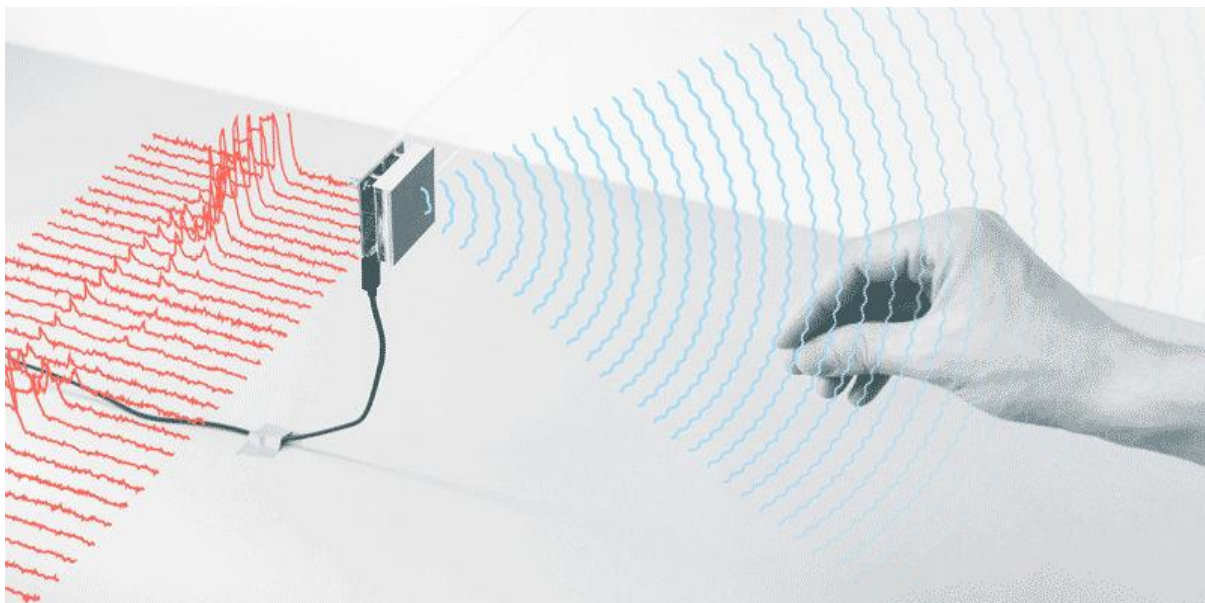


Fig 4.10

Gesture classification model

A convolutional neural network (CNN) model consists of two convolutional blocks and two fully connected layers. Each convolution block contains convolution operations including smoothed linear unit (ReLU) and maximum pooling with a group normalization layer added after the first block. Convolution layers act as feature separators and provide abstract representations of input sensor data in feature maps. They reflect the short-term dependencies (spatial relationships) of the data. In CNN, features are extracted and then used as inputs to a fully connected network using softmax activation for classification.

The Netron model for Radar Gesture Classification is a sequential model that uses different layers to extract relevant features from the input data. It takes an input shape of (128, 6, 1) and follows the below steps:

1. Input layer: This is the first layer in the model. It takes in the input data with a shape of (128, 6, 1).
2. Convolutional layer (Conv2D): The input data is passed to the Conv2D layer, which applies filters to the input data to capture important features. It has 8 filters, a kernel size of (3, 3), padding of "same," and uses the "relu" activation function to introduce non-linearity in the output.
3. Max Pooling layer (MaxPooling2D): The output from the Conv2D layer is then passed to a MaxPooling2D layer which reduces the spatial dimensions of the output by taking the maximum value of a (3, 3) pooling window. This reduces the dimensionality of the data and helps in preventing overfitting.
4. Batch Normalization layer: This layer normalizes the activations of the previous layer, reducing the internal covariate shift and improving the training speed and performance of the model.
5. Dropout layer: This layer randomly sets a fraction of the input units to 0 at each training update, which helps to prevent overfitting of the model.
6. Another Conv2D layer: The output from the Dropout layer is passed through another Conv2D layer that uses 16 filters, a kernel size of (3, 3), padding of "same," and again uses the "relu" activation function. Conv2D layer is passed to another MaxPooling2D layer with a pool size of (3, 3).
7. GlobalAveragePooling2D layer: This layer computes the average value of each feature map in the previous layer. This reduces the dimensionality of the data and helps in preventing overfitting.
8. Output Dense layer: This is the final layer of the model and has 4 units with a "softmax" activation function. This layer outputs the predicted probabilities for each of the 4 possible gesture classes.

The resulting output of the final Dense layer has a shape of (128, 4), which represents the predicted probabilities for each of the 4 possible gesture classes as shown in Fig 4.11 Model Diagram

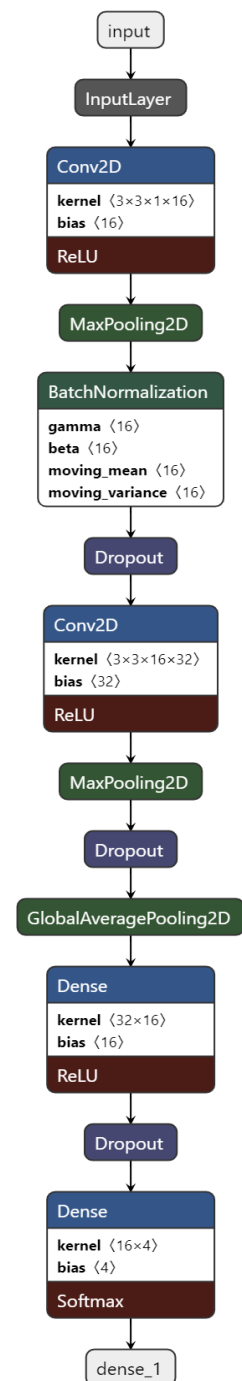


Fig 4.11

Block diagram of Design Flow:

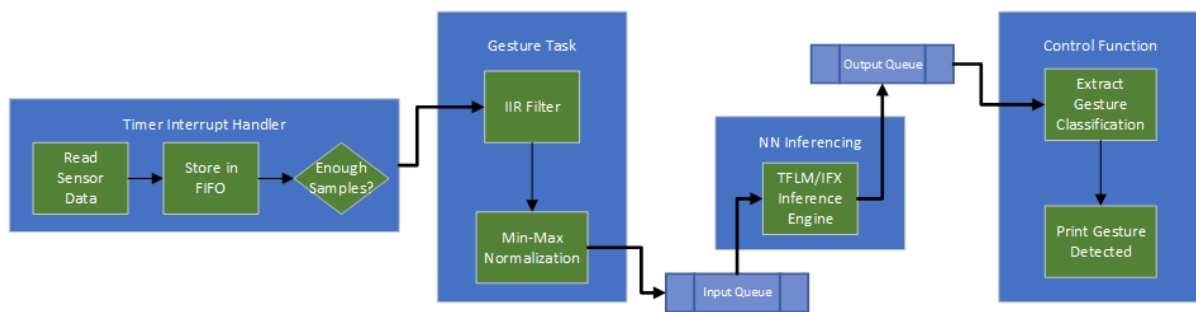


Fig 4.12

The design flow for the Radar Gesture Classification model deployed on a FMCW 60 GHz RADAR SENSOR involves multiple components that work together to accurately classify a user's gestures from Fig 4.12. These components include a Timer Interrupt Handler, Gesture Task, NN Inference, and Control Function [11].

The Timer Interrupt Handler is responsible for triggering the data acquisition process and analysing the radar signals. It runs on a timer and is triggered at predefined intervals to initiate the radar system and collect new data from the environment. The radar system captures the reflected signals from the user's hand and provides a time-domain representation of the hand movement.

The Gesture Task is responsible for preprocessing the raw radar data, which includes filtering, normalization, and feature extraction. This preprocessing is critical to ensure that the input data is of high quality and contains only relevant information. The pre-processed data is then passed to the NN Inference component.

The NN Inference component is responsible for classifying the gesture being performed by the user. It uses a trained machine learning model to accurately identify the gesture. The machine learning model has been trained on a dataset of labelled radar data and is capable of accurately identifying the gesture being performed by the user.

The Control Function is responsible for translating the classification result into a desired action. For example, if the user performs a gesture to increase the volume of a device, the classification result is used to trigger the volume to be increased. The Control Function is also responsible for sending feedback to the user regarding the success or failure of the desired action.

Overall, the design flow for the Radar Gesture Classification model deployed on a FMCW 60 GHz RADAR SENSOR involves the Timer Interrupt Handler triggering the data acquisition process, the Gesture Task preprocessing the raw radar data, the NN Inference component classifying the gesture being performed by the user, and the Control Function translating the classification result into a desired action. This allows for an accurate and efficient system for gesture recognition and control [12].

Chapter 5

Research & Development of Neural Architecture Search Algorithm

5.1 Introduction to Neural Architecture search (NAS)

Deep Neural Networks (DNN), the cornerstone of deep learning, have shown great success in various real-world applications, including image classification [23], [25], natural language processing, speech recognition [24] to name just a few. The promising performance of DNNs has been widely documented due to their deep architecture [1], which can learn important features directly from raw data with almost no special features. In general, the performance of DNNs depends on two aspects: their architecture and their associated weights. Only when both reach the optimal state at the same time, the performance of DNNs can be promising. Optimal weights are often obtained through learning: a continuous loss function is used to measure the differences between the actual output and the desired output, and then gradient-based algorithms are often used to minimize the loss. Once the termination condition is satisfied, which is usually the maximum number of iterations, the algorithm can often find a good set of weights. Such processes have been very popular in practice due to their efficiency and have become the dominant practice in weight optimization [26], although they are mainly local search algorithms. On the other hand, obtaining optimal architectures cannot be directly formulated by a continuous function, and there is not even any explicit function that measures the process of finding optimal architectures. To this end, promising DNN architectures are designed by hand and with great expertise. This can be demonstrated by prior art such as VGG, ResNet and DenseNet. These promising Convolutional Neural Network (CNN) models are hand-designed by researchers with extensive knowledge of both neural networks and image processing.

Neural Architecture Search Components Neural Architecture Search (NAS) is an emerging field in deep learning research that aims to improve model performance and applicability. Despite all its potential, setting up a NAS can be difficult. Specifically, NAS can be divided into three main components: search space, search strategy/algorithm, and evaluation strategy. These elements can be manipulated in several ways to maximize the search for efficient neural network designs. Understanding the interaction of these components is important for using NAS to improve the performance and capability of deep learning models and applications [22]. These components are:

- Search space: The search space of Neural Architecture Search defines the set of possible neural network architectures that the algorithm explores to find the optimal model.
- Search Strategy: The search strategy of a neural architecture specifies the method or algorithm used to navigate the specified search space and find optimal neural network architectures.

- Evaluation Strategy: The evaluation strategy of Neural Architecture Search involves evaluating the performance of candidate neural network architectures in the search space to determine their performance against predefined criteria or goals.

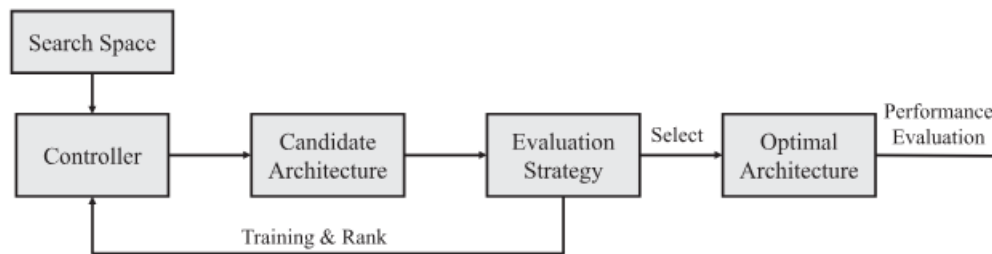


Fig 5.1 The general framework for NAS

5.2 The Need for Hardware-Aware NAS

The need for efficient implementation of DNN networks is increasing as they are increasingly used in many industries. However, this is a difficult task that requires unique engineering skills and a lot of resources. NAS allows you to[26]:

- Automated design: NAS automates the neural network architecture design process, reducing the need for manual intervention. This allows you to explore simple and effective architectures.
- Best performance: NAS aims to find the best architecture for specific tasks and data sets. These requirements improve model performance compared to manually generated architectures.
- Complexity of Neural Network Design: Designing an optimal neural network architecture is a complex task that requires a deep understanding of the problem, significant machine learning expertise, and significant computing resources. A NAS automates this process, reducing complexity and increasing accessibility.
- Efficiency and Efficiency: NAS algorithms help identify the optimal neural network architecture that provides the best performance for a task, improving the efficiency and accuracy of predictions.
- Prototyping: As the volume and complexity of data increases, the NAS can perform prototyping to better handle this growth, creating larger and more complex architectures without manual correction.

5.3 Neural Network & Hardware Co-optimization

Cooperative optimization uses a block-based search space for the neural network architecture. The search for accelerated architecture configurations and objective neural networks is carried out as an evolutionary multi-objective optimization. As shown in Algorithm 2, the first search selects random neural networks with their hardware configurations from the common search space. S . The neural network is trained with quantile learning and the trained neural network is evaluated with a validation set to obtain accurate measurements [23]. After the optimization phase, architectural parameters and performance metrics are added to the search history. Once the initial population size s is reached, a new architecture is derived from the existing population through elementary mutations. During the search, NAS was randomly selected from among the following variables.

1. Add/Remove Blocks: This involves adding or removing entire blocks (or layers) from the neural network architecture. Since each block contributes to the complexity and power of the model, this has a significant impact on the performance and computational requirements of the model.
2. Change the block type between residual and feed forward: This involves changing the block type from a feed forward block (information flows in one direction, from input to output) to a rest block (shortcuts or connections that allow the information flows between layers). This can affect the learner's ability to learn complex patterns and handle cases where the gradient is missing.
3. Convolutional layers can remove features from the input data, and adding or removing features can change the model's ability to learn from the data. Rotate Size
4. Increase/Decrease: Refers to adjusting the spatial extent of the rotation filter. Increasing the size allows the model to capture larger patterns in the data, while decreasing the size allows the model to overestimate local subsamples.
5. Increase/Decrease Key Block Element: The step is the step size, or amount of movement, that the rotary filter makes when moving the input data. Increasing the step can reduce the computer load and output size, and decreasing it makes the model more sensitive to the details of the input data.
6. Increase/decrease measurement word width: Neural network measurement reduces the accuracy of weights and biases, thereby reducing memory requirements and computational complexity. Adjusting the measurement term width affects the balance between model accuracy and computational efficiency.
7. Multiple Accumulation (MAC) element size increase/decrease: In hardware design, MAC operations are central to many machine learning algorithms. Changing a set affects the parallelism and output of calculations and the overall performance of the neural network.
8. Increase/decrease the number of output channels: This involves changing the number of feature maps created by the convolutional layer. Upscaling allows the model to learn more complex features, and downscaling can reduce computing requirements.

5.4 Architecture of Hardware-Aware NAS

NAS architecture is a complex technique designed to automatically find the optimal neural network architecture. It uses data, hardware information, and search methods to integrate various components into connected systems. Complete the process and create a perfect neural network ready for deployment [24].

1. **Data and target hardware:** The first phase includes the data used to train and validate the NN and the target hardware on which the nn is deployed. This includes the selection of appropriate data sets and the hardware constraints that the neural network must meet. Hardware limitations may vary based on factors such as computer power, memory capacity, power consumption, and storage requirements. It is important to consider these factors so that the neural network works well on the selected device.
2. **NAS Algorithm:** The core of the NAS architecture.
 - **Search Space:** NAS algorithms first define a search space, usually a set of possible neural network architectures. For a hardware compatible NAS, this search field will be adjusted according to the hardware requirements. This means that the NAS algorithm only considers architectures that can run efficiently on the target devices and ensures that the product architecture is optimized for the device.
 - **Optimization:** Once the search space is defined, the NAS algorithm uses optimization techniques to select the best architecture in the search space. This may include techniques such as reinforcement learning, developmental transitions, or gradient-based techniques. The optimization process considers many metrics such as model accuracy, computational cost, effort, etc. to determine the most efficient and effective architecture.
3. **Constructed neural network:** Once the NAS algorithm finds the optimal architecture, it generates the corresponding neural network to be used in real applications. These networks are designed to provide the best performance on the target devices, taking into account the constraints and requirements specified during the search. The resulting template can be expanded or edited as needed.

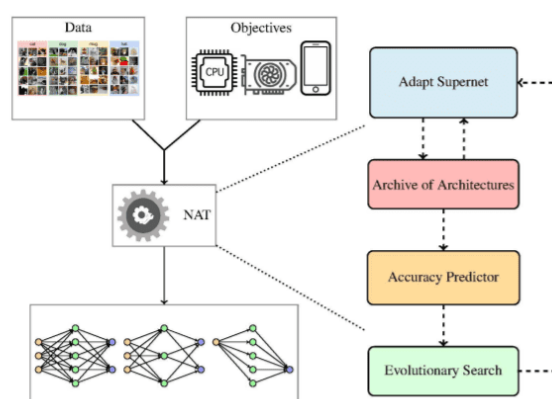


Fig 5.2 NAS Architecture

5.5 Aging Evolution in NAS

Aging evolution is a powerful strategy used in hardware-aware neural network architecture (NAS) research. Inspired by the principles of biological evolution, this strategy aims to optimize the design of neural networks, fostering a balance between the exploration of new architectures and the use of known successful architectures. In the context of a hardware-aware NAS, legacy evolution plays an important role in ensuring that the evolution process is aware of the hardware limitations of the target device. The goal is to identify an architecture that not only performs well in terms of predictive accuracy, but also meets the specific resource constraints of the selected device.

Working of NAS algorithm:

Initial Population: The aging evolution starts with an initial population of neural network architectures, each of which is randomly generated. The size of this population is determined.

Assessment: Each architecture in the population is evaluated on its "baseline", which measures how well the architecture performs its intended task and respects hardware constraints. This physical measure usually considers factors such as prediction accuracy, computational complexity, effort, and duration.

Selection and Modification: Based on physical evaluation, certain architectures are selected for the next generation. The selection process favors architectures with more relevant points, but also includes less efficient architectures to maintain diversity. The selected architecture receives a modification function to slightly change its configuration. This may include changing the number of layers, layer types, connection patterns, etc.

Ageing: An ancient method is implemented so that the population is not dominated by a few successful architectures at the beginning and promotes diversity. In this process, body art symbols gradually disappear over time, due to the continuous search for new and possibly better architectures in the evolutionary process.

Iteration: Physical evaluation, selection, modification, and aging are performed in several iterations until the optimal architecture is found.

In essence, aging evolution in Hardware-Aware NAS provides a systematic and efficient approach to navigating the vast search space of potential neural network architectures. By incorporating hardware constraints into the evolution process, it ensures the identified architectures are not only effective in predictive performance but also feasible for the target device.

5.6 Evaluation Metrics for Hardware-Aware NAS

The need for computational resources (GPU) is the main limitation for the development of new NAS algorithms for accuracy and performance evaluation. As a result, many NAS benchmarks have been published as LUTs/datasets of various networks and their accuracy/performance metrics. NAS-Bench-101 is the first large-scale NAS benchmark. The 423K cell-based architecture is trained using CIFAR-10 to record test accuracy and training time. NAS-Bench-201[25] extends the NAS-Bench-101 dataset by including four-node, five-task networks in three different datasets. NASBench-301 [15] initially generated 1018 unique neural structures in the ARROW search space. 60,000 sample networks as part of surrogate training. Independent NAS tests provide training times that do not measure the length of each network on the device. These specifications limit their use in developing HW-compatible NAS switches, especially for researchers unfamiliar with the hardware. LatBench[18] is a large dataset that measures the latency of the NAS-Bench-201[25] sample on various devices, including desktop, mobile, CPU/GPU, TPU, and DSP. . HW-NAS-Bench[75] evaluates various networks based on cellular architecture (NAS-Bench-201) and layered networks (FBNet search space) using Jetson TX2 Edge GPU, Raspberry Pi 4. , EdgeTPU, Pixel 3, Eyeriss and XilinxZC706 FPGA

5.7 Advantages of Hardware-Aware NAS

Neural Architecture Search (NAS) offers several advantages in the field of deep learning:

- Hardware config part of guided architecture search: One of the key advantages of Hardware-Aware NAS is the inclusion of hardware configuration in the search process for the optimal neural network architecture. By taking hardware limitations into account when searching, NAS systems not only achieve better performance in terms of model accuracy but can also improve the efficiency of device-optimized search architecture.
- Single OFA model can support diverse platforms: One of the main advantages of hardware-enabled NAS is the ability to develop single-factor authentication (OFA) that can be implemented on a variety of hardware platforms. This OFA model can be replicated on various hardware architectures without recycling, providing flexibility and efficiency in deploying AI models on a variety of hardware platforms.
- Significantly reduced training resource and time: Hardware-aware NAS algorithms can significantly reduce training resources and time. By focusing on architectures that meet hardware limitations, the NAS system can avoid wasting resources on training architectures that may work well but are not feasible for the target hardware. This can significantly save computing resources and training time.
- Smaller model footprint as compared to previous methods: Hardware-aware NAS often yields neural network architectures with a smaller model footprint compared to traditional methods. Optimizing for hardware limitations, these designs can be compact but offer comparable or even better performance.

- Hardware centric optimization: Focusing on the hardware during the optimization process results in models that are highly optimized for the specific target hardware. This can improve model performance, reduce power consumption, and reduce memory usage on the target hardware. This hardware-centric optimization makes hardware-aware NAS especially valuable for peripherals and mobile devices where resource constraints are critical.

5.8 Future Prospect of NAS:

In recent years, HW-NAS has completely transformed the way neural networks have traditionally been designed for various devices. HW-NAS provided the possibility to implement hybrid functions in a single network, which was not available in previous models such as ResNet50 [27]. In this paper, we have provided an overview and summary of several HW-NAS methods targeting resource-constrained devices in high-performance systems, followed by a general overview of several accelerators and algorithmic aspects. In designing robust and efficient architectures, we emphasize the importance of hardware-aware search space, search space construction, and hardware-based multi objective search. Automatically designed networks are more efficient than manually constructed models, and the commonly searched accelerator-network pair outperforms manually constructed models in both accuracy and efficiency. We believe that further research is needed in the following areas.

(1) MCU-NAS: We are still in the early stages of NAS being able to make MCU inferences with reasonable accuracy. MCUNet is, to our knowledge, the only method that provides nearly 70% accuracy in ImageNet. More efficient algorithms and compiler designs are needed to improve the accuracy and efficiency of a limited computing system.

(2) Efficient search space: search elements often constrain the NAS algorithm to find more efficient models. Most HW-NAS methods choose the mobile search mode (MobileNetV2, ShuffleNetV2) as the backbone, even in advanced devices. The accuracy of the models found in this search space is 72-76%, even in the mixed search space. More efficient spaces like EfficientNet can lead to the development of a more rigorous and efficient search space. Recently, convolutions have been used to replace traditional Circuits for Vision applications [26]. Transformer architecture and accelerator co-option can be very effective SOTA designs to achieve higher accuracy.

(3) Algorithm and Accelerator Co-Design: Auto-NBA targets architecture, precision, and accelerator micro-architecture, ignoring data flow. NAAS [24] targets the neural architecture and accelerator (microarchitecture and translator mapping) while still leaving the quantization aspect. Thus, the implementation of DNNs in many use cases requires efficient differential common search methods for all four dimensions, that is algorithm (architecture and accuracy) and accelerator (hardware architecture and translation routines).

(4) Sparse CNN and Accelerator Joint Search: Current HW-NAS methods only search regular/dense convolutional layers, ignoring sparse matrices, and do not consider the sparse Tensor Core hardware of the latest Nvidia A100 GPUs. Sparse CNN and sparse support booster search can be very powerful studies to achieve higher model compression and speedup than the currently searched conventional booster-CNN pair.

(5) Benchmarks: There are multiple hardware such as MCUs, server-grade CPUs and multi-GPUs, core architectures such as MobileNetV3, EfficientNet and data to develop HW-NAS benchmarks. Since quantization is very important for real-time reasoning, HW-quantized NAS benchmarks should be preferred in the future, as current benchmarks do not include precision. NAS HW accelerator brands could be considered because the co-design process covers both dimensions.

(6) NAS for other purposes: NAS algorithms were originally designed and focused only on finding efficient architectures. Later, when NAS became an accessible concept, it was used for other purposes such as pruning, quantization and Winograd search. Thus, the potential of NAS can be extended to other applications and use cases.

(7) HW-NAS for other applications: Although HW-NAS has been used to develop latency efficient models for image classification tasks, little attention has been paid to hardware for other tasks, such as MobileDets [15] for object detection-aware Transformers [25] for NLP, SqueezeNAS [24] for semantic segmentation. Research to optimize and build custom DNN device designs for other vision applications is still an open challenge.

Chapter 6

Conclusion

The report encapsulates the groundbreaking strides made in the integration of Artificial Intelligence (AI) into embedded systems. It demonstrates the transformative potential of AI, not merely as a theoretical concept but as a pragmatic tool that can be seamlessly integrated into everyday devices.

The successful development and deployment of a radar gesture classification model on the Infineon XENSIV™ kit mark a significant stride in the field of embedded AI. The model's real-time identification of distinct gestures presents a leap towards creating more intuitive human-machine interfaces and interactive user experiences across a multitude of applications. This accomplishment underscores the potential of AI to elevate the interaction between humans and machines.

Moreover, the development of various machine learning models throughout the project exhibits how AI can be harnessed to solve complex tasks, thereby enhancing the efficiency and effectiveness of embedded systems. It further amplifies the potential of data-driven solutions and machine learning models in revolutionizing the embedded AI landscape.

The exploration and advancement in the area of "Neural Architecture Search" denote an innovative approach towards the design of neural networks. The emphasis on hardware-aware design could indeed be a gamechanger in the way AI models are developed and deployed in real-world settings. By acknowledging the hardware constraints during the design process, the research opens avenues for creating more practical and efficient AI solutions.

This report manifests the immense potential and promising future of AI in the sphere of embedded systems. Through tangible achievements in radar gesture recognition, the development of machine learning models, and the breakthroughs in hardware-aware neural architecture search, it provides a glimpse into the exciting possibilities that the confluence of AI and embedded systems hold for the future.

References:

- [1] Zhang Z, Li J. A Review of Artificial Intelligence in Embedded Systems. *Micromachines* (Basel). 2023 Apr 22;14(5):897. doi: 10.3390/mi14050897. PMID: 37241521; PMCID: PMC10220566.
- [2] Zhengjie Wang, Fei Liu, Xue Li, Mingjing Ma, Xiaoxue Feng, and Yijing Guo. 2023. A Survey of Hand Gesture Recognition Based on FMCW Radar. In *Proceedings of the 8th International Conference on Communication and Information Processing (ICCIP '22)*. Association for Computing Machinery, New York, NY, USA, 73–79. <https://doi.org/10.1145/3571662.3571674>
- [3] S. Sun, Z. Cao, H. Zhu and J. Zhao, "A Survey of Optimization Methods From a Machine Learning Perspective," in *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3668-3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779.
- [4] K. Alirezazad and L. Maurer, "FMCW Radar-Based Hand Gesture Recognition Using Dual-Stream CNN-GRU Model," 2022 24th International Microwave and Radar Conference (MIKON), Gdansk, Poland, 2022, pp. 1-5, doi: 10.23919/MIKON54314.2022.9924984.
- [5] WEVOLVER [WEVOLVER]. (2023). 2023 EDGE AI TECHNOLOGY REPORT. In <https://www.wevolver.com/article/2023-edge-ai-technology-report>. WEVOLVER. Retrieved October 10, 2023, from <https://www.wevolver.com/article/2023-edge-ai-technology-report>
- [6] Dong, Yaoyao & Qu, Wei. (2021). Review of Research on Gesture Recognition Based on Radar Technology. 10.1007/978-3-030-69066-3_34.
- [7] Infineon Technologies (no date) Infineon-CY8CKIT-064B0S2-4343W_Kit_Guide-UserManual. Available at: www.infineon.com/dgdl/Infineon-CY8CKIT-064B0S2-4343W_Kit_Guide-UserManual-v01_00-EN.pdf (Accessed: 03 December 2023).
- [8] H. Lin, "Embedded Artificial Intelligence: Intelligence on Devices" in *Computer*, vol. 56, no. 09, pp. 90-93, 2023. Doi: 10.1109/MC.2023.3280397 <https://doi.ieeecomputersociety.org/10.1109/MC.2023.3280397>
- [9] Infineon/MTB-example-ml-profiler: This code example demonstrates how to run through the ModusToolbox machine learning (MTB-ml) development flow with psoc 6 MCU, where the end user has a pre-trained neural network (NN) model, which can be profiled and validated at the PC and target device., GitHub. Available at: <https://github.com/Infineon/mtb-example-ml-profiler> (Accessed: 29 November 2023).

- [10] Infineon Technologies. (2022, October 10). ModusToolbox™ Machine Learning Configurator user guide. Bangalore
- [11] Infineon Technologies (no date a) BGT60TR13C 60 GHz Radar Sensor . Available at www.infineon.com/dgdl/Infineon-DS_BGT60TR13C-DataSheet-v02_46-EN.pdf?fileId=8ac78c8c7d718a49017d94bac88e5d43 (Accessed: 03 December 2023).
- [12] Infineon Technologies (no date) CY8CPROTO-062-4343W . Available at: www.infineon.com/dgdl/Infineon-CY8CPROTO-062-4343W_PSoC_6_Wi-Fi_BT_Prototyping_Kit_Guide-UserManual-v01_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0f0118571844 (Accessed: 17 September 2023).
- [13] AG, I.T. (no date) Artificial-intelligence - infineon technologies, artificial-intelligence - Infineon Technologies. Available at: <https://www.infineon.com/cms/en/product/promopages/artificial-intelligence/> (Accessed: 03 December 2023).
- [14] AG, I.T. (no date b) ModusToolbox™ for Machine Learning, Infineon Technologies. Available at: <https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/modustoolbox-machine-learning/> (Accessed: 03 December 2023).
- [15] Our products are designed to make Edge Ai Easy. (no date) Imagimob. Available at: <https://www.imagimob.com/products> (Accessed: 03 December 2023).
- [16] Infineon psoc™ 6 Wi-Fi Bt Pioneer kit (2022) SensiML. Available at: <https://sensiml.com/supported-platforms/board/infineon-cypress-psoc6-mcu/> (Accessed: 03 December 2023).
- [17] Infineon CY8CKIT-062S2 pioneer kit (no date) Infineon CY8CKIT-062S2 Pioneer Kit - Edge Impulse Documentation. Available at: <https://docs.edgeimpulse.com/docs/development-platforms/officially-supported-mcu-targets/infineon-cy8ckit-062s2> (Accessed: 03 December 2023).
- [18] Kulkarni, U. et al. (2021). AI Model Compression for Edge Devices Using Optimization Techniques. In: Gunjan, V.K., Zurada, J.M. (eds) Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough. Studies in Computational Intelligence, vol 956. Springer, Cham. https://doi.org/10.1007/978-3-030-68291-0_17
- [19] Get started Onnx (no date) ONNX. Available at: <https://onnx.ai/get-started.html> (Accessed: 06 October 2023).

- [20] Tensorflow Lite: ML for Mobile and edge devices (no date) TensorFlow. Available at: <https://www.tensorflow.org/lite> (Accessed: 09 August 2023).
- [21] PYTORCH documentation¶ (no date) PyTorch documentation - PyTorch 2.1 documentation. Available at: <https://pytorch.org/docs/stable/index.html> (Accessed: 15 September 2023).
- [22] *CMSIS components* (no date) *Introduction*. Available at: https://arm-software.github.io/CMSIS_5/General/html/index.html (Accessed: 20 November 2023).
- [23] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen and K. C. Tan, "A Survey on Evolutionary Neural Architecture Search," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 550-570, Feb. 2023, doi: 10.1109/TNNLS.2021.3100554. keywords: {Computer architecture;Optimization;Convolutional neural networks;Search problems;Neural networks;Deep learning;Statistics;Deep learning;evolutionary computation (EC);evolutionary neural architecture search (NAS);image classification}.
- [24] Krishna Teja Chitty-Venkata and Arun K. Somani. 2022. Neural Architecture Search Survey: A Hardware Perspective. *ACM Comput. Surv.* 55, 4, Article 78 (April 2023), 36 pages. <https://doi-org.elibrary.nirmauni.ac.in/10.1145/3524500>
- [25] GeeksforGeeks (2024) Neural Architecture Search algorithm, GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/neural-architecture-and-search-methods/> (Accessed: 19 May 2024).
- [26] *Neural Architecture Search: Everything you need to know* (2024) *Deci*. Available at: <https://deci.ai/neural-architecture-search/> (Accessed: 19 May 2024).
- [27] K. T. Chitty-Venkata, M. Emani, V. Vishwanath and A. K. Somani, "Neural Architecture Search Benchmarks: Insights and Survey," in *IEEE Access*, vol. 11, pp. 25217-25236, 2023, doi: 10.1109/ACCESS.2023.3253818.

22mece08_MajorProject.pdf

by PRAJAPATI DHRUVIL

Submission date: 22-May-2024 11:42AM (UTC+0530)

Submission ID: 2105910491

File name: 22MECE08_MajorProject.pdf (1.66M) **Word count:** 9695

Character count: 19289

22MECE08_MajorProject.pdf

ORIGINALITY REPORT

3%

SIMILARITY INDEX

0%

INTERNET SOURCES

0%

PUBLICATIONS

1%

STUDENT PAPERS

PRIMARY SOURCES

1	"Neural Architecture Search Survey: A Hardware Perspective" ACM Computing Surveys, 2022	2 %
	Student Paper	
2	en.teknopedia.teknokrat.ac.id	1 %
	Internet Source	
3	www.infineon.com	<1 %
	Internet Source	

Exclude quotes On

Exclude matches < 6 words

Exclude bibliography On