Sensorless Pattern Recognition and Anomaly Detection of PMSM/BLDC Motors at the Edge using Intelligent Neural Processing

Thesis

Submitted for the partial fulfillment of the requirement for the Degree of

Master of Technology in Intelligent Automation and Robotics

Department of Electronics and Telecommunication Engineering,

Jadavpur University

By

Subharshi Roy

Registration Number: **160234** of **2021-22**

Examination Roll Number: **M6IAR24016**

Under the guidance of

Prof. Amit Konar

Dept. of Electronics & Telecommunication Engineering

Jadavpur University, Kolkata-700032,

West Bengal, India

FACULTY OF ENGINEERING AND TECHONOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE

This is to certify that that the dissertation entitled "**Sensorless Pattern Recognition and Anomaly Detection of PMSM/BLDC Motors at the Edge using Intelligent Neural Processing**" has been carried out **SUBHARSHI ROY** (University Registration Number **160234** of **2021-22**, Examination Roll Number: **M6IAR24016**) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Intelligent Automation and Robotics offered by the Department of Electronics & Telecommunication Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other University or Institute.

_____

Prof. Amit Konar
Supervisor,
Dept. of Electronics &
Telecommunication Engineering,
Jadavpur University

_____

Prof. Manotosh Biswas
Head of the Department,
Dept. of Electronics &
Telecommunication Engineering,
Jadavpur University

_____

Prof. Dipak Laha
Dean,
Faculty Council of Engineering and Technology,
Jadavpur University

FACULTY OF ENGINEERING AND TECHONOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The forgoing thesis is hereby approved as a creditable study of an engineering subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.

Committee on Final Examination
for the Evaluation of the Thesis

_____
Signature of the Examiner

_____
Signature of the Supervisor

*Only in case the thesis is Approved

FACULTY OF ENGINEERING AND TECHONOLOGY
JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC THESIS

I hereby declare that the thesis entitled "**Sensorless Pattern Recognition and Anomaly Detection of PMSM/BLDC Motors at the Edge using Intelligent Neural Processing**" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Intelligent Automation and Robotics offered by the Department of Electronics & Telecommunication Engineering.

All information has been obtained and presented with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: SUBHARSHI ROY

Class Roll No: 002110704018

Thesis Title: Sensorless Pattern Recognition and Anomaly Detection of PMSM/BLDC Motors at the Edge using Intelligent Neural Processing

_____

Signature of the Candidate

ACKNOWLEDGEMENT

First and foremost, I extend my deepest gratitude to my supervisor, Prof. Amit Konar from the Department of Electronics and Telecommunication Engineering, for graciously guiding my journey towards earning my Master of Technology degree in Intelligent Automation and Robotics. His unwavering support and mentorship have been instrumental in bringing this work to fruition.

I am also immensely thankful to my mentor, Mr. Nabarun Dasgupta from STMicroelectronics, for his trust in my abilities and for fostering an environment of creativity and exploration during my internship. Our insightful discussions have left an indelible mark on my professional growth.

I extend heartfelt appreciation to Mr. Vipin Kumar Gupta and Mr. Sujay Kumar Singh from STMicroelectronics. Their expertise and guidance have been invaluable in shaping the course of my research and professional development.

I am deeply grateful to Ms. Sayantani Ghosh, PhD Research Scholar from Jadavpur University, and Mr. Dipayan Chatterjee, PhD Research Scholar from IIT Kharagpur, for their invaluable assistance and encouragement throughout this journey.

Finally, to my family - my parents, my elder brother Dada, and my beloved little brother Gulu - I owe a debt of gratitude for their unwavering love, support, and guidance throughout my academic pursuits and life.

Subharshi Roy

PREFACE

The intersection of electrical engineering and artificial intelligence has revolutionized industrial applications in the field of intelligent systems. Electric motors are the heart of many industrial and automotive processes, and efficiency and safety heavily depend on monitoring their performance. This thesis explores the field of motor-actuated control systems and presents a novel neural network framework for analysis and diagnosis in resource-constrained edge-computing devices such as microcontrollers. The framework takes into account a wide range of anomalies, including uneven loads, internal control parameter mismatches, and external aerodynamic perturbations. These elements have the potential to cause motor damage, erratic behavior, and performance fluctuations. This framework is unique in that it does not require any additional physical sensors; instead, it relies only on the operational electrical signatures of typical motor control algorithms, effectively making the motor act as a sensor. Advanced sensorless vector control and pattern recognition algorithms powered by deep learning techniques, all functioning inside edge device boundaries, are at the core of this system. Because of this, real-time examination of motor health and operational status is possible without the computational load that cloud computing is known for. Through innovative architecture and compression techniques designed for devices with limited resources, such as microcontrollers, this thesis creates opportunities for realistic and affordable implementation in a variety of applications.

The thesis comprises of nine chapters. Chapter 1, Introduction, provides an introductory perspective on the thesis and the framework for diagnosing electric motor-driven systems with sensorless state estimation and deep learning on edge devices for real-time anomaly detection and predictive maintenance. Chapter 2, Sensorless Field Oriented Control of PMSM/BLDC Motors, delves into the fundamentals of Field-Oriented Control (FOC) and its application in controlling BLDC and PMSM motors, providing a comprehensive understanding of the underlying principles. Chapter 3, Convolution Neural Network for Pattern Recognition, here, the focus shifts to Convolutional Neural Networks (CNNs) and their role in pattern recognition, particularly in the context of electric motor control systems. Chapter 4, Neural Network Model Compression for Edge Devices – Quantization, this

chapter explores techniques for compressing neural network models for deployment on edge devices, facilitating efficient real-time inference. Chapter 5, Intelligent Anomaly Detection for Predictive Maintenance, in this chapter examination of anomaly detection techniques and their application in predictive maintenance scenarios, with a focus on identifying anomalies in electric motor control systems. Chapter 6, Experiment - Sensorless Pattern Recognition and Anomaly Detection in the Intelligent Edge Device (seNNsEdge), in this chapter presents experimental results and findings from implementing the proposed framework on an intelligent edge device, demonstrating its effectiveness in real-world scenarios. Chapter 7, Expanding the Horizons of Edge Computing: Stacked Convolutional Auto Associative Neural Network for 3-phase Shunt Current Measurement Denoising, this chapter explores the application of edge computing in denoising shunt current measurements, expanding the capabilities of edge devices in electric motor control systems. Chapter 8, Conclusion, summarizes the thesis work and its key findings. Chapter 9, Future Scope, explores potential future developments and the broad experimental possibilities stemming from this research.

Table of Contents

Chapter1: Introduction

1.1 Introduction …… 1
1.2 Schematic Summary ……..… 5
1.3 Thesis Objective …… 6
1.4 Contributions ……. 6
1.5 Scope of the Work ……. 7
1.6 References ……..…. 9

Chapter 2: Sensorless Field Oriented Control of PMSM/BLDC Motors

2.1 Introduction ……. 11
2.2 Permanent Magnet Synchronous Motors / Brushless DC Motors Control Fundamentals …… 12
2.3 Field Oriented Control Mechanism …… 12
2.4 Space Vector Transformations ……..... 13
2.4.1 Clarke Transformation ……..… 15
2.4.2 Park Transformation …… 15
2.4.3 Inverse Park Transformation …… 16
2.4.4 Inverse Clarke Transformation …… 17
2.5 Field Oriented Control Operation…… 17
2.6 Conclusion …….. 21
2.7 References …….. 22

Chapter 3: Convolution Neural Network for Pattern Recognition

3.1 Introduction …… 23
3.2 Origins of CNN ……..… 24
3.2.1 Biological Inspiration of CNN …… 24
3.2.2 CNN for Computer Vision …….. 27
3.3 CNN for Pattern Recognition – Computer Vision Inspired Signal Pattern Recognition (for Classification and Anomaly Detection tasks) …… 29
3.3.1 Image Data as Input to the CNN …….. 29
3.3.2 Real-time Signal Waveforms Features Stacked to form an Image-like Array of Pixels …….. 30
3.4 Convolution Filters and Operations …… 33
3.4.1 Convolution Kernels …… 34

3.4.2 Kernel Vs Filters 34
3.4.3 Filter Application Example 35
3.4.4 Feature Extraction 36
3.4.5 Padding Operation 37
3.4.6 Striding Operation 38
3.4.7 Pooling Layers in CNN 39
3.4.7.1 Average Pooling 40
3.4.7.2 Maximum Pooling 40
3.4.7.3 Stochastic Pooling 41
3.5 Conclusion 43
3.6 References 44

Chapter 4: Neural Network Model Compression for Edge Devices – Quantization
4.1 Introduction 48
4.2 Numerical Representation and their influence in Neural Network Training and Inference 49
4.2.1 Integers 49
4.2.2 Floating-point number 50
4.2.3 Understanding FLOPS – Floating Point Operations and their Influence in Neural Networks 51
4.3 Quantization in Neural Networks 53
4.4 Quantization Types 54
4.4.1 Asymmetric Quantization: 54
4.4.1.1 Asymmetric Quantizer Input 55
4.4.1.2 Asymmetric Quantization equation: 55
4.4.1.3 Asymmetric Quantization Output 55
4.4.1.4 Dequantization Equation 55
4.4.1.5 Asymmetric Dequantization Output 56
4.4.2 Symmetric Quantization 56
4.4.2.1 Symmetric Quantizer Input 56
4.4.2.2 Symmetric Quantization equations 56
4.4.2.3 Symmetric Quantization Output 57
4.4.2.4 Dequantization Equation 57
4.4.2.5 Symmetric Dequantization Output 57
4.4.3 Comparing Asymmetric and Symmetric Quantization 57
4.5 Quantization Granularity 58
4.5.1 Per-Channel Quantization 58
4.6 Quantization Methodologies 60

4.6.1 Quantization Aware Training 60
4.6.2 Post-Training Quantization 61
4.7 Experiment - Model Compression for STM32 Microcontrollers for Real-time Inference 61
4.7.1 Model Compression of a Neural Net 61
4.8 Conclusion 62
4.9 References 64

Chapter 5: Intelligent Anomaly Detection for Predictive Maintenance
5.1 Introduction 66
5.2 Predictive Maintenance in the current Industrial Revolution 67
5.2.1 Evolution of Maintenance techniques 68
5.2.1.1 Reactive / Breakdown Maintenance 69
5.2.1.2 Preventive Maintenance 69
5.2.1.3 Predictive Maintenance 69
5.3. Anomaly Detection for Predictive Maintenance using Machine Intelligence 70
5.3.1 Anomalies in Time Series Data 70
5.3.1.1 Point Anomaly 71
5.3.1.2 Contextual Anomaly 71
5.3.1.3 Collective Anomaly 71
5.3.1.4 Challenges in Multivariate Time-Series Data 72
5.4 Types of Anomalies Targeted to Detect in Aerial Vehicle Motor Control Applications 72
5.4.1 Aerodynamic anomaly 72
5.4.1.1 Quadrotor Aerodynamic Anomaly 74
5.4.1.2 Rotor Aerodynamic Disturbance Model 75
5.4.2 Control algorithm based parameter mismatch or Uncontrollability anomaly 77
5.4.3 Mechanical Asymmetry based anomaly 78
5.5 Conclusion 80
5.6 References 81

Chapter 6: Experiment - Sensorless Pattern Recognition and Anomaly Detection in the Intelligent Edge Device (seNNsEdge)
6.1 Introduction 83
6.2 Schematic Summary / Architecture 84
6.3 Experimental Setup 86
6.3.1 Sensorless control of PMSM/BLDC Motor using Field Oriented Control (FOC) Algorithm 86

6.3.2 STEVAL-PTOOL3A 87
6.3.3 STM32-H755ZI-Q (Edge Inference System) 89
6.3.4 Anomaly Generation Experimental Setup 91
6.3.4.1 Artificial Aerodynamic Disturbance Generator 92
6.3.4.2 Uncontrollability Anomaly 94
6.3.4.3 Asymmetrical Load-based anomalies 94
6.4 Data Acquisition Phase for acquiring Training Data 95
6.5 Comprehensive Data Acquisition for Aerodynamic, Uncontrollability, and Asymmetry-Based Anomalies 99
6.5.1 Acquiring Data for Healthy Operation 100
6.5.2 Acquiring Data for Aerodynamic Anomaly 101
6.5.3 Acquiring Data for Uncontrollability Anomaly 102
6.5.4 Acquiring Data for Asymmetric Broken Propeller Anomaly 103
6.6 Neural Network Architecture for Anomaly Detection 104
6.6.1 Input Layer 104
6.6.2 Feature Extractor 104
6.6.3 Max Pooling Layer 106
6.6.4 Flatten Layer 106
6.6.5 Dense / Fully Connected Layers 107
6.6.6 Rectified Linear Unit 107
6.6.7 Output Layer 107
6.6.8 Softmax Activation Function 107
6.7 Training the Neural Net for Anomaly Classification / Motor Health Estimation / Diagnosis 108
6.7.1 Training Methodology 108
6.7.2 Defining Callbacks 108
6.7.3 Compiling the Model 109
6.7.4 Training the Model 109
6.7.5 Training Hardware 110
5.7.6 Training Metrics – Training Loss and Accuracy 111
6.8 Results – CNN Model Performance Validation 112
6.8.1 Classification Metrics 113
6.9 Realization of the Trained Neural Net on the Edge Device 117
6.9.1 Model Compression – Tensorflow Quantization 117
6.9.2 Optimized C-kernel Conversion using STM32 AI Tools 118
6.9.3 Model metrics on the Edge Device 119
6.10 Conclusion 121
6.11 References 121

Chapter 7: Expanding the Horizons of Edge Computing: Stacked Convolutional Auto Associative Neural Network for 3-phase Shunt Current Measurement Denoising
7.1 Introduction 122
7.2 Introduction to Auto Associative Neural Networks or Autoencoders 124
7.3 Denoising Autoencoder Algorithm 125
7.4 Experiment and Results – Training a Convolution Neural Network based Stacked Autoencoder to Denoise Shunt Sensing like Simulation Data 126
7.4.1 Denoising Reconstruction Results 131
7.5 Performance Metrics and Interpretation of the ConvAutoencoder Model 134
7.6 Experiment and Results – Real time Inference in the Edge Device STM32H755 Microcontroller to Denoise Noisy Signal 135
7.7 Conclusion 136
7.8 References 137

Chapter 8: Conclusion 138

Chapter 9: Future Scope of Work 140

List of Figures

Fig. 1.1	Schematic Summary	5
Fig. 2.1	Stator current space vector and its component in (A, B, C)	14
Fig. 2.2	Stator current space vector and its components in (α, β)	15
Fig. 2.3	Stator current space vector and its component in (α, β) and in the d,q rotating reference frame	16
Fig. 2.4	Field Orinted Control Block Diagram STM32 MCSDK	18
Fig. 2.5	Field Orinted Control Block Diagram	18
Fig. 2.6	Current, voltage, and rotor flux space vectors in the d,q rotating reference frame and their relationship with a,b,c and α, β stationary reference frame.	20
Fig. 3.1	Neurophysiologists David Hubel (left) and Torsten Wiesel (right)	24
Fig. 3.2	An partially anesthetized cat is equipped with contact lenses to ensure the eyes remain focused on a screen. Images are then projected onto the screen, and neuron responses are recorded using an extracellular electrode	25
Fig. 3.3	Fig. 3.3 illustrates the selective response of a neuron in the primary visual cortex to line segments. (a) An anesthetized cat is equipped with contact lenses to ensure precise eye fixation on a screen. Images are then projected onto the screen, and neuron responses are recorded using an extracellular electrode. (b) The neuron typically exhibits robust responses to a bar of light oriented at a specific angle, with minimal or no response to other orientations. (c) The spike rate of the neuron varies as the orientation of the stimulation changes, as depicted in the curve	25 - 26
Fig. 3.4	Architecture of Neocognitron	27
Fig. 3.5	AlexNet architecture	28
Fig. 3.6	Arbitrary waveforms of Sin, Square, Triangular, Sawtooth and Rippled Waveforms.	31
Fig. 3.7	Five samples considered for 5x5 dimension Input Image	32
Fig. 3.8	Heatmap representation to create a mental image for Input image	32
Fig. 3.9	Kernel Operation to Input Images	34

Fig. 3.10	Different Filters applied to an original image and their results	35
Fig. 3.11	Visualization of Feature Extraction	37
Fig. 3.12	Padding in a Kernel	38
Fig. 3.13	Example of Strided Convolution	39
Fig. 3.14	Illustration of Average Pooling	40
Fig. 3.15	Illustration of Maximum Pooling	41
Fig. 3.16	Illustration of Stochastic Pooling	42
Fig. 4.1	A Neural Net Example Picture	54
Fig. 4.2	Quantization and Dequantization w.r.t I/P vs O/P Layer of a Neural Network	54
Fig. 4.3	Asymmetric Quantization	55
Fig. 4.4	Symmetric Quantization	56
Fig. 5.5	Acquired Data - Motor Control Parameters used for Neural Network Training – Alpha-Beta Current and Voltage	58
Fig. 4.6	Quantization Granularity	59
Fig. 4.7	Quantization Aware Training	60
Fig. 4.8	Post-Training Quantization	61
Fig. 4.9	Compressed Model Size – 20344 Bytes	62
Fig. 4.10	Uncompressed Model Size – 239936	62
Fig. 5.1	Different maintenance techniques and their effectiveness	68
Fig. 5.2	Types of Anomaly in Time-Series Data	71
Fig. 5.3	Description of the detailed classification of anomalies in time-series data	73
Fig. 5.4	Quadrotor-Unmanned Aerial Vehicle Dynamics	74
Fig. 5.5	Rotor Aerodynamic Disturbance Model	75
Fig. 5.6	Aerodynamics Direction	77
Fig. 5.7	Experimental assessment of position sensor breakdown faults, specifically at no-load operation, utilizing a misaligned piezoelectric sensor. The investigation involved analyzing waveform outputs from the piezoelectric sensor PSI, frequency spectra, and narrow frequency spectra	79

Fig. 6.1	Overview of the framework	83
Fig. 6.2	Schematic Architecture of the framework	85
Fig. 6.3	Quadcopter Motor driven by PTOOL3A	86
Fig. 6.4	Quadcopter Motor driven by STEVAL-PTOOL3A Motor Driver Board	87
Fig. 6.5	STEVAL-PTOOL3A Motor Control Board	88
Fig. 6.6	STM32-H755ZI-Q Nucleo Board	90
Fig. 6.7	Types of anomalies subjected to the Quadcopter Motor	91
Fig. 6.8	Conjunction between the Artificial Aerodynamic Disturbance Generator and the Quadcopter Motor	92
Fig. 6.9	Experimental Setup for Aerodynamic Anomaly Generation	93
Fig. 6.10	Aerodynamic Disturbance based Anomaly being subjected to the Quadcopter motor	93
Fig. 6.11	Different types of healthy and damaged propellers	95
Fig. 6.12	Data Extraction from the FOC Operated Motor Driver	96
Fig. 6.13	Data buffer extracted from FOC Control Electrical Signatures	96
Fig. 6.14	Electrical Signature Extraction from PTOOL3A Board	98
Fig. 6.15	Databuffer acquired from source PTOOL3A Motor Driver to PC as a CSV file	99
Fig. 6.16	Healthy Waveform Waveforms	100
Fig. 6.17	Aerodynamic induced anomaly waveform	101
Fig. 6.18	Uncontrollability induced anomaly waveform	102
Fig. 6.19	Asymmetric propeller / broken propeller induced anomaly	103
Fig. 6.20	Tesla T4 GPU architecture	110
Fig. 6.21	Training hardware details using !nvidia-smi command	111
Fig. 6.22	Training Accuracy and Loss Waveform respectively	112
Fig. 6.23	Validation metrics results	115
Fig. 6.24	ROC curve as validation metric	115
Fig. 6.25	Precision-Recall Curve as validation metric	116
Fig. 6.26	Confusion Matrix as validation metric	116

Fig. 6.27	Tensorflow to Tensorflow Lite Conversion	118
Fig. 6.28	Representation from Quantization layer to Deployed Layer	119
Fig. 6.29	Figure displays the Model size – Content Length	120
Fig. 7.1	Field Oriented Control of PMSM/BLDM Motor Control Loop	123
Fig. 7.2	Green colored waveform displays the noisy shunt current which is eventually fed into ADC	123
Fig. 7.3	ADC Sensing Circuit in PMSM/BLDM Motor Controller – STEVAL PTOOL4A	124
Fig. 7.4	Autoencoder graphical representation	125
Fig. 7.5	Denoising Autoencoder - I. Goodfellow, Y. Bengio, A. Courville	126
Fig. 7.6	PMSM Motor driven by STEVAL-PTOOL4A Motor Driver Board	127
Fig. 7.7	Waveform of Dataset Pure (Up) vs Gaussian Noise induced Noisy Signal (Down)	128
Fig. 7.8	Training Architecture for Convolutional Autoencoder	129
Fig. 7.9	Architecture of the Convolutional Autoencoder Neural Network	130
Fig. 7.10	Training Loss Waveform for 86 Epochs	131
Fig. 7.11	Reconstruction performance of Convolutional Autoencoder	
Fig. 7.12	Reconstruction performance of Convolutional Autoencoder via Spectogram	132
Fig. 7.13	3D Spectogram Representing Results	133
Fig. 7.14	Results and Interpretation results	134
Fig. 7.15	Real Inference Convolutional Autoencoder running on STM32H755 MCU	135

List of Tables

Table 6.1	CNN architecture for Motor Anomaly Detection	105
Table. 6.2	Mathematical Operations and Variables Explained	106
Table. 6.3	Confusion Matrix	114
Table. 6.4	Memory Consumption for CNN C-kernel	120

Chapter 1

Introduction

1.1 Introduction

In the world of intelligent systems, combining electrical engineering with machine learning has revolutionized industrial applications [1]. Electric motors drive many industrial and automotive processes, and monitoring their performance is crucial for efficiency and safety [2]. Detecting anomalies, monitoring conditions, and predicting maintenance needs are vital tasks in this domain [3]. Brushless DC Motors and Permanent Magnetic Synchronous Motors are particularly favored for their efficiency and torque-to-weight ratio, making them ideal for various applications like aerial drones, terrestrial vehicles, and power tools [4].

This thesis introduces an innovative framework aimed at diagnosing and analyzing electric motor-driven control systems. It tackles both external and internal influences, such as aerodynamic effects [5-7], control parameter mismatches leading to uncontrollability [8], and loads causing shaft asymmetry [9]. External aerodynamic forces affecting a system, such as a quadcopter motor in operation, are highly vulnerable to external aerodynamic perturbations. These disruptions introduce intricate nonlinear dynamics that pose formidable challenges for the quadcopter motor, often culminating in motor damage and eventual failure. The motor's performance and health may be impacted by the misadjustment in control parameters [8], which can cause performance fluctuations ranging from small deviations to significant uncontrollable behaviors [10]. Predictive maintenance strategies and system behavior analysis become more challenging under such operating conditions because of the unpredictable irregularities they give rise to. Also, uneven loads worsen rotary misalignment problems, accelerating the deterioration of motor health and encouraging the creation of aberrant vibrations, all of which can seriously reduce operational dependability and efficiency [9].

What's unique is that it relies solely on traditional motor control algorithm's operational electrical signatures, the current and voltage measurements originating from shunt current measurements. Hence, no additional physical sensors like inertial or motion sensors are needed for health determination of the motors.

The foundation of this framework lies in its advanced sensorless state estimation and pattern recognition technique, driven by cutting-edge deep learning methodologies operating in the edge device. It harnesses the electrical signatures generated by the motor control algorithm, the field-oriented control (FOC) algorithm, to analyze the motor's health and operational status. The analysis is conducted in real-time through an intelligent neural algorithm, operating concurrently alongside the control algorithm within the edge device.

While traditionally cloud computing has been relied upon for neural network inference due to the computational and memory demands of these algorithms, this thesis pioneers an architecture and compression methodologies tailored for resource-constrained devices like microcontrollers. By optimizing the neural network execution to operate efficiently on microcontroller platforms, it opens avenues for practical and cost-effective implementation in diverse applications.

The core setup features a traditional motor drive system STDES-PTOOL3A [11] outfitted with an STM32G4 ARM Cortex M4 [12] microcontroller, specifically tailored for controlling the BLDC/PMSM motor. The electrical signatures stem from the control algorithm embedded within the conventional motor driver, such as Field Oriented Control (FOC), as facilitated by established frameworks like the STMicroelectronics Motor Control Software Development Kit (MCSDK). Field-oriented control relies on the sensing of 3-phase currents and rotor position estimation. The algorithm executes a 3-phase to 2-phase AC (Clarke) transformation followed by a Stationary D-Q (Park) transformation [14]. Subsequently, a closed-loop control is enacted to orient and regulate torque-producing q-axis

current while minimizing d-axis current. Moreover, the controller is equipped with the ability to measure the back-electromotive forces generated in the motor windings.

Aerodynamic, uncontrollability, and asymmetry-based anomalies are intentionally introduced into a functioning quadcopter setup to analyze their effects, discern patterns, and gather relevant datasets. All these datasets serve the purpose of identifying patterns and categorizing both external and internal influences encountered during the operation of the electric motor, thereby reducing the necessity for additional inertial, motion, or audio-based sensors. Platforms with higher computational capabilities are utilized for the analysis and training of these datasets [15]. However, the conventional controller can leverage this learning to implement control and classification algorithms on the traditional edge device. These control parameters shoulder the primary responsibility of conducting feature engineering. The system leverages the external influences encountered by the electric motor during operation to comprehend the dynamic behaviors occurring in real-time, thereby enriching unique signal patterns and reducing the need for additional inertial, motion, or audio-based sensors. Moreover, the system maximizes the utilization of existing hardware and parameters for pattern recognition and anomaly detection. This ability to interpret external aerodynamic influences acting on the motor during operation suggests a concept wherein the motor itself acts as a sensor. Similarly, any anomaly or deviation from the motor's healthy operation can be detected early before causing significant damage, thanks to the motor driver's capability coupled with the edge computing device to perform real-time intelligent neural inference with low latency and high efficiency.

At the heart of this sensorless estimation process lies a Quantization-Aware trained Deep Convolutional Neural Network (DCNN) algorithm, strategically operating on the extreme edge, within the motor drive controller. The Field Oriented Control (FOC) algorithm running on the PTOOL3A motor drive system plays a crucial role in maintaining stable operating conditions, making it indispensable to the process. This intelligent algorithm, developed to diagnose and classify the dynamic responses of the motors to external influences, runs

concurrently on the same controller. Its integration enhances the application of Edge AI for condition monitoring and pattern recognition.

The implementation of the Deep CNN architecture within the custom layer of Keras [16], utilizing the Tensorflow backend [17], involved extensive experimentation to fine-tune hyperparameters and ensure flexibility in handling various data patterns while maintaining efficient and high-precision pattern recognition capabilities. Numerous experiments were conducted with diverse data patterns to achieve an optimal balance between low latency and high precision at the edge device. These experiments aimed to recognize anomalies and patterns, achieving remarkable levels of Precision, Recall, F1-Score, Matthews Correlation Coefficient, ROC Curve, Precision-Recall Curve and Confusion Matrix leveraging parameters utilized by traditional motor control algorithms. The utilization of electric motors and motor controllers spans a wide range of applications, showcasing their versatility and importance. The prospect of further advancing these technologies to equip motors with the capability to function as sensors, endowed with intelligent observability capabilities, holds significant potential to contribute to the ongoing cyber-physical industrial revolution.

In this thesis, we delve beyond the mere application of sensorless anomaly detection at the edge, to unveil the extensive potential of edge computing. We introduce a cutting-edge approach where an edge microcontroller device, empowered by intelligent neural algorithms, replaces traditional hardware filters. This innovation optimizes the denoising process of 3-phase motor shunt current, thereby enhancing ADC sampling capabilities significantly

1.2 Schematic Summary

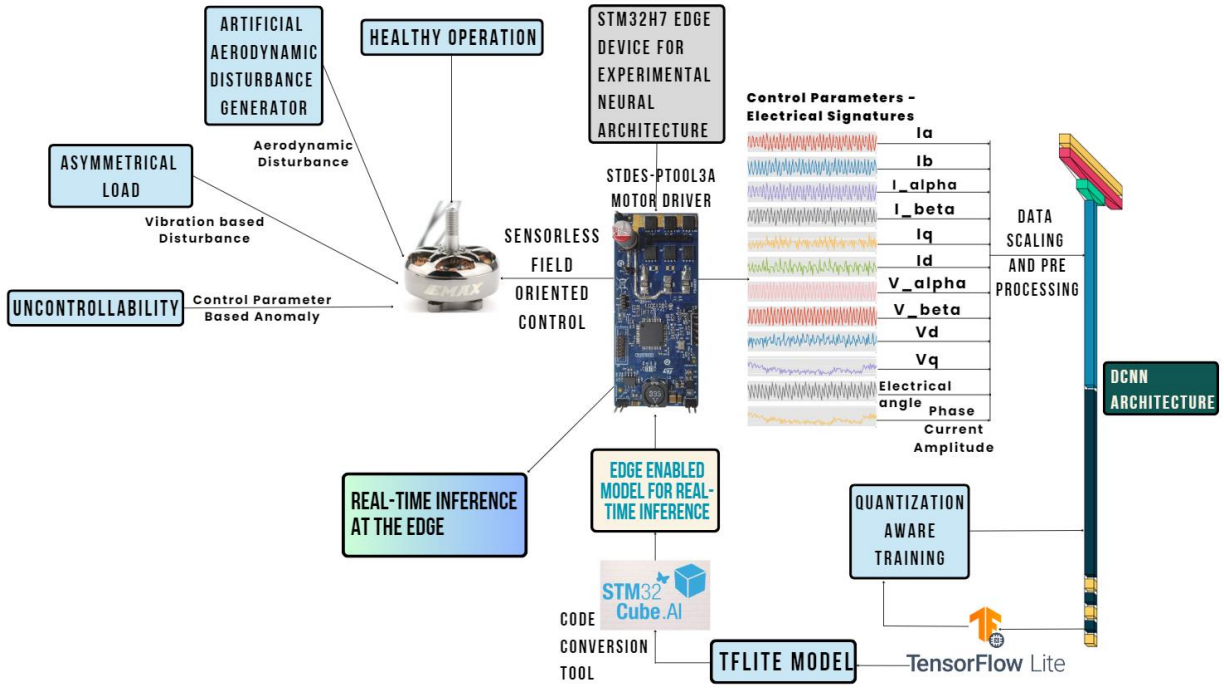


Fig 1.1 Schematic Summary

Figure 1.1 illustrates the schematic summary of the project architecture. In the subsequent chapters and experiments, this schematic architecture will be used as a reference and will be explained in detail with respect to individual operations. For now, let us discuss the overview of the schematic architecture. Uncontrollability, Asymmetrical load, Aerodynamic Disturbance, and Healthy Operation are the operating conditions of the BLDC/PMSM motor. The motor is controlled using sensorless vector control, i.e., field-oriented control (FOC) algorithm, using the STEVAL-PTOOL3A Motor Driver. The STEVAL-PTOOL3A Motor Driver's FOC algorithm leverages electrical signatures such as 3-phase currents (Ia, Ib, Ic) and their respective Clarke and Park transformations (I_alpha, I_beta, I_d, I_q), and similar transformations for back-emf (V_alpha, V_beta, V_d, V_q), Electrical Angle, and Phase current amplitude for its operation.

These electrical signatures are extracted to train a Deep Convolution Neural Network (DCNN) algorithm for pattern recognition. The training is performed in a Quantization Aware manner for model compression using Tensorflow Lite. The compressed model is converted into C-kernel to be executed on an STM32H7 microcontroller, which performs inference in real-time. Additionally, the STEVAL-PTOOL3A communicates with the STM32H7 edge device via the I2C protocol at a 1MHz transmission speed for real-time anomaly detection.

1.3 Thesis Objective

The primary objective of this thesis is to develop a robust methodology that harnesses the inherent electrical signatures generated by the Field-Oriented Control (FOC) algorithm for real-time diagnosis and classification of motor operating conditions and impactful anomalies. Specifically, the focus lies on leveraging deep learning techniques and algorithms at the edge to enable accurate detection of abnormal patterns indicative of motor degradation or malfunction. The objective also includes exploration towards deep learning model compression techniques to integrate intelligent neural algorithms into application-specific hardware consisting of low memory and operating power, enabling them to behave intelligently. Finally, the thesis focuses on other applications of Edge AI in the domain of signal denoising by replacing hardware filters with intelligent neural algorithms operating at the edge.

1.4 Contributions

This thesis makes several significant contributions to the field:

- Development of a framework for sensorless pattern recognition and anomaly detection in electric motor control systems, leveraging the intrinsic electrical signatures of Brushless DC and Permanent Magnet Synchronous motors and traditional motor control algorithms.

- Integration of deep learning algorithm and methodologies such as Convolution Neural Network at the edge to facilitate real-time diagnosis and classification of motor operating conditions, thereby enhancing predictive maintenance capabilities.

- Exploration of the potential for electric motors themselves to function as sensors, enabling early detection of anomalies without the need for additional physical sensors through the concurrent operation of neural algorithms at the edge device alongside the control algorithm.

- Significant reduction in memory footprint through the compression of memory-intensive neural network algorithms. This advancement addresses the challenge of deploying complex neural networks on resource-constrained platforms, paving the way for widespread adoption of deep learning techniques in edge computing applications.

- Exploring potential applications of neural algorithms in edge computing for signal denoising by utilizing Stacked CNN Autoencoders to eliminate hardware filters, thereby enhancing signal processing efficiency in the edge device.

1.5 Scope of the Work

This thesis endeavors to construct a comprehensive methodology for the diagnosis and examination of electric motor-driven control systems, concentrating particularly on Brushless DC Motors (BLDC) and Permanent Magnet Synchronous Motors (PMSM). It encompasses an investigation into both external and internal factors impacting motor performance, including aerodynamic effects, control parameter discrepancies, and load-induced asymmetry. The proposed framework hinges solely on the operational electrical signatures produced by traditional motor control algorithms, notably the Field Oriented Control (FOC) algorithm. These signatures, derived from shunt current measurements, are fundamental to determining motor health. Central to this framework is the utilization of advanced sensorless state estimation and pattern recognition techniques driven by deep learning methodologies.

Specifically, a Quantization-Aware trained Deep Convolutional Neural Network (DCNN) algorithm is employed for real-time analysis of motor health and operational status. Notably, this analysis is conducted within the edge device, alongside the control algorithm, eliminating the need for additional physical sensors. The research explores innovative strategies to address computational limitations associated with edge computing, pioneering architectures and compression methodologies tailored for resource-constrained devices such as microcontrollers. By optimizing neural network execution for microcontroller platforms, the thesis aims to facilitate practical and cost-effective implementation across various applications. Experimental setups involve deliberately introducing aerodynamic, uncontrollability, and asymmetry-based anomalies into functioning quadcopter setups to assess their impact and gather pertinent datasets. These datasets aid in identifying patterns and categorizing external and internal influences encountered during motor operation. Moreover, the thesis investigates the potential for electric motors to serve as sensors themselves, enabling the early detection of anomalies to prevent significant damage. This is achieved through the simultaneous operation of neural algorithms at the edge device alongside the control algorithm. In summary, the scope encompasses the development of a robust methodology for sensorless anomaly detection and pattern recognition in electric motor control systems, with a specific focus on real-time diagnosis and classification of motor operating conditions. Additionally, the research explores the integration of deep learning techniques at the edge to enhance predictive maintenance capabilities and signal denoising processes, contributing to the advancement of edge computing applications in the industrial domain.

1.6 References

[1] *Application of Artificial Intelligence Technology in Automatic Control of Electrical Engineering*. (2021, November 22). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9696536

[2] *Electrical Motor Maintenance Techniques and Life Cycle Assessment- A Review with Case Studies*. (2019, August 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9036543/

[3] Júnior, F. F., Romero, R. A. F., & Hsieh, S. (2023, December 9). *Machine Learning for the Detection and Diagnosis of Anomalies in Applications Driven by Electric Motors*. Sensors. https://doi.org/10.3390/s23249725

[4] *DC Brushless Motor (BLDC) Benefits | Cence Power*. (n.d.). https://www.cencepower.com/blog-posts/ac-electric-motors-vs-dc-brushless-motors#:~:text=BLDC%20motors%20also%20have%20a,of%20torque%20at%20higher%20speeds

[5] Lei, Y., Huang, Y., & Wang, H. (2021, April 21). Effects of Wind Disturbance on the Aerodynamic Performance of a Quadrotor MAV during Hovering. *Journal of Sensors, 2021*, 1–13. https://doi.org/10.1155/2021/6681716

[6] Lei, Y., & Wang, H. (2020). Aerodynamic Performance of a Quadrotor MAV Considering the Horizontal Wind. *IEEE Access*, *8*, 109421–109428. https://doi.org/10.1109/access.2020.3002706

[7] Lei, Y., & Lin, R. (2019, April 25). Effect of wind disturbance on the aerodynamic performance of coaxial rotors during hovering. *Measurement and Control*, *52*(5–6), 665–674. https://doi.org/10.1177/0020294019834961

[8] *Sensorless Field-Oriented Control for Open-End Winding Five-Phase Induction Motor With Parameters Estimation*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9399810

[9] *Comparison of vibration sources between symmetric and asymmetric HDD spindle motors with rotor eccentricity*. (2000). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/881067

[10] Ciurys, M. (2017, January 1). *Electromagnetic phenomena analysis in brushless DC motor with speed control using PWM method*. Open Physics. https://doi.org/10.1515/phys-2017-0109

[11] *STDES-PTOOL3A - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/evaluation-tools/stdes-ptool3a.html

[12] *STM32G4 Series - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/microcontrollers-microprocessors/stm32g4-series.html

[13] *X-CUBE-MCSDK - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/embedded-software/x-cube-mcsdk.html

[14] *X-CUBE-MCSDK - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/embedded-software/x-cube-mcsdk.html#documentation

[15] *Cloud vs. edge*. (n.d.). https://www.redhat.com/en/topics/cloud-computing/cloud-vs-edge

[16] Team, K. (n.d.). *Keras: Deep Learning for humans*. https://keras.io/

[17] *TensorFlow*. (n.d.). TensorFlow. https://www.tensorflow.org/

Chapter 2

Sensorless Field Oriented Control of PMSM /BLDC Motors

2.1 Introduction

In recent years, the domain of controlled electrical drives has experienced significant growth, largely attributed to the advancements in semiconductor technology, particularly in power and signal electronics, along with the evolution of micro-electronic microprocessors and digital signal processors (DSPs) [1]. These technological strides have facilitated the development of highly efficient AC drive control systems, characterized by reduced power dissipation hardware and increasingly precise control architectures [2]. Notably, the refinement of electrical drive controls extends beyond mere regulation of DC current and voltage to encompass the management of three-phase currents and voltages through vector control techniques. This chapter delineates one of the most efficient vector control schemes known as Field Oriented Control (FOC) [3-4]. FOC hinges upon three pivotal elements: the manipulation of machine current and voltage space vectors, the transformation of a three-phase speed and time-dependent system into a two-coordinate time-invariant system, and the generation of effective Pulse Width Modulation (PWM) patterns [5]. These components collectively empower the AC machine control, affording it the advantages previously associated with DC machine control while mitigating the limitations of mechanical commutation. Moreover, the FOC framework, characterized by precise steady-state and transient control, engenders high dynamic performance in terms of response times and power conversion efficiency.

2.2 Permanent Magnet Synchronous Motors / Brushless DC Motors Control Fundamentals

Traditional AC motor control configurations typically employ three sinusoidal voltages with a spatial displacement of 120 degrees applied to the three stator phases. In classic AC drive systems, the generation of these three sine waves relies on the electromechanical characteristics of the motor and an equivalent model representing the motor in its steady state [6]. Consequently, the control architecture resembles three distinct single-phase system controls rather than a unified control approach for the three-phase system.

The machine models and characteristics utilized are primarily applicable in steady-state conditions. However, this approach tends to permit high peak voltage and current transients, which not only impair the dynamic performance of the drive but also compromise power conversion efficiency [7]. Moreover, to withstand these transient electrical spikes, power components often need to be oversized.

Significant challenges arise in regulating variables with sinusoidal references. PI regulators struggle to achieve sinusoidal regulation without distorting the reference waveform, while hysteresis controllers introduce high-bandwidth noise into the system, making it challenging to filter out effectively. Furthermore, there is typically a lack of mechanisms for managing imbalance in the three-phase system and considering phase interactions.

Finally, the control structure necessitates customization based on the type of motor being employed, whether asynchronous or synchronous, further complicating the control scheme design.

2.3 Field Oriented Control Mechanism

Field Oriented Control (FOC) involves the regulation of stator currents represented as vectors. This control strategy relies on projections that transform a three-phase system, which is dependent on time and speed, into a two-coordinate system (d and q coordinates) that is

time-invariant. Through these projections, the control structure resembles that of a DC machine control. FOC-controlled machines require two constant input references: the torque component (aligned with the q coordinate) and the flux component (aligned with the d coordinate).

Since Field Oriented Control is primarily based on projections, the control structure deals with instantaneous electrical quantities. This characteristic ensures accuracy in all operating conditions, including steady-state and transient states, independent of the bandwidth limitations of mathematical models.

FOC addresses the challenges of traditional control schemes in several ways:
- Facilitating the attainment of constant references for the torque and flux components of the stator current.|
- Simplifying the implementation of direct torque control, as the torque expression in the (d,q) reference frame is explained as:

$$m \propto \psi_R i_{Sq} \tag{1}$$

By maintaining the amplitude of the rotor flux ψ_R at a fixed value we have a linear relationship between torque and torque component $\mathbf{i_{Sq}}$. We can then control the torque by controlling the torque component of stator current vector.

2.4 Space Vector Transformations

The analysis of three-phase voltages, currents, and fluxes in AC motors can be effectively conducted using complex space vectors. Concerning currents, the space vector can be defined as follows: if i_a, i_b, and i_c represent the instantaneous currents in the stator phases, then the complex stator current vector i_S is defined as follows:

$$\bar{i}_S = i_a + \alpha i_b + \alpha^2 i_c \tag{2}$$

Here, $\alpha = e^{j\frac{2}{3}\pi}$ and $\alpha^2 = e^{j\frac{4}{3}\pi}$ represent the spatial operators. The diagram below illustrates the stator current complex space vector:

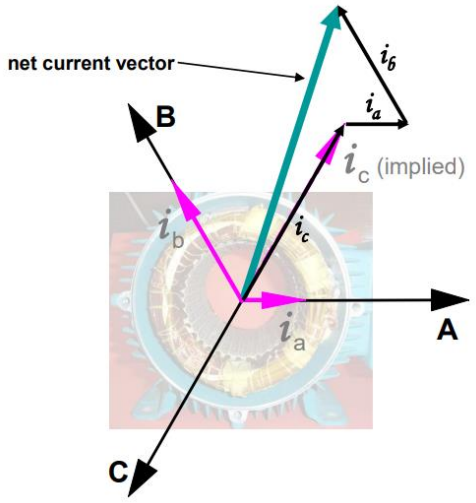


Fig 2.1 Stator current space vector and its component in (A, B, C) [8]

In this representation, (A, B, C) denote the axes of the three-phase system. This current space vector illustrates the three-phase sinusoidal system. However, to facilitate analysis, it must be transformed into a two-coordinate time-invariant system. This transformation comprises two sequential steps:

- Transformation from (a, b, c) to (α, β) (the Clarke transformation), yielding a two-coordinate time-variant system.
- Transformation from (α, β) to (d, q) (the Park transformation), resulting in a two-coordinate time-invariant system.

2.4.1 Clarke Transformation (a, b, c → α, β)

The space vector can be transformed into another reference frame with only two orthogonal axes called (α, β). Assuming that the axis a and the axis α are in the same direction, we have the following vector diagram:

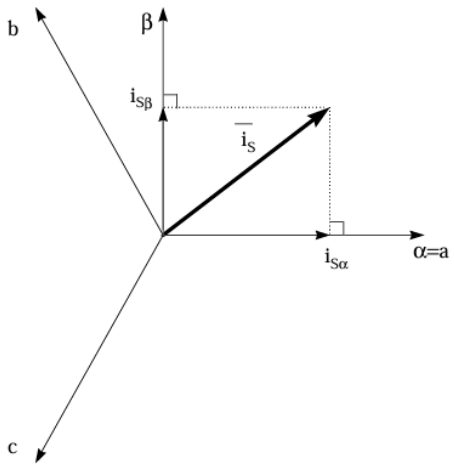


Fig 2.2 Stator current space vector and its components in (α, β)

$$\begin{cases} i_{S\alpha} = i_a \\ i_{S\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \end{cases} \quad (3)$$

The projection that converts the three-phase system into the (α, β) two-dimensional orthogonal system is depicted below. We obtain a two-coordinate system that still depends on time and speed :

$$\begin{pmatrix} i_{S\alpha} \\ i_{S\beta} \end{pmatrix}$$

2.4.2 Park Transformation (α, β → d, q)

This transformation is pivotal in Field Oriented Control (FOC). It modifies a two-phase orthogonal system (α, β) into the d,q rotating reference frame. If we align the d axis with the

rotor flux, the following diagram illustrates the relationship between the current vector in the two reference frames:

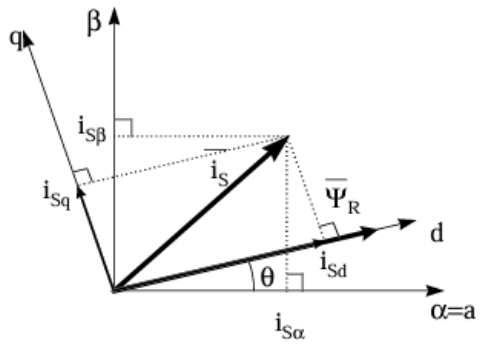


Fig 2.3 Stator current space vector and its component in (α, β) and in the d,q rotating reference frame

Where, theta represents the rotor flux position. The flux and torque components of the current vector are determined by the following equations:

$$\begin{cases} i_{Sd} = i_{S\alpha}\cos\theta + i_{S\beta}\sin\theta \\ i_{Sq} = -i_{S\alpha}\sin\theta + i_{S\beta}\cos\theta \end{cases} \quad (4)$$

These components rely on the current vector (α, β) components and the rotor flux position. If the correct rotor flux position is known, then through this projection, the d,q component becomes a constant. We obtain a two-coordinate system with the following characteristics:

- Two-coordinate time-invariant system
- With, i_{Sd} (flux component) and i_{Sq} (torque component), direct torque control is possible and straightforward.

2.4.3 Inverse Park Transformation (d, q → α, β)

The transformation from the (d, q) rotating reference frame to the (α, β) two-phase orthogonal system, often termed as the Inverse Park Transformation, involves an equation that alters the voltages:

$$\begin{cases} V_{S\alpha ref} = V_{Sdref}\cos\theta - V_{Sqref}\sin\theta \\ V_{S\beta ref} = V_{Sdref}\sin\theta + V_{Sqref}\cos\theta \end{cases} \quad (5)$$

This equation calculates the components of the reference vector denoted as V_r, where it represents the voltage space vector intended for application to the motor phases.

2.4.4 Inverse Clarke Transformation (α, β → a, b, c)

The Inverse Clarke Transform block performs the reversal of Clarke transformation on balanced, two-phase orthogonal elements in the stationary αβ reference frame, generating balanced, three-phase elements in the stationary abc reference frame. Alternatively, it can reverse Clarke transformation on elements α, β, and 0 to produce three-phase elements a, b, and c. In a balanced system, the zero component is consistently zero. Adjust the Number of inputs parameter to utilize either two or three inputs.

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} \quad (6)$$

2.5 Field Oriented Control Operation

The following diagrams summarize the functionality of Field Oriented Control mechanism.

Two motor phase currents are measured and fed into the Clarke transformation module. The outputs of this projection are designated as "i_{Sa}" and "i_{Sb}" These two components of the current serve as inputs for the Park transformation, which provides the current in the d,q rotating reference frame. The "i_{Sd}" and "i_{Sq}" components are then compared to the flux reference (the flux reference) and torque reference (the torque reference).

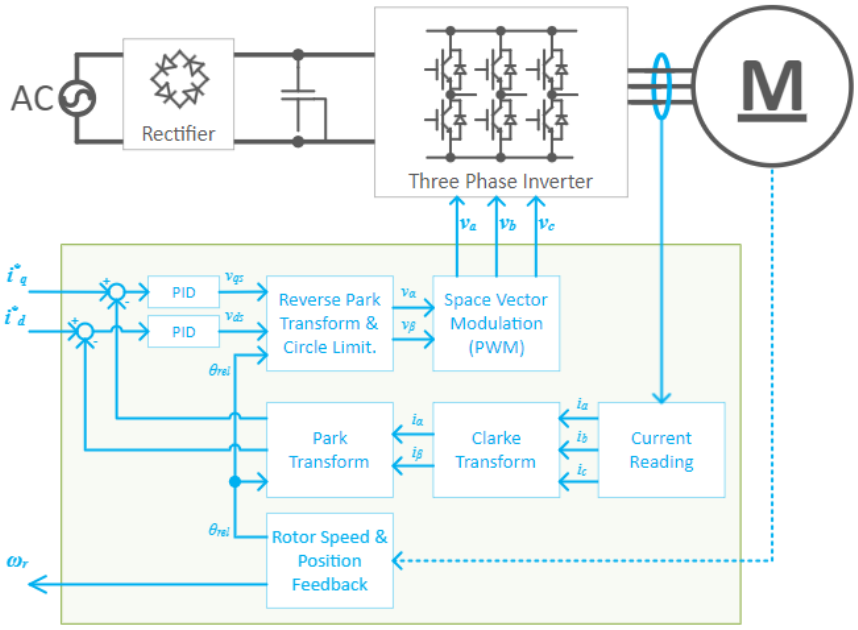


Fig 2.4 Field Orinted Control Block Diagram STM32 MCSDK [9]

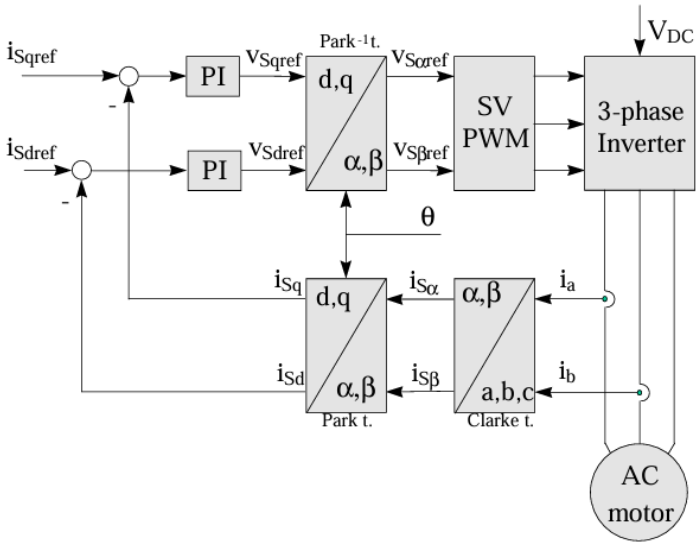


Fig 2.5 Field Orinted Control Block Diagram

Ref: [8]

Two motor phase currents are measured and fed into the Clarke transformation module. The outputs of this projection are designated as "i_{Sa}" and "i_{Sb}" These two components of the current serve as inputs for the Park transformation, which provides the current in the d,q rotating reference frame. The "i_{Sd}" and "i_{Sq}" components are then compared to the flux reference (the flux reference) and torque reference (the torque reference).

At this juncture, the "flux reference" control structure exhibits a notable advantage: it can effectively control both synchronous and induction machines by merely adjusting the flux reference and determining the rotor flux position. In synchronous permanent magnet motors, where the rotor flux is fixed (determined by the magnets), no creation is necessary. Therefore, when controlling such motors, the flux reference should be set to zero. Conversely, for induction motors, which require rotor flux creation to operate, the flux reference must not be zero. This addresses a significant limitation of traditional control structures: the lack of portability from asynchronous to synchronous drives.

The torque command "torque reference" could be derived as the output of the speed regulator when employing speed Field Oriented Control (FOC). The outputs of the current regulators are "v_{Sdref}" and "v_{Sqref}" which are then applied to the inverse Park transformation. The outputs of this transformation are "$v_{S\alpha ref}$" and "$v_{S\beta ref}$" representing the components of the stator vector voltage in the a,b stationary orthogonal reference frame. These values serve as inputs for the Space Vector Pulse Width Modulation (PWM). The signals produced by this block drive the inverter.

It's worth noting that both the Park and inverse Park transformations require knowledge of the rotor flux position. Determining this position depends on the type of AC machine being used (synchronous or asynchronous), which is further discussed in subsequent paragraphs.

The input requirements for Field Oriented Control (FOC) encompass knowledge of two-phase currents (given that the motor is star-connected, the third phase current is also known, since ia + ib + ic = 0), and the rotor flux position.

Current Sampling: Phase currents ia and ib are sampled and converted by an A/D converter. The accuracy of FOC hinges on the precise measurement of these currents.

Rotor Flux Position: Understanding the rotor flux position is paramount for FOC. Any error in this variable results in misalignment of the rotor flux with the d-axis, leading to incorrect flux and torque components of the stator current (i_{Sd} and i_{Sq}). The following diagram illustrates the (a,b,c), (α, β), and (d,q) reference frames, along with the correct position of the rotor flux, the stator current, and the stator voltage space vector rotating with d,q reference at synchronous speed.

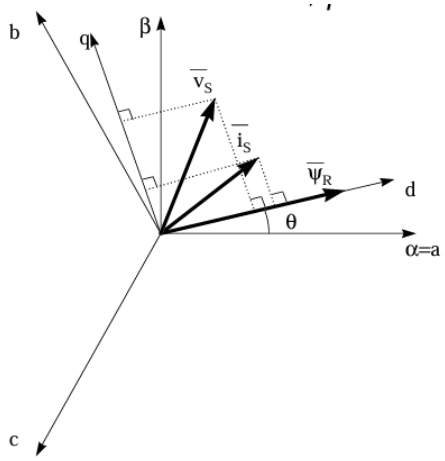


Fig 2.6 Current, voltage, and rotor flux space vectors in the d,q rotating reference frame and their relationship with a,b,c and α, β stationary reference frame

The measurement of the rotor flux position differs depending on whether it is a synchronous or induction motor.

In synchronous machines, the rotor speed equals the rotor flux speed. Therefore, theta (rotor flux position) is directly measured by a position sensor or by integrating the rotor speed. In induction machines, the rotor speed is not equal to the rotor flux speed (there is a slip speed), necessitating a specific method to calculate theta.

2.6 Conclusion

In conclusion, Chapter 2 provides a comprehensive exploration of Sensorless Field Oriented Control (FOC) of Permanent Magnet Synchronous Motors (PMSM) and Brushless DC (BLDC) Motors. It begins by elucidating the backdrop of controlled electrical drives, underscoring the pivotal role of semiconductor advancements and digital signal processing in fostering the evolution of efficient AC drive control systems. The chapter then delves into the fundamentals of traditional AC motor control configurations, pinpointing the limitations associated with sinusoidal control approaches and the challenges in managing transient electrical spikes.

Central to the chapter is the exposition of Field Oriented Control (FOC) mechanisms, which revolutionize AC motor control paradigms by regulating stator currents as vectors in a two-coordinate time-invariant system. Through meticulous projections and transformations, FOC enables precise control over torque and flux components, facilitating direct torque control and mitigating the shortcomings of traditional control schemes. The elucidation of space vector transformations, including the Clarke and Park transformations, underscores the theoretical underpinnings of FOC and its practical implications in motor control applications.

Furthermore, the chapter delineates the operational intricacies of FOC, elucidating its block diagram and elucidating the role of current sampling and rotor flux position determination in ensuring accurate control. By addressing the distinctions between synchronous and induction motors and providing insights into rotor flux position measurement techniques, the chapter equips readers with the requisite knowledge to implement FOC effectively across diverse motor types. In essence, Chapter 2 serves as a foundational resource for understanding the principles and applications of Sensorless Field Oriented Control in the realm of PMSM and BLDC motor control. Its elucidation of theoretical concepts, coupled with practical insights into control scheme design and implementation considerations, positions readers to navigate the complexities of modern motor control systems with proficiency.

2.7 References

[1] *The evolution of DSP processors*. (2000, March 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/826411

[2] Xu, D., Wang, B., Zhang, G., Wang, G., & Yu, Y. (2018, March 1). *A review of sensorless control methods for AC motor drives*. CES Transactions on Electrical Machines and Systems. https://doi.org/10.23919/tems.2018.8326456

[3] *Implementation of field oriented control for permanent magnet synchronous motor*. (2014, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/7045278

[4] *Field Oriented Control of Permanent Magnet Synchronous Motor (PMSM) Driven Electric Vehicle And Its Performance Analysis*. (2021, September 24). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9573814

[5] *Model-Based Sensored Field Oriented Control Implementation for Permanent Magnet Synchronous Motor*. (2021, September 20). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9588787

[6] *Permanent Magnet Synchronous Motor Controllers | Microchip Technology*. (n.d.). https://www.microchip.com/en-us/solutions/technologies/motor-control-and-drive/motor-types/permanent-magnet-synchronous-motors#:~:text=Permanent%20Magnet%20Synchronous%20Motors%20(PMSMs,AC%20Induction%20Motors%20(AICMs)

[7] Krishnan, R. (2017, December 19). *Permanent Magnet Synchronous and Brushless DC Motor Drives*. CRC Press eBooks. https://doi.org/10.1201/9781420014235

[8] *Motor drives design resources | TI.com*. (n.d.). https://www.ti.com/applications/industrial/motor-drives/overview.html

[9] *X-CUBE-MCSDK - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/embedded-software/x-cube-mcsdk.html

Chapter 3

Convolution Neural Network for Pattern Recognition

3.1 Introduction

In the vast landscape of artificial intelligence, Convolutional Neural Networks (CNNs) stand out as powerful tools for pattern recognition. Imagine you're looking at a photo of your favorite pet. How does your brain effortlessly recognize their furry face? Similarly, CNNs emulate this process, but in the realm of computers. At its core, a CNN mimics the human brain's ability to identify patterns in visual data [1]. Just as you can recognize a friend's face from different angles or lighting conditions, CNNs can learn to detect complex patterns in images, distinguishing between objects, shapes, and textures.

But why are CNNs so effective? It all boils down to their architecture. Inspired by the organization of the visual cortex, CNNs consist of layers of interconnected neurons, each processing specific features of an image. Through a process of convolution and pooling, CNNs extract meaningful information while reducing computational complexity [2].

In this chapter, we delve into the inner workings of CNNs, exploring how they learn from data to recognize patterns. From understanding the basics of convolution to unraveling the mysteries of pooling layers, we'll guide you through the fundamental concepts behind CNNs.

But CNNs aren't just limited to image recognition. They find applications in various fields, from medical diagnosis to autonomous driving [3]. By the end of this chapter, we will have an understanding of the fundamentals of CNNs and also appreciate their wide-ranging impact on pattern recognition tasks.

3.2 Origins of CNN

3.2.1 Biological Inspiration of CNN

The evolution of convolutional neural networks (CNNs) is deeply rooted in both neuroscience and artificial intelligence, representing a convergence of brain-inspired concepts with advancements in computer science and engineering.

In the mid-twentieth century, Dr. Hubel and Dr. Wiesel conducted pioneering research in the field of Sensory Processing, particularly focusing on the visual system.



Fig. 3.1 Neurophysiologists David Hubel (left) and Torsten Wiesel (right) [4].

Hubel and Wiesel made significant discoveries about the primary visual cortex (V1) in cats [5]. Their experiments involved inserting micro-electrodes into the primary visual cortex of partially anesthetized cats, immobilizing them to observe neural responses. The cats were presented with images of lines oriented at various angles.

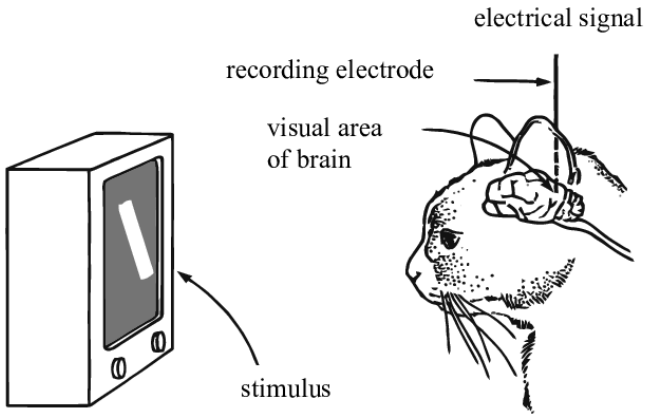


Fig. 3.2An partially anesthetized cat is equipped with contact lenses to ensure the eyes remain focused on a screen. Images are then projected onto the screen, and neuron responses are recorded using an extracellular electrode [6]

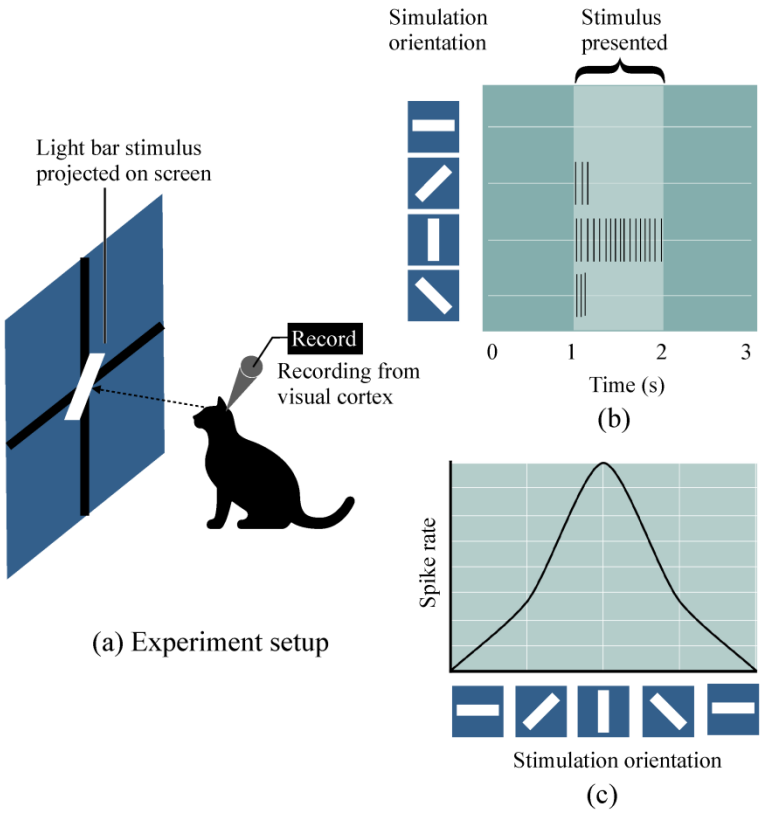


Fig. 3.3 illustrates the selective response of a neuron in the primary visual cortex to line segments. (a) An anesthetized cat is equipped with contact lenses to ensure precise eye

fixation on a screen. Images are then projected onto the screen, and neuron responses are recorded using an extracellular electrode. (b) The neuron typically exhibits robust responses to a bar of light oriented at a specific angle, with minimal or no response to other orientations. (c) The spike rate of the neuron varies as the orientation of the stimulation changes, as depicted in the curve [7]

Their findings revealed that certain neurons exhibited rapid firing rates when exposed to lines at specific angles, while others responded optimally to lines oriented differently. Additionally, some neurons displayed differential responses to light and dark patterns, while others were specialized in detecting motion in particular directions.

They identified two major cell types: simple cells, which respond to bars of light or dark at specific spatial locations with a preferred orientation, and complex cells, which exhibit less strict response profiles but still have preferred orientations. Hubel and Wiesel proposed that complex cells likely receive input from multiple simple cells with similar preferred orientations but different preferred locations.

In 1980, Fukushima translated Hubel and Wiesel's findings into a functional model of the visual system called the Neocognitron [8]. This model, a precursor to modern CNNs, featured two main cell types: S-cells, akin to simple cells, which applied a 2-D grid of weights at each location in the input image to generate responses, and C-cells, resembling complex cells, whose responses were nonlinear functions of S-cells from the same plane but at different locations.

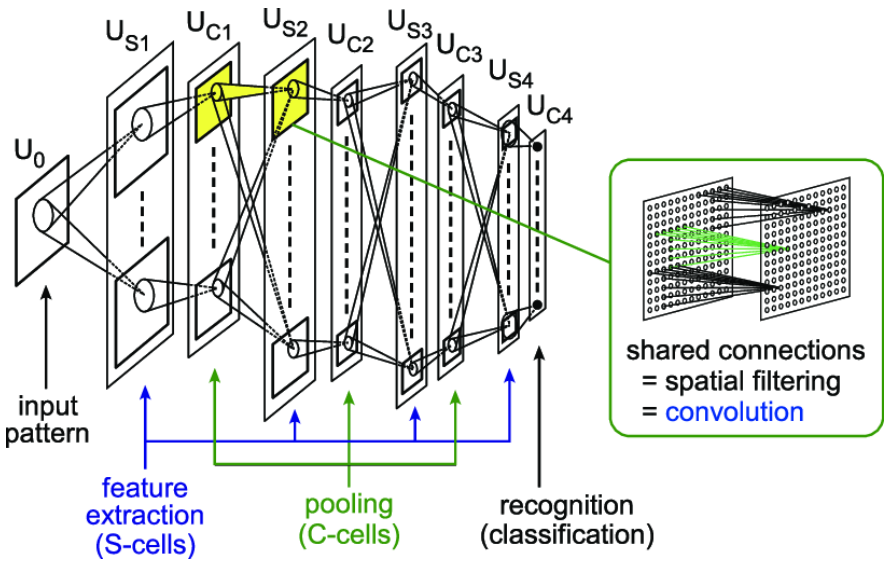


Fig. 3.4 Architecture of Neocognitron [9]

The Neocognitron replicated the basic computations of V1 by alternating layers of simple and complex cells. This hierarchical structure mirrored not only the operations of V1 but also the entire ventral visual pathway. The network exhibited "self-organization," with weights adjusting over time through exposure to unlabeled images.

3.2.2 CNN for Computer Vision

Convolutional Neural Networks (CNNs) represent a specialized category of artificial neural networks, drawing inspiration from the architecture and fundamental processes of biological vision. Unlike non-convolutional fully connected networks, where each neuron in a layer connects to all neurons in the preceding layer, CNNs adopt local connectivity and shared weights across convolutional layers [10].

Typically, a CNN comprises numerous processing units resembling neurons, arranged in interconnected layers often likened to brain regions, with connections governed by weights that emulate synaptic integration and activation properties [11] .

The output of each stage undergoes non-linear combination before being forwarded to subsequent layers, forming a hierarchical structure. This iterative process continues until the cascade concludes with discriminative classification or regression provided by the final readout layers. Within a convolutional layer, multiple filters with distinct receptive fields are applied to the input image via convolution operations, enabling the network to capture spatial and temporal patterns in the input data[12]. The weights of these filters serve as learnable parameters, with the learning process typically facilitated by standard gradient descent algorithms. The initial breakthrough demonstrating the efficacy of Convolutional Neural Networks (CNNs) occurred in 1989, showcasing the capability of a small CNN trained with supervised learning via the backpropagation algorithm to classify handwritten digits [13].

However, it wasn't until 2012 that CNNs gained widespread attention with the emergence of an 8-layer network, known as 'AlexNet'[14] or 'Standard Architecture', which significantly surpassed previous state-of-the-art performance in the ImageNet challenge [15]. The ImageNet dataset comprises over a million real-world images, and the challenge involves classifying images into one of a thousand object categories.

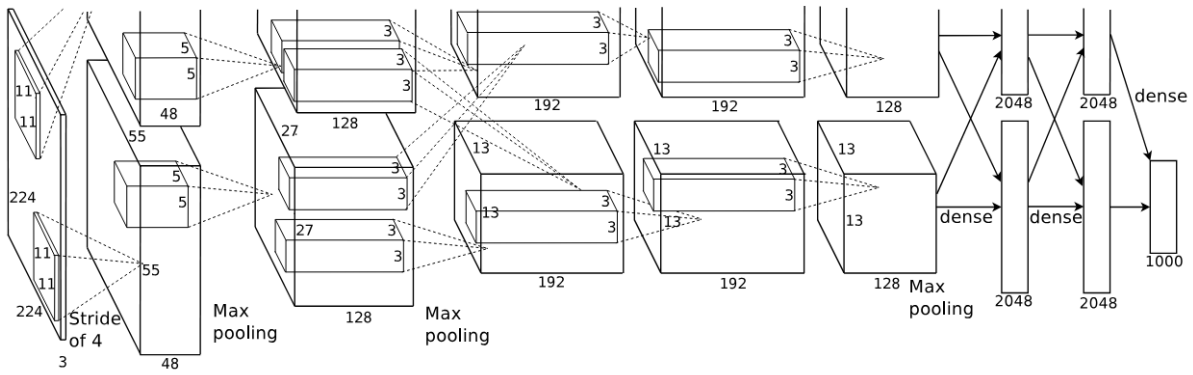


Fig. 3.5 AlexNet architecture [15]

This breakthrough underscored that the foundational features of the visual system, as identified by neuroscientists, could indeed facilitate vision given appropriate learning algorithms and data.

Since this milestone, numerous CNN architectures have been explored, with variations in parameters such as network depth, pooling layer placement, number of feature maps per layer, training procedures, and the presence of residual connections that bypass layers [16]. The primary objective within the computer vision community has been to develop models that outperform standard image benchmarks, with secondary aims including reducing network size and training with less data. Alignment with biological principles is not the primary driving force behind these explorations.

3.3 CNN for Pattern Recognition – Computer Vision Inspired Signal Pattern Recognition (for Classification and Anomaly Detection tasks)

The field of computer vision, which focuses on the automatic analysis and interpretation of image data, stands as a prominent application area for machine learning [17]. Traditionally, computer vision heavily relied on 3-dimensional projective geometry, with hand-crafted features serving as input to simple learning algorithms [18]. However, it was one of the earliest domains to undergo transformation through the deep learning revolution, largely propelled by the emergence of Convolutional Neural Network (CNN) architecture. Initially developed for image analysis, CNNs have since found application in diverse domains, including the analysis of sequential data. Recently, alternative architectures based on transformers [19] have emerged as contenders, demonstrating competitiveness with convolutional networks in certain applications competitiveness with convolutional networks in certain applications.

3.3.1 Image Data as Input to the CNN

An image is represented as a rectangular array of pixels, where each pixel typically holds either a grey-scale intensity or, more commonly, a triplet of red, green, and blue channels, each with its own intensity value. These intensity values are non-negative numbers, often

constrained by the limits of the capturing hardware. While we usually treat intensities as continuous variables, they are practically represented with finite precision, often as 8-bit numbers ranging from 0 to 255 [20].

Certain types of images, such as magnetic resonance imaging (MRI) scans used in medical diagnosis, consist of three-dimensional grids of voxels [21]. Similarly, videos are sequences of two-dimensional images, thus can be conceived as three-dimensional structures where successive frames are stacked through time.

Now, let's consider the challenge of applying neural networks to image data for various applications. Images typically exhibit high dimensionality, with modern cameras capturing images containing tens of megapixels. Treating such image data as unstructured would necessitate a model with an impractically large number of parameters. Moreover, disregarding the structured nature of image data, where pixel positions are crucial, would lead to suboptimal performance. This structured nature is evident when we permute the pixels of an image randomly, resulting in an image that does not resemble a natural scene. Similarly, generating a synthetic image by randomly assigning pixel intensities independently for each pixel rarely produces a natural-looking image. Local correlations are integral to image data, with nearby pixels in natural images more likely to have similar colors and intensities compared to distant pixels. Leveraging this prior knowledge allows encoding strong inductive biases into neural networks, leading to models with fewer parameters and enhanced generalization accuracy [22].

3.3.2 Real-time Signal Waveforms Features Stacked to form an Image-like Array of Pixels

For CNN-based pattern recognition applications involving signals, real-time acquired signals can be sampled and arranged to form an array of pixels resembling an image. Let's consider multiple arbitrary signal waveforms from a generated dataset, as depicted in Figure 3.5.

These waveforms represent 5 different signals observed over a duration of 50 seconds (x-axis).

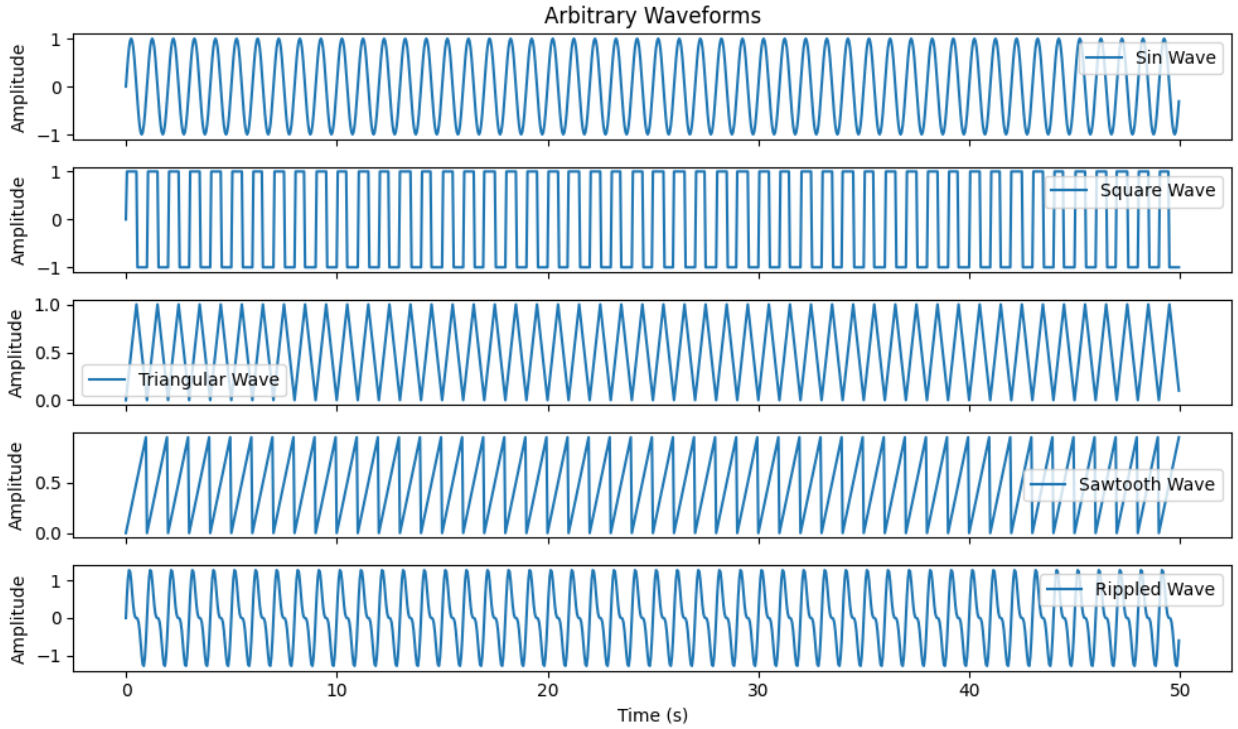


Fig. 3.6 Arbitrary waveforms of Sin, Square, Triangular, Sawtooth and Rippled Waveforms

To transform our dataset into image-like samples suitable for CNN input, we construct a 5x5 matrix by selecting 5 samples (seconds) for each signal, as illustrated in Figure X. Consequently, our dataset comprises 10 sets of 5x5 matrices or images.

Given that CNN algorithms excel at visual tasks, we demonstrate the concept of representing signals as image-like inputs for CNNs, as depicted in Figure X, where the heatmap representation gives a mental image for signals represented as images. In subsequent chapters, we will explore the application of this concept to create image-like signal matrices involving stacked electric motor signals.

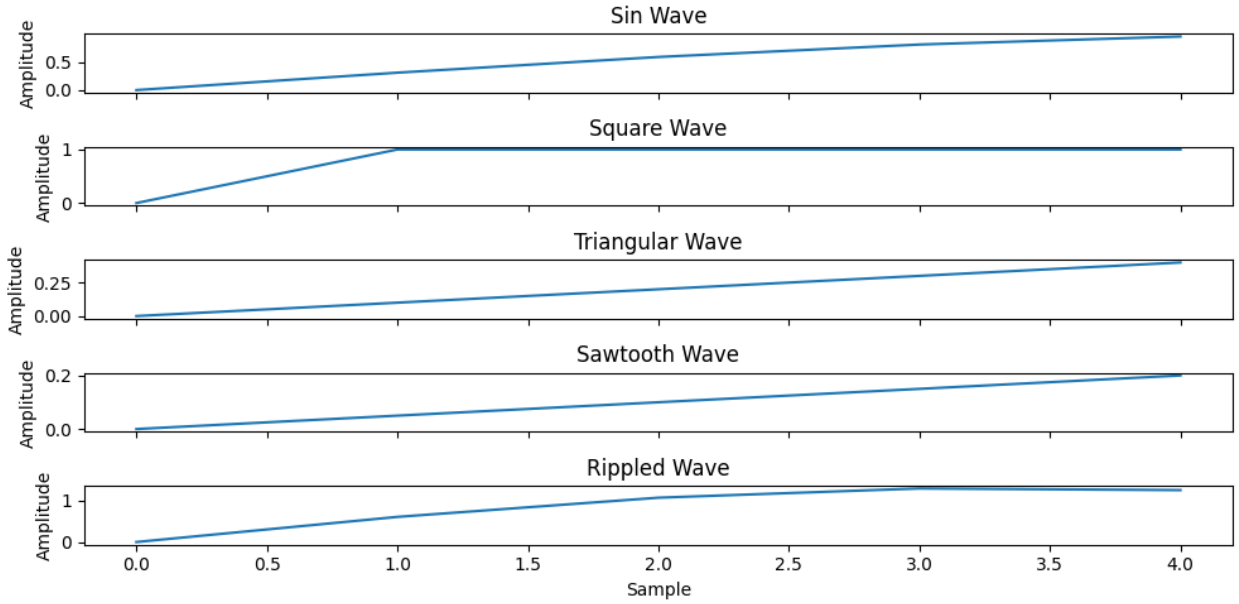


Fig. 3.7 Five samples considered for 5x5 dimension Input Image

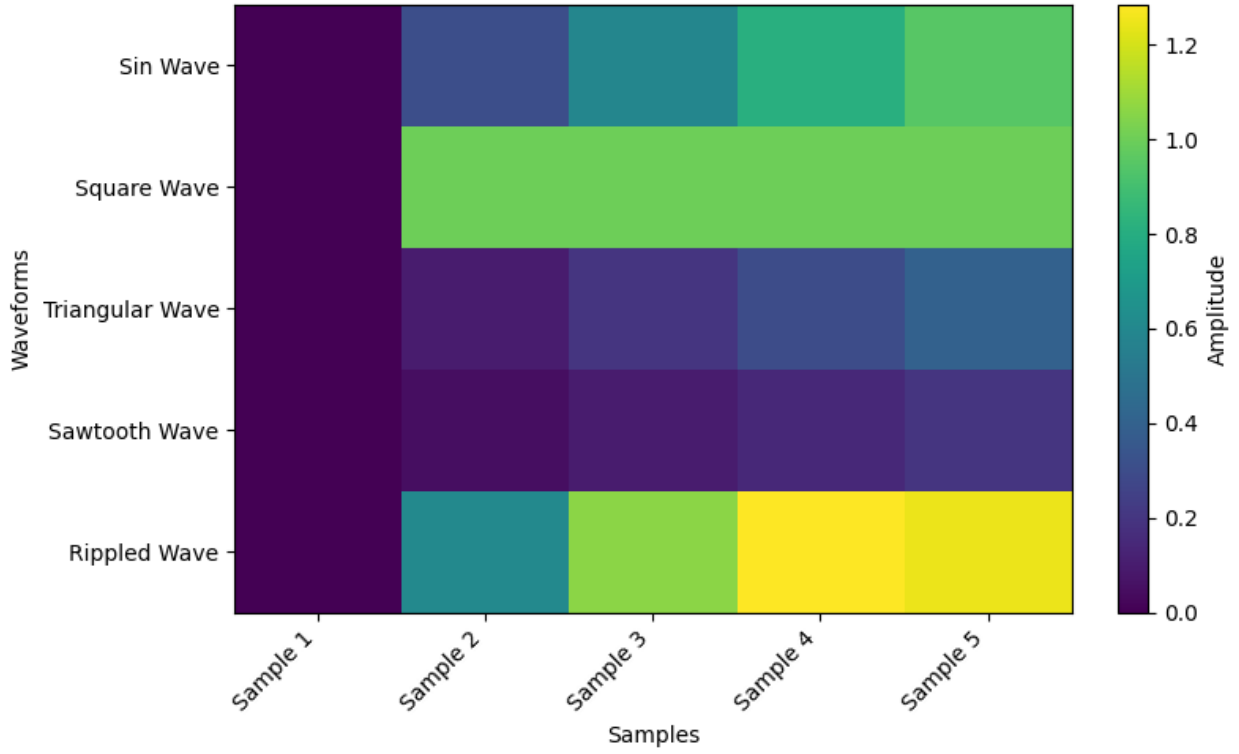


Fig. 3.8 Heatmap representation to create a mental image for Input image

3.4 Convolution Filters and Operations

One rationale behind the adoption of convolutional networks is rooted in the inherent characteristics of image data, the primary domain for which CNNs were devised. A conventional fully connected architecture applied to image data would necessitate an exceedingly large number of parameters due to the high-dimensional nature of images.

By crafting an architecture that integrates our prior knowledge regarding the structure of images, we can significantly reduce the data requirements and enhance generalization concerning symmetries within the image space.

To harness the two-dimensional structure of image data and instill inductive biases, we can leverage four interrelated concepts: hierarchy, locality, equivariance, and invariance[23]. For instance, in the task of facial detection, a natural hierarchical structure emerges as an image may encompass multiple faces, each face comprising elements such as eyes, and each eye exhibiting structures like an iris, which in turn consists of edges. At the lowermost tier of the hierarchy, a node in a neural network could discern the presence of a feature, such as an edge, utilizing local information from a small image region, thereby requiring access to only a subset of the image pixels. More intricate structures higher up the hierarchy can be identified by amalgamating multiple features detected at preceding levels. However, it is imperative that while we embed the notion of hierarchy into the model, the specifics of the hierarchy, including the types of features extracted at each level, should be learned from the data rather than hardcoded[24].

Hierarchical models align seamlessly with the deep learning paradigm, facilitating the extraction of intricate concepts from raw data through a succession of possibly numerous "layers" of processing, wherein the entire system undergoes end-to-end training [25].

3.4.1 Convolution Kernels

Convolution kernels, also referred to as filters, are small 2D matrices utilized for the convolution operation. These kernels traverse the input data, executing element-wise multiplication with the corresponding pixels. Subsequently, they produce a feature map that accentuates specific patterns or features within the input. The dimensions of a kernel are intricately linked to the input shape of the image and the architecture of the entire network. Typically, kernels are square matrices with dimensions denoted as (MxM). During convolution, the movement of a kernel occurs systematically, from left to right and top to bottom, across the input data[26].



Fig. 3.9 Kernel Operation to Input Images [27]

3.4.2. Kernel Vs Filters

Generally, the terms "kernel" and "filter" are often used interchangeably. However, in the context of convolutional neural networks (CNNs), they possess distinct meanings and roles.

A kernel, as previously described, refers to a matrix of weights utilized in the convolution operation to extract information from the input.

On the other hand, a filter is a collection of multiple kernels, with each kernel assigned to a specific channel of the input data. Filters extend one dimension beyond kernels. For instance, in 2D convolutions, filters are represented as 3D matrices, constituting a concatenation of 2D kernels. Therefore, for a CNN layer with kernel dimensions h * w and input channels k, the filter dimensions would be k * h * w [28].

3.4.3 Filter Application Example

The provided illustration showcases a kernel's versatility in image processing tasks, notably Gaussian blur, sharpening, and edge detection. Discussion of functionality will be done in further sections.

Original	*Gaussian Blur*	*Sharpen*	*Edge Detection*
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

Fig. 3.10 Different Filters applied to an original image and their results [29]

Gaussian blur employs a specific kernel configuration to smooth images, reducing noise and enhancing overall visual coherence. This preprocessing step is often utilized to prepare images for subsequent analysis or processing stages.

Conversely, sharpening employs a distinct kernel design aimed at accentuating edge contrast, thereby enhancing image clarity and definition. By selectively enhancing edge features, this operation contributes to the perceptual sharpness of the image.

Additionally, kernels tailored for edge detection play a crucial role in identifying abrupt changes in pixel intensity, effectively delineating object boundaries within an image. This operation facilitates essential tasks such as object recognition and segmentation, thereby enhancing the robustness and efficacy of image processing algorithms.

Overall, the demonstrated kernel configurations exemplify their pivotal role in various image enhancement and analysis tasks, underscoring their utility across diverse applications in image processing and computer vision.

3.4.4 Feature Extraction

For simplicity, let's initially focus on grayscale images, which have a single channel. Consider a single unit in the first layer of a neural network, which takes as input the pixel values from a small rectangular region, or patch, of the image. This patch, termed the receptive field of the unit, embodies the concept of locality. We aim for the weight values associated with this unit to learn to detect useful low-level features[30]. The output of this unit follows the standard functional form, comprising a weighted linear combination of the input values, and subsequently transformed using a nonlinear activation function:

$$z = \mathrm{ReLU}(w^T x + w_0)$$

Here, x denotes a vector representing pixel values for the receptive field, and we assume a ReLU activation function. Each input pixel corresponds to a weight, forming a small two-dimensional grid known as a filter, also referred to as a kernel, which can be visualized as an image [31].

Let's assume that w and w_0 are fixed. We aim to determine the value of the input image patch x that maximizes the response of this hidden unit. To accomplish this, we need to constrain x in some manner. Let's suppose that the Euclidean norm of x is fixed ($||x||^2$). Then, the solution for x that maximizes $w^T x$, and consequently maximizes the response of the hidden unit, takes the form $x = \alpha w$ for some coefficient α.

This indicates that the maximum output response from this hidden unit occurs when it detects an image patch that, up to an overall scaling, resembles the kernel image. Notably, the ReLU activation function generates a nonzero output only when $w^T x$ exceeds a threshold of $-w_0$. Therefore, the unit functions as a feature detector, signaling when it identifies a sufficiently good match to its kernel.

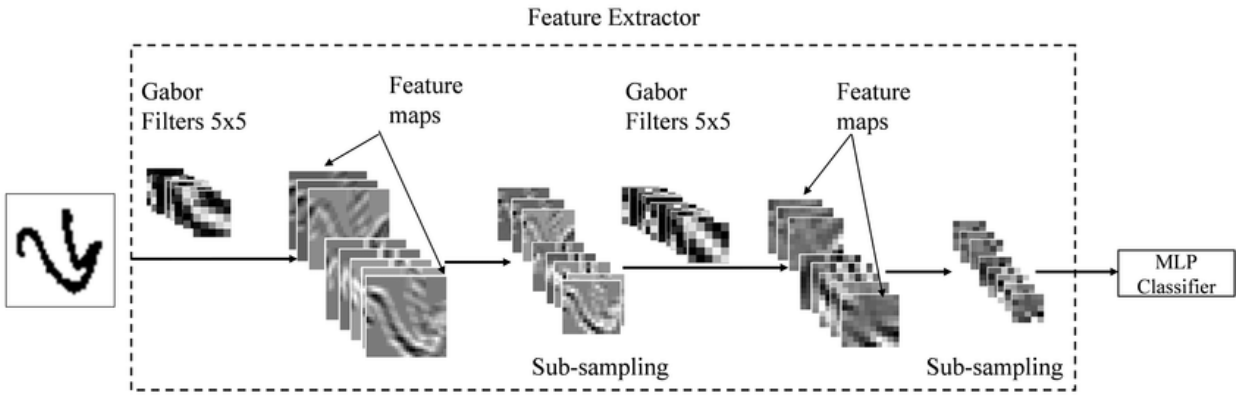


Fig. 3.11 Visualization of Feature Extraction [32]

3.4.5 Padding Operation

Based on the discussion so far, it's evident that the convolution map is typically smaller than the original image. If the image has dimensions J × K pixels and is convolved with a kernel of dimensions M × M (filters are typically square), the resulting feature map has dimensions: $(J - M + 1) \times (K - M + 1)$

However, in some cases, we may desire the feature map to have the same dimensions as the original image. This can be achieved by padding the original image with additional pixels around the outside. If we pad with P pixels, then the output map has dimensions $(J + 2P - M + 1) \times (K + 2P - M + 1)$.

When there is no padding, such that P = 0, this is referred to as a valid convolution.

When choosing the value of P so that the output array matches the input size, with $P = (M - 1)/2$, it's termed a same convolution because the image and the feature map possess

identical dimensions. In computer vision, filters typically utilize odd values of M to facilitate symmetric padding on all sides of the image and ensure a well-defined central pixel associated with the filter's location.

Lastly, selecting appropriate values for the intensities associated with the padding pixels is crucial. A common approach is to set the padding values to zero, following the subtraction of the mean from each image to ensure zero represents the average value of the pixel intensity. Additionally, padding can also be applied to feature maps for processing by subsequent convolutional layers[33].



Fig. 3.12 Padding in a Kernel [34]

3.4.6 Striding Operation

In typical image processing applications, where images often contain a large number of pixels, and the kernels are relatively small (M << J, K), (where (J, K) and M are image and the kernel dimension respectively) the resulting convolutional feature map tends to be of a similar size to the original image. Same padding ensures that the feature map retains the same dimensions as the input image.

However, there are scenarios where it's desirable to use feature maps significantly smaller than the original image. This approach offers flexibility in designing convolutional network architectures. One method to achieve this is through the use of strided convolutions. Instead

of moving the filter over the image one pixel at a time, it is advanced in larger steps of size S, known as the stride [35].

When employing the same stride horizontally and vertically, the number of elements in the feature map can be calculated using the formula:

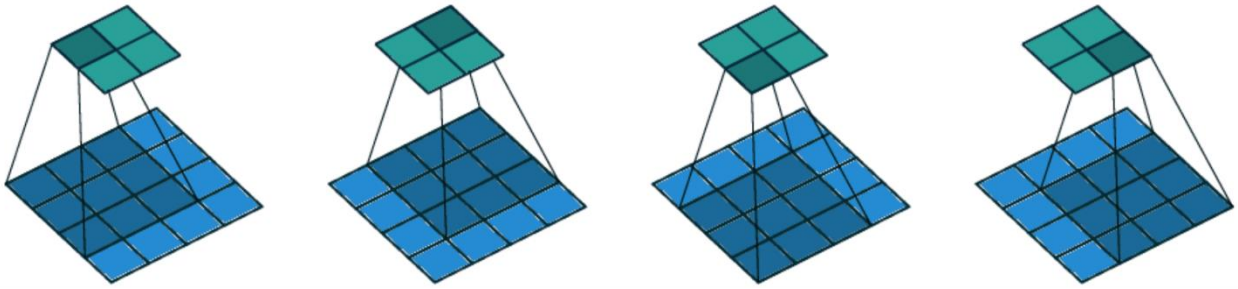


Fig. 3.13 Example of Strided Convolution [36]

3.4.7 Pooling Layers in CNN

The pooling layer is a crucial component in convolutional neural networks (CNNs) that performs down-sampling on the feature maps from the preceding layer, generating new feature maps with reduced resolution. This layer significantly diminishes the spatial dimension of the input, serving two primary objectives. Firstly, it diminishes the number of parameters or weights, thereby reducing computational complexity. Secondly, it helps regulate the risk of overfitting in the network [37].

An ideal pooling method is expected to extract relevant information while discarding irrelevant details. Numerous techniques exist for implementing the pooling operation in deep neural networks, each with its own characteristics and suitability for different tasks. Let us discuss some of the well-known and effective pooling methods, aiming to offer insights into their functionalities and applicability in various contexts.

3.4.7.1 Average Pooling

Average pooling, represents a fundamental approach for pooling and feature extraction within convolutional neural networks (CNNs). Illustrated in Fig, an average pooling layer conducts down-sampling by partitioning the input into rectangular pooling regions and subsequently calculating the average values within each region. This process aids in reducing the spatial dimensions of the input while preserving essential information, contributing to effective feature extraction within the network architecture.

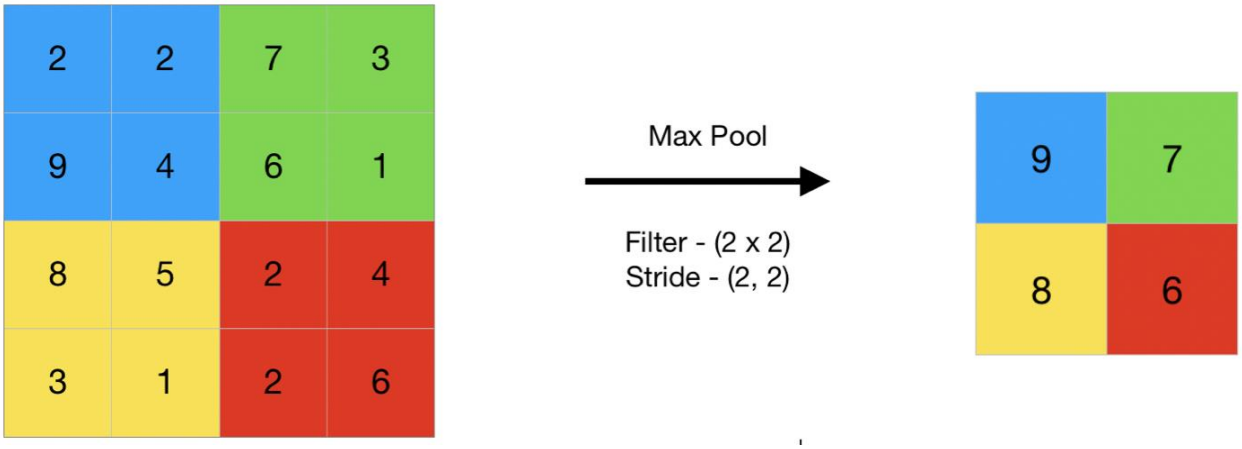


Fig. 3.14 Illustration of Average Pooling [38]

3.4.7.2 Maximum Pooling

The max-pooling layer serves to down-sample the output bands of convolutional layers, thereby reducing variability within the network. This operator functions by forwarding the maximum value within a group of *R* activations. Specifically, the *m*-th max-pooled band comprises *J* related filters represented as

$$p_m = [p_{1,}, \dots, p_{j,m}, \dots, p_{J,m}] \in R^J$$

Where, $p_{j,}$ is computed as: $\max(h_{j,(m-1)N+r})$.

Here, $N \in \{1, \dots, R\}$ represents a pooling shift, allowing for overlap between pooling regions when $N < R$. Consequently, the pooling layer reduces the output dimensionality from K convolutional bands to $M = (K - R)/N + 1$ pooled bands, yielding a resulting layer

$$p = [p_1, \dots, p_M] \in R^{M \cdot J}$$

This process facilitates effective feature extraction while controlling the size of the output, contributing to the overall efficiency of the convolutional neural network architecture.

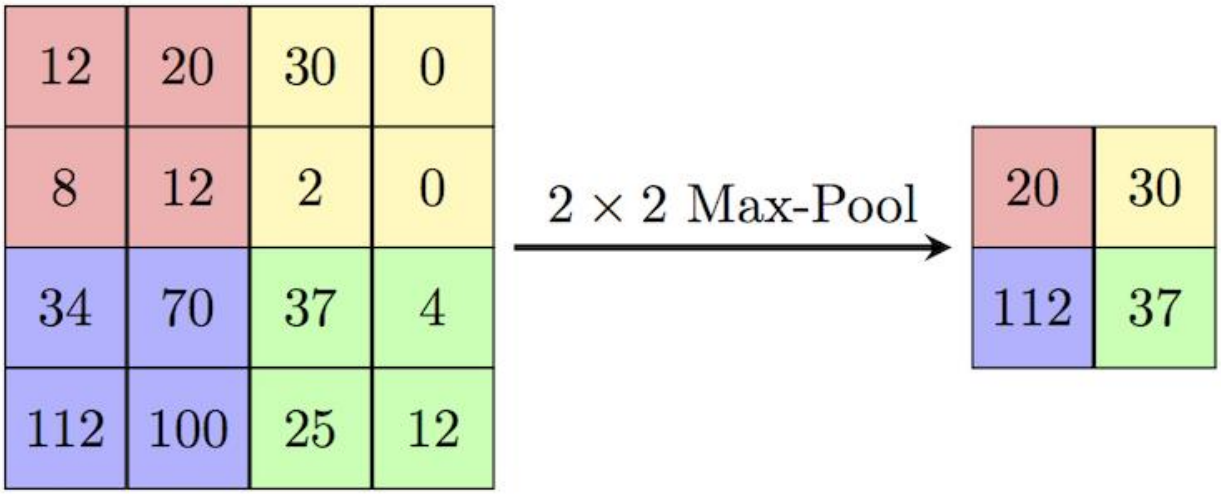


Fig. 3.15 Illustration of Maximum Pooling [39]

3.4.7.3 Stochastic Pooling

In max pooling, the highest activation within each pooling region is chosen, whereas average pooling computes the average of all elements in the pooling region, effectively downweighting areas of high activation by those with low activation. However, a notable drawback of average pooling is its inability to effectively address the issue of preserving important features. To mitigate these limitations, stochastic pooling was introduced[40].

Stochastic pooling tackles the shortcomings of both max and average pooling by applying a multinomial distribution to randomly select values. This approach incorporates non-maximal

activations within the feature map. Initially, probabilities p_i are computed for each region j by normalizing the activations within those regions.

$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k}$$

These probabilities form a multinomial distribution, which is then utilized to select a location l and the corresponding pooled activation a_l based on p. The multinomial distribution facilitates the selection of a location l within the region based on the calculated probabilities.

$$s_j = a_l \; where \; l \sim P(p_1, \ldots, p_{|R_j|})$$

In simpler terms, activations are chosen based on probabilities determined by the multinomial distribution. Each activation has a chance of being selected proportional to its probability. Stochastic pooling introduces a stochastic element, thus helping prevent overfitting. Moreover, it inherits some of the advantages of max-pooling while also incorporating non-maximal activations, thereby offering a more comprehensive approach to feature extraction.

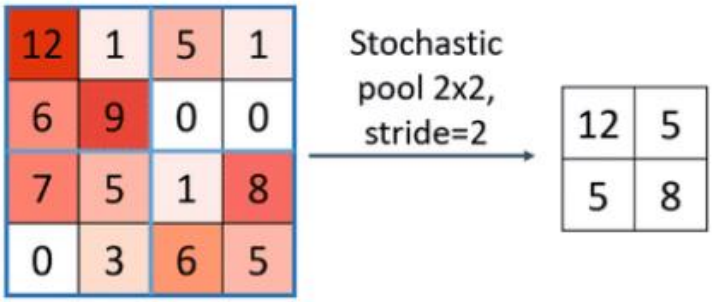


Fig. 3.16 Illustration of Stochastic Pooling [41]

3.5 Conclusion

In conclusion, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for pattern recognition, particularly in the domain of computer vision. Inspired by the organization of the visual cortex, CNNs mimic the human brain's ability to identify patterns in visual data, making them highly effective in tasks such as image classification, object detection, and segmentation. Throughout this chapter, we have explored the fundamental concepts behind CNNs, from their biological inspiration to their practical applications. We have delved into the origins of CNNs, tracing back to the pioneering work of neurophysiologists such as Hubel and Wiesel, and the development of early models like the Neocognitron. We have also discussed the evolution of CNN architectures, from the landmark AlexNet to more recent advancements. Furthermore, we have examined the key operations involved in CNNs, including convolution, pooling, padding, and striding. These operations play a crucial role in extracting meaningful features from input data while reducing computational complexity and controlling the size of feature maps. Additionally, we have explored how CNNs can be applied not only to traditional image data but also to signal data by representing it as image-like inputs. This opens up new possibilities for applying CNNs to a wide range of pattern recognition tasks beyond traditional computer vision applications. In summary, CNNs have revolutionized the field of pattern recognition, offering a versatile and efficient approach to extracting features from complex data. As we continue to explore and innovate in the realm of artificial intelligence, CNNs are sure to remain at the forefront, driving advancements in various domains and unlocking new opportunities for understanding and interpreting the world around us.

3.6. References

[1] Mocz, V., Jeong, S.K., Chun, M. *et al.* Multiple visual objects are represented differently in the human brain and convolutional neural networks. *Sci Rep* **13**, 9088 (2023). https://doi.org/10.1038/s41598-023-36029-z

[2] Yamashita, R., Nishio, M., Gian, R. K., & Togashi, K. (2018, June 22). *Convolutional neural networks: an overview and application in radiology*. Insights Into Imaging. https://doi.org/10.1007/s13244-018-0639-9

[3] Craig, L., & Awati, R. (2024, January 11). *convolutional neural network (CNN)*. Enterprise AI. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network#:~:text=Because%20CNNs%20are%20so%20effective,recognition%20and%20medical%20image%20analysis

[4] *Golden Goose Award to Wiesel and Hubel*. (n.d.). EurekAlert! https://www.eurekalert.org/multimedia/929647

[5] Hubel, D. H., & Wiesel, T. N. (1959, October). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, *148*(3), 574–591. https://doi.org/10.1113/jphysiol.1959.sp006308

[6] Hubel, D. H., & Wiesel, T. N. (1998, March). Early Exploration of the Visual Cortex. *Neuron*, *20*(3), 401–412. https://doi.org/10.1016/s0896-6273(00)80984-8

[7] Li, B., Todo, Y., & Tang, Z. (2022, April 1). *Artificial Visual System for Orientation Detection Based on Hubel–Wiesel Model*. Brain Sciences. https://doi.org/10.3390/brainsci12040470

[8] Fukushima, K. (1980, April 1). *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biological Cybernetics. https://doi.org/10.1007/bf00344251

[9] *Fig. 1. The architecture of the neocognitron.* (n.d.). ResearchGate. https://www.researchgate.net/figure/The-architecture-of-the-neocognitron_fig1_336163445

[10] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021, March 31). *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. Journal of Big Data. https://doi.org/10.1186/s40537-021-00444-8

[11] Celeghin, A., Borriero, A., Orsenigo, D., Diano, M., Guerrero, C., Perotti, A., Petri, G., & Tamietto, M. (2023, July 6). *Convolutional neural networks for vision neuroscience: significance, developments, and outstanding issues*. Frontiers in Computational Neuroscience. https://doi.org/10.3389/fncom.2023.1153572

[12] Adaloglou, N. (2020, July 2). *Understanding the receptive field of deep convolutional networks | AI Summer*. AI Summer. https://theaisummer.com/receptive-field/

[13] LeCun, Y., Bengio, Y., & Hinton, G. E. (2015, May 27). *Deep learning*. Nature. https://doi.org/10.1038/nature14539

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017, May 24). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

[15] *ImageNet*. (n.d.). https://image-net.org/challenges/LSVRC/

[16] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021, March 31). *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. Journal of Big Data. https://doi.org/10.1186/s40537-021-00444-8

[17] Szeliski, R. (2022, January 3). *Computer Vision*. Springer Nature. http://books.google.ie/books?id=QptXEAAAQBAJ&printsec=frontcover&dq=Computer+Vision:+Algorithms+and+Applications,+2nd+ed.&hl=&cd=1&source=gbs_api

[18] Hartley, R., & Zisserman, A. (2003, January 1). *Multiple View Geometry in Computer Vision*. Cambridge University Press. http://books.google.ie/books?id=si3R3Pfa98QC&printsec=frontcover&dq=Multiple+View+Geometry+in+Computer+Vision+Second+Edition&hl=&cd=1&source=gbs_api

[19] Ankit, U. (2022, June 29). *Transformer Neural Networks: A Step-by-Step Breakdown*. Built In. https://builtin.com/artificial-intelligence/transformer-neural-network

[20] *Math 70 Spring 2020: Elements of Multivariate Statistics and Statistical Learning*. (n.d.). https://math.dartmouth.edu/~m70s20/

[21] Solana-Lavalle, G., & Rosas-Romero, R. (2021, January 1). *Classification of PPMI MRI scans with voxel-based morphometry and machine learning to assist in the diagnosis of*

Parkinson's disease. Computer Methods and Programs in Biomedicine. https://doi.org/10.1016/j.cmpb.2020.105793

[22] *ICL-Net: Global and Local Inter-Pixel Correlations Learning Network for Skin Lesion Segmentation*. (2023, January 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9745347

[23] Wang, T., Sun, Q., Pranata, S., Jayashree, K., & Zhang, H. (2022, July 25). *Equivariance and Invariance Inductive Bias for Learning from Insufficient Data*. arXiv.org. https://arxiv.org/abs/2207.12258

[24] *Role of hard coded feature extraction in deep neural networks based classification | ResearchGate*. (n.d.). ResearchGate. https://www.researchgate.net/post/Role-of-hard-coded-feature-extraction-in-deep-neural-networks-based-classification

[25] Taherdoost, H. (2023, September 8). *Deep Learning and Neural Networks: Decision-Making Implications*. Symmetry. https://doi.org/10.3390/sym15091723

[26] Ganesh, P. (2021, December 12). *Types of Convolution Kernels : Simplified - Towards Data Science*. Medium. https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37

[27] Ganesh, P. (2021, December 12). *Types of Convolution Kernels : Simplified - Towards Data Science*. Medium. https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37#:~:text=1D%2C%202D%20and%203D%20Convolutions,input%20can%20have%20multiple%20channels

[28] Panchal, S. (2024, March 21). *No, Kernels & Filters Are Not The Same - Towards Data Science*. Medium. https://towardsdatascience.com/no-kernels-filters-are-not-the-same-b230ec192ac9

[29] Jain, A. (2024, February 12). *All about convolutions, kernels, features in CNN - Abhishek Jain - Medium*. Medium. https://medium.com/@abhishekjainindore24/all-about-convolutions-kernels-features-in-cnn-c656616390a1

[30] *Feature Extraction By Using Deep Learning: A Survey*. (2018, March 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8474912

[31] Ved, M. (2018, July 20). *Feature Selection and Feature Extraction in Machine Learning: An Overview*. Medium. https://medium.com/@mehulved1503/feature-selection-and-feature-extraction-in-machine-learning-an-overview-57891c595e96

[32] Alom, M. Z., Sidike, P., Taha, T. M., & Asari, V. K. (2017, May 7). *Handwritten Bangla Digit Recognition Using Deep Learning*. arXiv.org. https://arxiv.org/abs/1705.02680

[33] *Padding Module: Learning the Padding in Deep Neural Networks*. (2023). https://ieeexplore.ieee.org/document/10021573

[34] G. (2022, March 18). *What is Padding in Neural Network?* GeeksforGeeks. https://www.geeksforgeeks.org/what-is-padding-in-neural-network/

[35] *Stride Based Convolutional Neural Network for Speech Emotion Recognition*. (2021, August 23). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9526320

[36] Antoniadis, P., & Antoniadis, P. (2024, March 18). *Neural Networks: Strided Convolutions | Baeldung on Computer Science*. Baeldung on Computer Science. https://www.baeldung.com/cs/neural-nets-strided-convolutions

[37] *Power Reduction in CNN Pooling Layers with a Preliminary Partial Computation Strategy*. (2018, June 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8585433

[38] G. (2023, April 21). *CNN Introduction to Pooling Layer*. GeeksforGeeks. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

[39] *Papers with Code - Max Pooling Explained*. (n.d.). https://paperswithcode.com/method/max-pooling

[40] Zeiler, M. D., & Fergus, R. (2013, January 16). *Stochastic Pooling for Regularization of Deep Convolutional Neural Networks*. arXiv.org. https://arxiv.org/abs/1301.3557

[41] Nirthika, R., Manivannan, S., Ramanan, A., & Wang, R. (2022, February 1). *Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study*. Neural Computing & Applications. https://doi.org/10.1007/s00521-022-06953-8

Chapter 4

Neural Network Model Compression for Edge Devices – Quantization

4.1 Introduction

The widespread adoption of Deep Neural Networks (DNNs) has revolutionized various fields, including classification tasks such as speech recognition, object recognition, and anomaly detection. However, one of the significant drawbacks of DNNs is their high energy consumption during both training and inference phases [1]. This energy demand necessitates the use of powerful computing resources, such as GPUs and ASICs, often deployed in the Cloud. While Cloud computing offers substantial computational power, it entails transmitting data to a network server, rising concerns about privacy and latency [2].

In response to the limitations of Cloud-based processing, there is a growing interest in performing computations at the edge, directly on devices. This approach eliminates the need for constant connectivity to the Cloud but presents challenges when deploying DNNs on resource-constrained devices like microcontrollers used in Internet of Things (IoT) devices or wearables [3]. These devices typically have limited memory, processing power, and battery capacity, making real-time inference difficult.

To address these challenges, the deployment of DNNs on microcontrollers must adhere to strict constraints on inference response time and model parameters. However, reducing the complexity of DNNs to meet these constraints often compromises accuracy. Deep networks, with their depth and width, tend to achieve better accuracy but require more memory and computational resources [4]. Thus, there exists a trade-off between memory footprint, response time, power consumption, and model accuracy.

Previous research has explored various strategies to optimize DNN deployment on embedded devices. Model compression techniques, a prominent approach, aim to reduce the size of DNN models by removing redundant information without significantly affecting performance. These techniques include pruning, quantization, and weight sharing. Quantization, in particular, involves reducing the precision of weights and activations, thereby reducing memory requirements and computational complexity [5]. By using fixed-point arithmetic instead of floating-point, quantization enables efficient inference on microcontrollers without dedicated floating-point units.

4.2 Numerical Representation and their influence in Neural Network Training and Inference

Within the realm of Deep Neural Network (DNN) model compression, the manner in which numerical values are represented holds a critical place, greatly influencing the efficiency and performance of neural networks. This section delves into the importance of numerical representation, focusing particularly on integers and floating-point numbers within the context of neural networks.

4.2.1 Integers

Integers, the building blocks of numerical computation, are integral in the operation of neural networks. These whole numbers, devoid of fractional components, serve multiple functions within neural networks, including indexing layers, storing biases, and encoding categorical variables. Unlike floating-point numbers, integers offer discrete precision, which can lead to notable memory savings and computational efficiency. In the domain of neural network model compression, integer quantization emerges as a prominent technique. By quantizing weights and activations to lower bit-width integers (such as 8-bit or even binary representations), the storage requirements of neural network parameters are significantly reduced without compromising accuracy. Moreover, integer operations lend themselves to

efficient execution on hardware platforms, making them well-suited for deployment on devices with limited computational resources.

Mathematically, an integer x is defined as any whole number within the set of integers, denoted by Z: $x \in Z$

Where, Z represents the set of integers encompassing positive and negative whole numbers, including zero.

4.2.2 Floating-Point Numbers

Floating-point numbers, characterized by their fractional components, occupy a pivotal position in neural network computations. These numbers, endowed with varying degrees of precision, facilitate the representation of a broad spectrum of real numbers. Within neural networks, floating-point numbers are commonly employed for storing parameters such as weights and activations, as well as for executing arithmetic operations during both training and inference phases.

While floating-point precision affords flexibility and accuracy, it necessitates greater memory consumption and computational complexity. Neural network models endowed with high precision floating-point parameters may demand larger storage capacities and impose heavier processing burdens, constraining their deployment on resource-constrained devices.

In the realm of model compression, techniques such as floating-point quantization aim to alleviate these challenges by reducing the precision of floating-point numbers. By quantizing parameters to lower precision representations (e.g., 16-bit or even lower), the memory footprint of neural network models can be substantially reduced, thereby enabling efficient deployment on embedded systems.

Mathematically, a floating-point number y can be represented in the form:

$$y = (-1)^s \times m \times 2^e$$

where:

- s denotes the sign bit,

- m represents the mantissa,

- e indicates the exponent.

Floating-point numbers adhere to the IEEE 754 standard, which specifies the representation format for single precision (32-bit) and double precision (64-bit) floating-point numbers [6].

4.2.3 Understanding FLOPS – Floating Point Operations and their Influence in Neural Networks

FLOPS (Floating Point Operations Per Second) serve as a crucial metric for assessing the computational performance of neural network models. It quantifies the number of floating-point operations, including additions, multiplications, etc., executed by a neural network per second during inference or training [7].

It's important to note that FLOPS estimation is approximate and may vary due to factors like implementation details, optimizations, and hardware-specific operations. Actual FLOPS during inference or training may also depend on hardware performance and software optimizations [8]. FLOPS and their associated metrics, such as **Multiply-ACCumulate (MAC)** operations, play a crucial role in various aspects of neural network development and deployment [9]:

- **Resource Efficiency**: Optimizing FLOPS can reduce the computational time for training or running neural networks.

- **Memory Efficiency**: MAC operations often influence memory usage, making networks more memory efficient.

- **Energy Consumption and Power Efficiency**: FLOPS and MAC operations contribute to hardware power consumption. Optimizing these metrics can reduce energy requirements, crucial for mobile and embedded devices.

- **Model Optimization**: Techniques like pruning and quantization aim to reduce computational and memory costs by leveraging FLOPS and MACs.

- **Performance Benchmarking**: FLOPS and MACs facilitate model and hardware performance comparisons.

- **Deployment on Edge Devices**: Optimizing FLOPS is critical for real-time applications on resource-constrained edge devices.

- **Battery Life**: Reduced computational costs extend battery life in battery-powered devices.

Understanding these metrics is essential for optimizing neural network performance and facilitating their deployment across diverse hardware platforms.

To estimate FLOPS in neural networks, one typically requires information about the network architecture and input size. The estimation involves several steps:

A. **Determine the network architecture**: Identify the specific neural network architecture utilized in the model, such as ResNet, VGG, or a custom architecture.

B. **Count the FLOPS per layer**: For each layer in the network, ascertain the number of FLOPS required to process a single input element. This depends on the operations performed in the layer, such as convolutional layers involving multiplications and additions or fully connected layers involving matrix multiplications.

C. **Calculate FLOPS for each layer**: Multiply the FLOPS per layer by the number of elements processed in that layer, based on the input and output sizes.

D. **Sum up the FLOPS for all layers**: Aggregate the FLOPS calculated for each layer to obtain the total FLOPS for the neural network model.

4.3 Quantization in Neural Networks

Quantization serves as a technique aimed at reducing the computational and memory expenses associated with inference tasks by representing weights and activations using low-precision data types such as 8-bit integer (int8), in contrast to the conventional 32-bit floating-point (float32) format.

By diminishing the number of bits utilized, the resultant model demands reduced memory storage, potentially consume less energy, and enables faster execution of operations like matrix multiplication through integer arithmetic. Additionally, quantization facilitates the deployment of models on embedded devices, which often support only integer data types.

The fundamental concept underpinning quantization is straightforward: transitioning from a high-precision representation, typically the standard 32-bit floating-point format, for weights and activations to a lower precision data type. Common lower precision data types include:

- float16, with a float16 accumulation data type.
- bfloat16, accompanied by a float32 accumulation data type.
- int16, coupled with an int32 accumulation data type.
- int8, associated with an int32 accumulation data type.

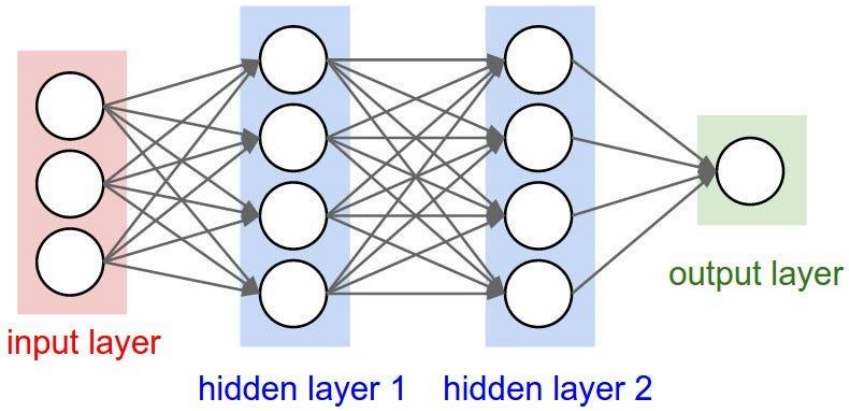


Fig. 4.1 A Neural Net Example Picture [10]

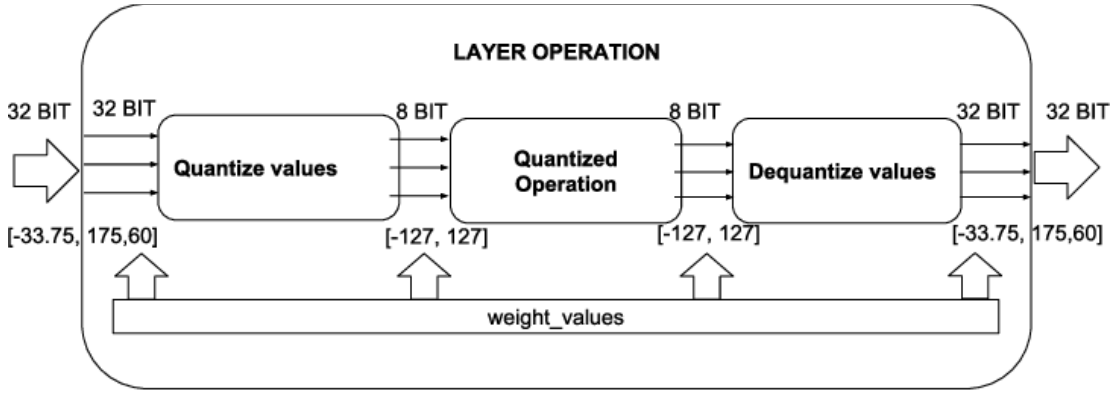


Fig. 4.2 Quantization and Dequantization w.r.t I/P vs O/P Layer of a Neural Network [11]

4.4 Quantization Types

4.4.1. Asymmetric Quantization:

Within the asymmetric mode, the mapping between the minimum and maximum values in the floating-point range to those in the integer range involves the utilization of a zero-point (commonly referred to as quantization bias or offset), alongside the scale factor. This technique ensures that the minimum and maximum values of the float range are appropriately aligned with those of the integer range [13].

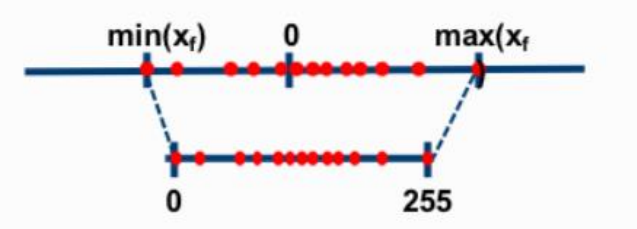


Fig. 4.3 Asymmetric Quantization [12]

4.4.1.1 Asymmetric Quantizer Input:

val_{max}	val_{min}								
43.31	-44.93	0	22.99	-43.93	-11.35	38.48	-20.49	-38.61	-28.02

4.4.1.2 Asymmetric Quantization equations:

$$x_{quantized} = clamp\left(\left[x_{input}/scale\right] + z; 0; (2^n - 1)\right)$$

$$scale = \frac{(val_{\max} - val_{\min})}{2^n - 1}$$

$$z = \left[-1 \times \frac{val_{\min}}{scale}\right]$$

Here, n is the number of bits used for quantization

4.4.1.3 Asymmetric Quantization Output:

255	0	130	196	3	97	241	71	18	49

4.4.1.4 Dequantization Equation:

$$x_{dequantized} = scale \times (x_{quantized} - z)$$

4.4.1.5 Asymmetric Dequantization Output:

43.25	-44.99	0	22.84	-43.95	-11.42	38.41	-20.42	-38.76	-28.03

4.4.2 Symmetric Quantization:

Contrarily, symmetric quantization adopts a different approach. Instead of directly mapping the exact minimum and maximum values of the float range to the quantized range, it selects the maximum absolute value between the minimum and maximum. Additionally, this mode eliminates the use of a zero-point [13]. Consequently, the floating-point range subject to quantization becomes symmetric with respect to zero, mirroring the symmetry of the quantized range.

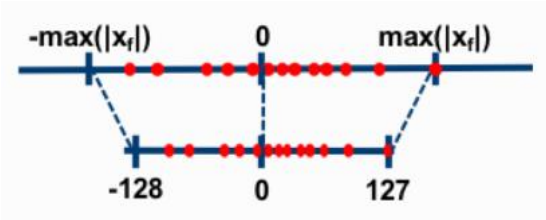


Fig. 4.4 Symmetric Quantization [12]

4.4.2.1 Symmetric Quantizer Input:

43.31	-44.93	0	22.99	-43.93	-11.35	38.48	-20.49	-38.61	-28.02

4.4.2.2 Symmetric Quantization equations:

$$x_{quantized} = clamp\left(\left[x_{input} / scale\right]; -(2^{n-1}-1); (2^{n-1}-i)\right)$$

$$scale = \left[\frac{abs(val_{\max_absolute})}{2^{n-1}-1}\right]$$

4.4.2.3 Symmetric Quantization Output:

122	-127	0	65	-124	-32	109	-58	-109	-79

4.4.2.4 Dequantization Equation:

$$x_{dequantized} = scale \times (x_{quantized})$$

4.4.2.5 Symmetric Deuqantization Output:

43.16	-44.93	0	22.99	-43.86	-11.32	38.56	-20.52	-38.56	-27.95

4.4.3 Comparing Asymmetric and Symmetric Quantization

The primary trade-off between asymmetric and symmetric quantization lies in the balance between simplicity and utilization of the quantized range [13].

Asymmetric quantization ensures full utilization of the quantized range by precisely mapping the minimum and maximum values from the floating-point range to the corresponding extremes of the quantized range. In contrast, symmetric mode may encounter inefficiencies if the float range exhibits a bias towards one side. In such cases, a significant portion of the dynamic range within the quantized range might be allocated to values that are unlikely to occur. For instance, after applying Rectified Linear Unit (ReLU), where the entire tensor becomes positive, symmetric quantization could result in the loss of one bit, as it doesn't account for the asymmetry of the data distribution. However, symmetric quantization offers a simpler implementation compared to its asymmetric counterpart, particularly evident in the convolutional and fully connected layers. In asymmetric mode, the inclusion of zero-points necessitates additional hardware logic, introducing potential overhead in terms of latency, power consumption, and area utilization. The extent of this additional overhead varies depending on the specific implementation details.

4.5 Quantization Granularity

Quantization emerges as a transformative technique aimed at reducing the memory footprint of a model. However, its application often introduces a trade-off, as it can potentially lead to a decline in model accuracy [14]. This dilemma underscores the significance of quantization granularity, which plays a pivotal role in striking a balance between memory optimization and model fidelity.

4.5.1 Per-Channel Quantization

Per-channel quantization represents a refinement in quantization methodology, where distinct quantization parameters are assigned to each channel of a tensor independently. This nuanced approach often mitigates errors associated with quantization, surpassing the performance of per-tensor quantization techniques [15]. In the seNNSEdge project, Motor Control parameters, such as Alpha-Beta Current and Voltage, are employed for training convolutional neural networks (CNNs), depicted in the figure, as distinct features or channels. Due to the diverse properties and functionalities inherent in these parameters, the utilization of Per-Channel Quantization emerges as a highly efficient methodology.

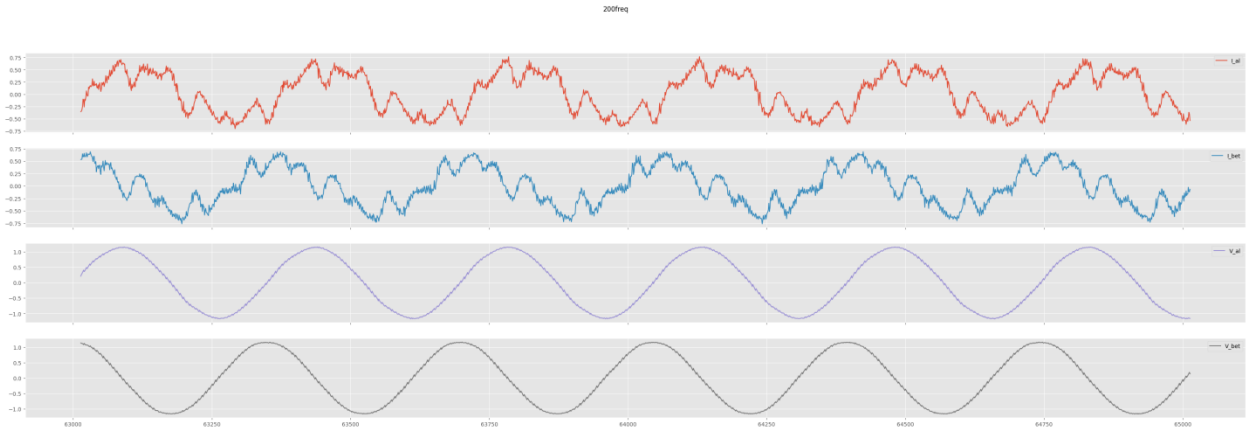


Fig. 4.5 Acquired Data - Motor Control Parameters used for Neural Network Training – Alpha-Beta Current and Voltage

By tailoring quantization parameters to individual channels, per-channel quantization effectively captures variations within each channel, particularly beneficial in convolutional neural network (CNN) models characterized by diverse weight ranges across channels.

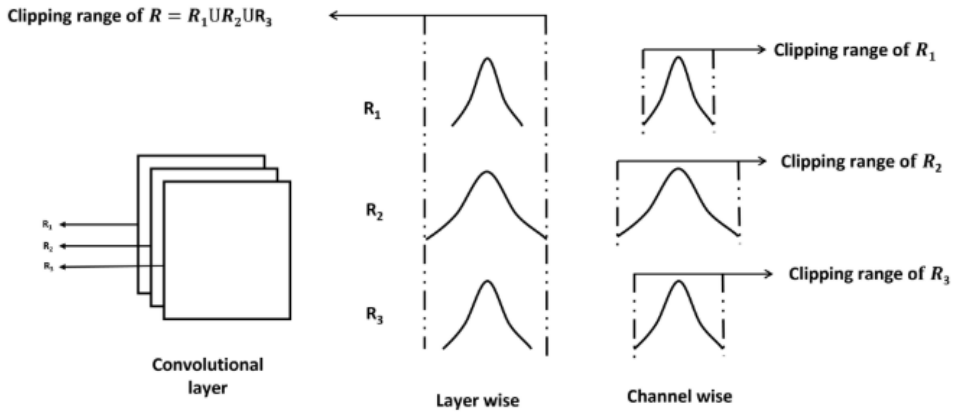


Fig. 4.6 Quantization Granularity [16]

Quantization strategies exhibit variance in granularity: per-tensor quantization applies uniform parameters to all values within a tensor, while per-channel quantization permits customized parameters for each channel. The adoption of per-channel quantization, attuned to the inherent diversity of features within channels, frequently yields superior accuracy outcomes. The choice between these strategies hinges upon the unique attributes of the model and its underlying data distribution.

Per-channel quantization has garnered attention for its ability to restore lost accuracy observed in numerous widely-used models. This efficacy has propelled its integration into popular deep learning frameworks such as PyTorch and TensorFlow, highlighting its prominence in modern quantization practices.

4.6 Quantization Methodologies

4.6.1 Quantization Aware Training

Quantization Aware Training (QAT) involves the retraining of neural networks (NNs) using quantizers to adjust the model parameters, thereby minimizing the drop in accuracy during model quantization. The process of QAT retraining spans multiple epochs, incorporating forward and backward passes utilizing the aforementioned method. The efficacy of these retrained models is contingent upon the number of epochs employed. However, a limitation of the QAT method is its reliance on labeled data for effective quantization.

During the preservation of the quantized model, both the weights and biases undergo quantization and are subsequently saved, along with the quantization parameters of the utilized quantizers from the final epoch. Conversely, solely the quantization parameters of the activation quantizers are saved. Notably, the quantization parameters of the activation quantizers fluctuate with each input batch, as the clipping range of the activations is contingent upon the input data. To determine an appropriate clipping range for the activation quantizers, the clipping ranges of all batches are aggregated, and based on a predetermined criterion, the clipping range of the activation quantizers is computed.



Fig. 4.7 Quantization Aware Training [17]

Quantization Aware Training endeavors to compute scale factors during training. Upon the completion of network training, Quantize (Q) and Dequantize (DQ) nodes are integrated into the graph according to specific rules. Subsequently, the network undergoes further training for several epochs, a process known as Fine-Tuning. During fine-tuning, Q/DQ nodes emulate quantization loss, which is incorporated into the training loss, enhancing the network's resilience to quantization. In essence, QAT excels in preserving accuracy when compared to Post-Training Quantization (PTQ).

4.6.2 Post-Training Quantization

Post Training Quantization (PTQ) refers to the quantization technique applied to an already trained neural network (NN) model without the necessity of retraining. Unlike Quantization Aware Training (QAT), PTQ does not mandate labeled data for quantization. Nonetheless, this approach may potentially result in diminished model performance. Consequently, in scenarios where labeled data is unavailable, PTQ may resort to utilizing representative data to quantize the NN model. Representative data constitutes a sample subset of the training or testing dataset devoid of labels.

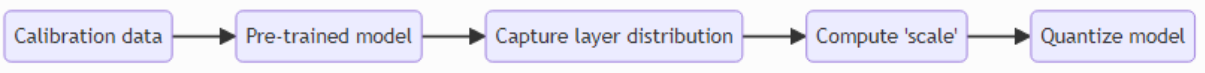


Fig. 4.8 Post-Training Quantization [17**]**

PTQ involves the computation of scale factors after the network has undergone training. A representative dataset is employed to capture the distribution of activations for each activation tensor. Subsequently, this distribution data is utilized to compute the scale value for each tensor. Similarly, the distribution of each weight is leveraged to compute the weight scale.

TensorRT offers a streamlined workflow for PTQ, known as calibration. This process facilitates the quantization of the NN model by utilizing the representative dataset to determine appropriate scale factors, thereby optimizing model performance post-training.

4.7. Experiment - Model Compression for STM32 Microcontrollers for Real-time Inference

4.7.1 Model Compression of a Neural Net

The CNN algorithm within Project seNNsEdge was initially trained using Quantization Aware techniques and subsequently compressed using TensorFlow Lite, enabling efficient inference on Edge Devices such as STM32G4 and STM32H7 Microcontrollers. While the

initial experiments assessed neural net performance without compression, Figures X and Y illustrate the significant compression rate achieved, resulting in more than a **tenfold reduction in memory consumption**.

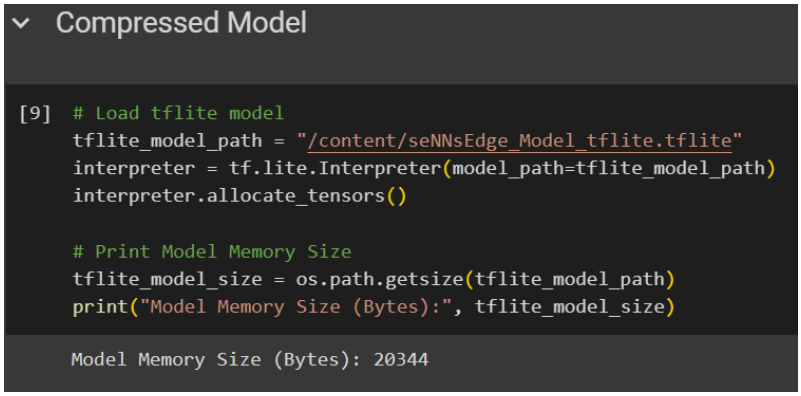


Fig. 4.9 Compressed Model Size – 20344 Bytes

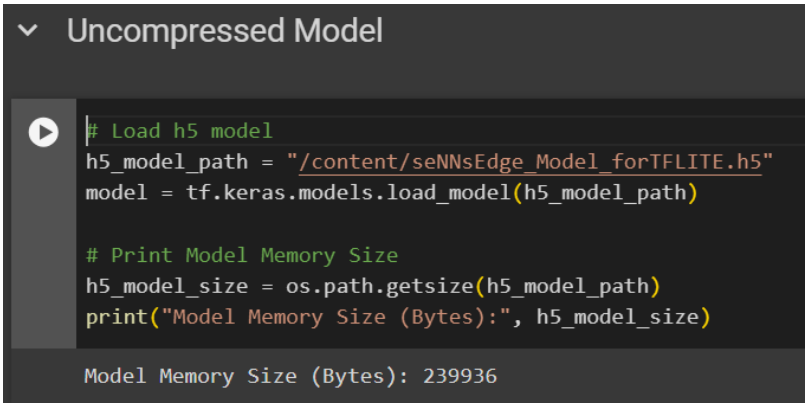


Fig. 4.10 Uncompressed Model Size – 239936

4.8 Conclusion

In conclusion, this chapter has provided a comprehensive exploration of neural network model compression techniques, particularly focusing on quantization. The discussion began by highlighting the challenges associated with deploying deep neural networks on edge devices with limited computational resources, necessitating the development of efficient compression methods. The chapter detailed the significance of numerical representation, distinguishing between integers and floating-point numbers and their roles within neural

networks. It described how quantization, by reducing the precision of weights and activations, significantly lowers memory requirements and computational complexity, making it conducive for deployment on resource-constrained devices. Additionally, the discussion on FLOPS underscored the importance of computational efficiency in neural network development and deployment. Further, the chapter delved into various types of quantization, including asymmetric and symmetric quantization, elucidating their respective methodologies and trade-offs. It also discussed quantization granularity, emphasizing the importance of striking a balance between memory optimization and model accuracy, with a particular focus on per-channel quantization. The chapter concluded by examining quantization methodologies such as Quantization Aware Training (QAT) and Post-Training Quantization (PTQ), highlighting their approaches and implications for model compression. Finally, an experiment showcasing model compression for STM32 microcontrollers demonstrated the practical application of quantization techniques, achieving significant reductions in memory consumption while maintaining inference efficiency.

Overall, this chapter provides an in-depth exploration of quantization as a pivotal technique for enabling efficient deployment of deep neural networks on edge devices, laying the foundation for further exploration and advancements in the field of neural network model compression.

4.9 References

[1] PolyThrottle: Energy-efficient Neural Network Inference on Edge Devices. (n.d.). https://arxiv.org/html/2310.19991v2

[2] A., & A. (2024, January 15). Edge Computing vs Cloud Computing: Differences, Similarities and More. Rishabh Software. https://www.rishabhsoft.com/blog/edge-computing-vs-cloud-computing

[3] The Internet of Things for Applications in Wearable Technology. (2022). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9963553

[4] Chen, C., Zhang, P., Zhang, H., Dai, J., Yi, Y., Zhang, H., & Zhang, Y. (2020, March 1). Deep Learning on Computational-Resource-Limited Platforms: A Survey. Journal of Mobile Information Systems. https://doi.org/10.1155/2020/8454327

[5] R. (2024, January 30). The Power of Model Compression: Guide to Pruning, Quantization, and Distillation in Machine Learning. Medium. https://medium.com/@thisisrishi/the-power-of-model-compression-guide-to-pruning-quantization-and-distillation-in-machine-dbc6d28bd3a3

[6] G. (2020, March 16). IEEE Standard 754 Floating Point Numbers. GeeksforGeeks. https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/

[7] Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. (2023, June 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10203371

[8] Number of FLOPs of Training DNNs for Learning Precoding. (2023, June 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10200945

[9] Approximate Multiply-Accumulate Array for Convolutional Neural Networks on FPGA. (2019, July 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9034956

[10] Dabbura, I. (2022, September 27). Coding Neural Network — Forward Propagation and Backpropagtion. Medium. https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76

[11] Fig. 2. Quantization and Dequantizated transformations in a node. (n.d.). ResearchGate. https://www.researchgate.net/figure/Quantization-and-Dequantizated-transformations-in-a-node_fig1_325804366

[12] Quantization - Neural Network Distiller. (n.d.). https://intellabs.github.io/distiller/algo_quantization.html

[13] Introduction to Quantization — TensorFlow 2.x Quantization Toolkit 1.0.0 documentation. (n.d.). https://docs.nvidia.com/deeplearning/tensorrt/tensorflow-quantization-toolkit/docs/docs/intro_to_quantization.html

[14] C. (2024, January 29). Quantization Granularity - CuriosityDeck - Medium. Medium. https://medium.com/@curiositydeck/quantization-granularity-aec2dd7a0bb4

[15] Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Mart, V. B., & Blankevoort, T. (2021, June 15). A White Paper on Neural Network Quantization. arXiv.org. https://arxiv.org/abs/2106.08295

[16] Mundichipparakkal, J., (2022). Evaluation of Quantization Methods for Neural Networks. https://pure.tue.nl/ws/portalfiles/portal/306604227/Mundichipparakkal_J.pdf

[17] Quantization Aware Training (QAT) — TensorFlow 2.x Quantization Toolkit 1.0.0 documentation. (n.d.). https://docs.nvidia.com/deeplearning/tensorrt/tensorflow-quantization-toolkit/docs/docs/qat.html

Chapter 5

Intelligent Anomaly Detection for Predictive Maintenance

5.1. Introduction

In today's technologically advancing landscape, the quest for efficient maintenance strategies has become paramount across industries. The realm of predictive maintenance and anomaly detection has emerged as a vital aspect, transcending traditional reactive and preventive approaches. This chapter delves into the realm of intelligent anomaly detection for predictive maintenance, exploring its significance, evolution, and application within contemporary industrial frameworks.

Predictive maintenance (PdM) stands at the forefront of modern maintenance paradigms, leveraging data-driven techniques to forecast equipment failures proactively. With the advent of Artificial Intelligence (AI) and Industry 4.0, predictive maintenance strategies have become synonymous with optimized operational efficiency, reduced downtime costs, and heightened reliability [1]. The integration of machine intelligence enables accurate prediction and analysis of machine failures, ushering in an era of enhanced asset management and sustainable practices.

Furthermore, as industries generate vast amounts of data, the need for sophisticated anomaly detection mechanisms becomes imperative. Anomalies, whether in multivariate time-series data or specific applications like unmanned aerial vehicle (UAV) motor control, signify deviations from expected patterns, potentially leading to operational inefficiencies or even accidents [2]. Addressing these anomalies demands a nuanced understanding of data dynamics and advanced analytical tools.

This chapter navigates through the intricacies of anomaly detection, unraveling its significance in the context of predictive maintenance. From historical maintenance practices to the latest advancements in deep learning algorithms, the journey encapsulates the evolution and applicability of anomaly detection techniques. By delving into specific anomalies targeted within UAV motor control applications, such as aerodynamic disturbances and parameter mismatches, the chapter provides insights into real-world challenges and mitigation strategies.

Ultimately, the synthesis of predictive maintenance and intelligent anomaly detection heralds a new era of maintenance excellence, where proactive strategies and data-driven insights converge to optimize asset performance and foster operational resilience.

5.2. Predictive Maintenance in the current Industrial Revolution

In today's technology-driven era, the imperative for predictive maintenance and advanced diagnostics extends beyond aviation to encompass the identification of damages, failures, and operational defects in rotating and moving machines. Implementing such services not only curtails maintenance costs but also extends machine lifespan, ensuring heightened operational efficiency. Moreover, it serves as a preventive measure against potential accidents or catastrophic events.

Predictive maintenance (PdM) is a concept implemented to effectively manage maintenance plans of assets by predicting their failures with data-driven techniques. In these scenarios, data is collected over a certain period to monitor equipment state, aiming to find correlations and patterns for predicting and preventing failures.

The incoming fourth industrial revolution demands a new maintenance paradigm to reduce costs and downtime while increasing system availability and reliability. Traditional maintenance approaches, such as Reactive Maintenance (RM) [3] and Preventive Maintenance (PM), suffer from high prevent and repair costs, inadequate mathematical degradation processes, and manual feature extraction. Predictive Maintenance (PdM) emerges as the solution. Scheduled maintenance checks equipment condition after fixed

intervals; however, this approach increases machine non-functioning time and maintenance costs [4].

The advent of Artificial Intelligence (AI) has revolutionized maintenance across industries, enabling accurate prediction and analysis of machine failures, conserving time and resources. Emphasized by the emergence of Industry 4.0 and smart systems, Predictive Maintenance (PdM) strategies aim to reduce downtime costs and increase manufacturing equipment availability. PdM also promises to bring about new sustainable practices by fully utilizing component useful lives.

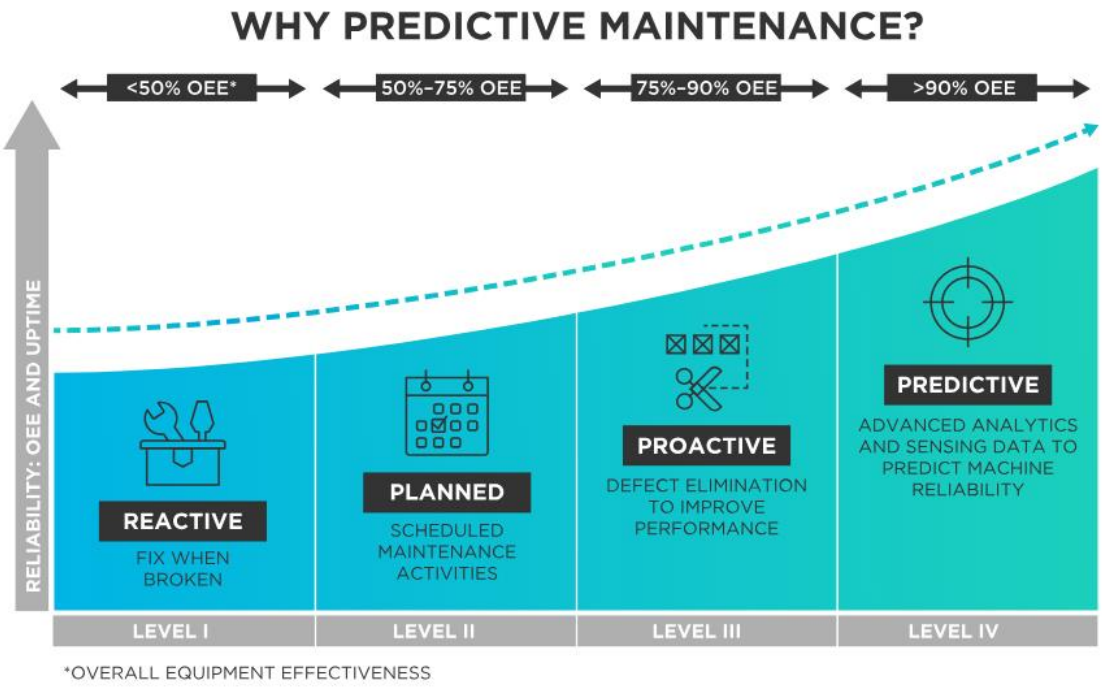


Fig. 5.1 Different maintenance techniques and their effectiveness [5]

5.2.1 Evolution of Maintenance techniques

Maintenance procedures have evolved significantly over time, driven by the need to manage costs and ensure operational efficiency. Historically, maintenance was primarily reactive, addressing issues only when machines failed.

5.2.1.1 Reactive / Breakdown Maintenance

Prior to 1945, and to some extent into the 1950s, maintenance strategies revolved around reactive practices, where machines were allowed to operate until they malfunctioned. Maintenance efforts focused on repairing damaged parts and basic tasks like cleaning and lubrication. This reactive approach, however, resulted in downtime and inefficiencies [3].

5.2.1.2 Preventive Maintenance

Preventive maintenance emerged as a response to the limitations of reactive strategies. It involves scheduled actions aimed at detecting, preventing, or mitigating component deterioration to maintain or extend equipment life. There are two primary types: periodic maintenance and condition-based maintenance [6]. While periodic maintenance may seem costly for many machine components, condition-based maintenance intervenes only when abnormalities are detected, making it more cost-effective. By investing in maintenance activities at the right time, companies can extend equipment life and increase reliability while also reducing overall costs.

5.2.1.3 Predictive Maintenance

Predictive maintenance represents the latest advancement in maintenance strategies, leveraging data and technology to anticipate equipment failures. Unlike preventive maintenance, which operates on a fixed schedule, predictive maintenance relies on real-time monitoring and analysis of equipment conditions to predict when maintenance is needed. This proactive approach offers numerous benefits, including minimizing downtime, optimizing operational efficiency, reducing inventory, and enhancing device reliability. By adopting predictive maintenance, companies can effectively manage maintenance needs while maximizing resource utilization and minimizing costs [7].

5.3 Anomaly Detection for Predictive Maintenance using Machine Intelligence

In an era of increasing automation and advancing connectivity technologies, industries are generating vast amounts of data. Extracting key indicators from this data is crucial for representing the overall system state and detecting anomalies in a timely manner to prevent potential accidents and economic losses.

Detecting anomalies in multivariate time series data presents unique challenges due to the need to consider temporal dependencies and variable relationships simultaneously. Classical methods such as linear models, distance-based approaches, and support vector machines remain relevant but face limitations, especially with larger and more complex systems [8].

Recent advancements in deep learning have shown promising results in addressing these challenges. Deep learning algorithms, inspired by the structure and function of the brain, are adept at learning complex data dynamics without making assumptions about underlying patterns. This flexibility makes them particularly suitable for time-series analysis [9].

Deep learning methods excel at handling large-scale sequences in an supervised, semi-supervised, self-supervised and unsupervised manner, allowing them to identify anomalies effectively [10]. However, their applicability often depends on domain-specific knowledge for optimal deployment, as they can be highly specific to individual use cases.

Anomaly detection, a critical aspect of data mining and quality assurance, continues to evolve with the advent of deep learning techniques. While classical methods remain relevant, deep learning offers new opportunities for detecting anomalies in complex systems, paving the way for enhanced predictive maintenance strategies.

5.3.1 Anomalies in Time-Series Data

Anomalies, often referred to as statistical outliers, represent data points that significantly deviate from the expected pattern, as in, outliers as observations that arouse suspicion of

being generated by a different mechanism [11]. In the context of time-series data, anomalies manifest as unexpected behaviors at specific time steps, diverging significantly from previous observations.

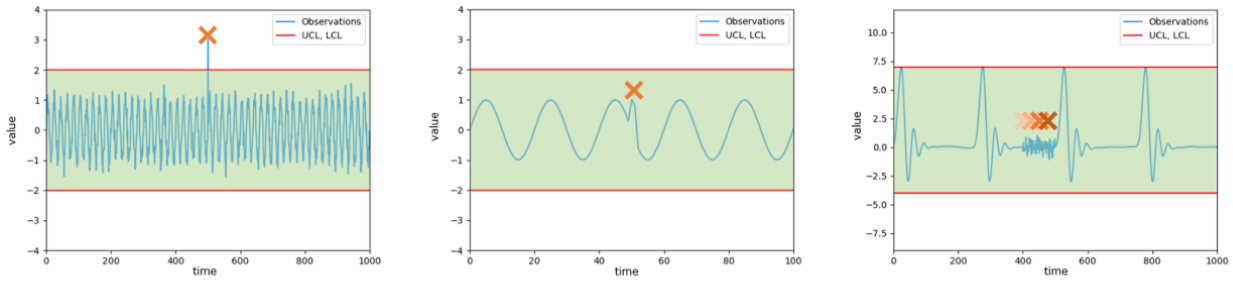


Fig. 5.2 Types of Anomaly in Time-Series Data [12]

5.3.1.1 Point Anomaly

Point anomalies are individual data points or sequences that abruptly deviate from the norm. These anomalies, often caused by sensor errors or abnormal system operations, can appear as temporal noise. Detection methods typically involve setting upper and lower control limits based on prior data, with values outside these limits considered as point anomalies [12].

5.3.1.2 Contextual Anomaly

Similar to point anomalies, contextual anomalies represent data points or sequences observed over a short time. However, they do not deviate from the normal range in the same abrupt manner. Instead, they exhibit deviations from the expected pattern or shape within a given context, making them challenging to detect [12].

5.3.1.3 Collective Anomaly

Collective anomalies refer to sets of data points that gradually exhibit a different pattern from normal data over time [12]. While individual values may appear normal, collectively they

deviate from the expected pattern. Detecting collective anomalies requires considering long-term contexts as they may not be easily recognizable at once.

5.3.1.4 Challenges in Multivariate Time-Series Data

In multivariate time-series data, anomalies require additional consideration due to the relationship between variables along with the time axis. As the number of variables increases, diverse patterns emerge, leading to ambiguous distinctions between normal and abnormal states. Detecting anomalies solely based on individual univariate time-series data may result in inaccurate results, as anomaly points can be obscured by other normal variables. Addressing such challenges involves dimensionality reduction, feature extraction, or utilizing complex models capable of detecting various patterns [13].

5.4 Types of Anomalies Targeted to Detect in Unmanned Aerial Vehicle Motor Control Applications

5.4.1 Aerodynamic anomaly

From their inception, unmanned aerial vehicles (UAVs) have found extensive application in both military and civilian domains, owing to their adaptable configuration and stable performance [15]. Despite their manifold benefits, UAVs confront numerous design challenges pertaining to stability and flight efficacy. For instance, when navigating urban landscapes, quadrotor aircraft typically encounter aerodynamic speeds not surpassing 5 m/s at an altitude of 20 m [15]. During such scenarios, aerodynamic interferences between adjacent rotors become pronounced, compounded by the influence of incoming airflow. Primarily, the horizontal and vertical aerodynamic components exert significant influence on inducing roll perturbations. This arises from the thrust differential across the aircraft's span, with the horizontal component predominantly affecting velocity vectors and the vertical component altering angle of attack [16]. The interaction of rotor wakes and vortices leads to the distortion, entanglement, and fragmentation of wakes upon encountering aerodynamic disturbances and anomalies.

Consequently, the downward flow of UAVs becomes disordered, culminating in heightened power consumption [17]. Ultimately, aerodynamic disturbances exert adverse effects on UAVs, inducing fluctuations in thrust and power, thereby impacting flight efficiency. Nonetheless, implementing appropriate anomaly detection measures can mitigate power consumption and potentially enhance propulsion system efficiency, and also prevent major accidents, particularly in the presence of aerodynamic disturbances. Presently, research on UAVs primarily concentrates on trajectory tracking and attitude control, with barely adequate attention directed towards power requirements or aerodynamic based anomalies.

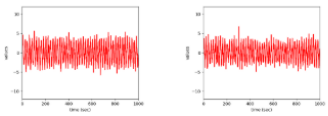
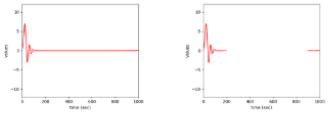
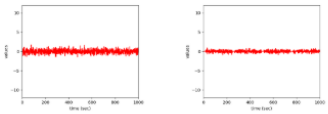
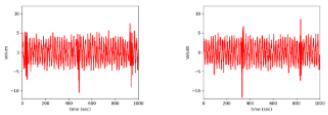
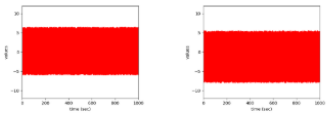
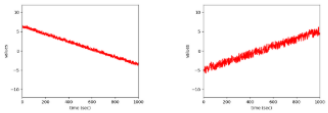
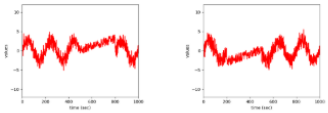
Anomaly patterns	Description	Examples
Normal (assumption)	The amplitude and frequency are stable over time steps, and the time response is symmetrical	
Missing	Most/all of the data are missing, and the time/frequency response becomes 0	
Minor	Compared to normal sensor data, the vibration amplitude is very small	
Outlier	One or more outliers appear in the time response	
Square	The time response oscillates within a limiting range like a square wave	
Trend	The data has an obvious non-stationary and monotonous trend	
Drift	The vibration response is nonstationary, with random drift	

Fig. 5.3 Description of the detailed classification of anomalies in time-series data [14]

5.4.1.1 Quadrotor Aerodynamic Model

The adjacent rotors exhibit clockwise and counterclockwise rotations, offsetting the torque generated by their motion. The flow pattern of a quadrotor UAV is depicted in Fig. A quadrotor aircraft can attain vertical motion, pitch, roll, yaw, and other orientations by adjusting rotational speeds.

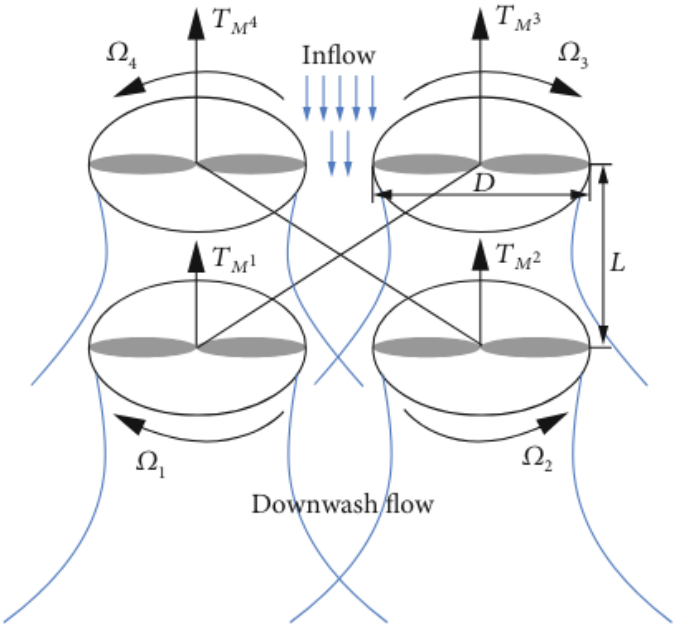


Fig. 5.4 Quadrotor-Unmanned Aerial Vehicle Dynamics [18]

In the absence of aerodynamic disturbances, the thrust of the quadrotor can be characterized as:

$$T = \sum_{i=1}^{4} T_{M_i} = \sum_{i=1}^{4} b\left(\Omega_i\right)^2$$

$$where, (b > 0; i = 1, 2, 3, 4)$$

Here,

T represents thrust,

Ω denotes rotational speed,

b is the proportionality constant,

L stands for the distance between the rotating centers of two adjacent rotors,

D signifies the diameter of the rotor.

5.4.1.2: Rotor Aerodynamic Disturbance Model

Each rotor of the UAV experiences additional thrust due to the influence of aerodynamic disturbances. The wind disturbance model for a rotor is illustrated in Fig.

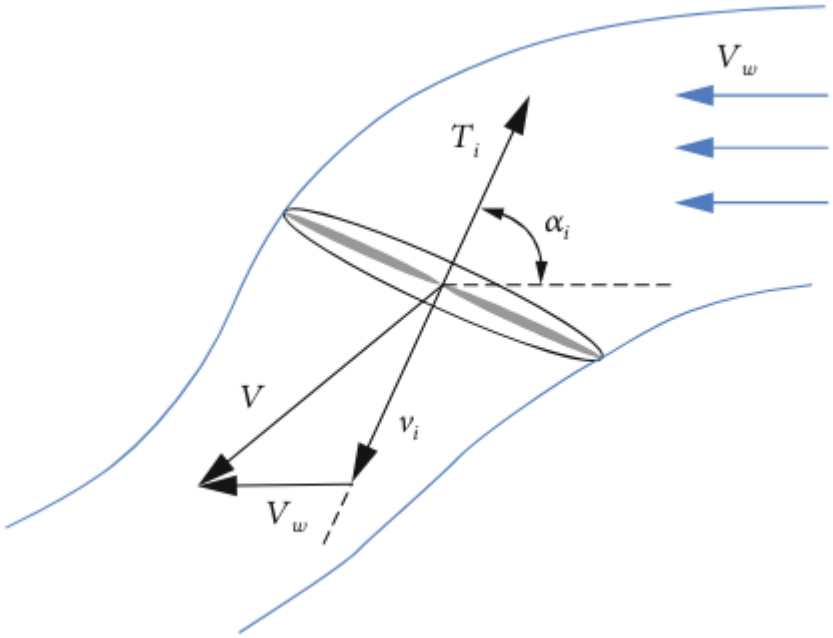


Fig. 5.5 Rotor Aerodynamic Disturbance Model [18]

The total thrust Ti for each rotor can be expressed as:

$$T_i = 2\rho A V v_i$$

$$V = V_w + v_i$$

Here,

ρ denotes air density,

A represents the area of the rotor disk,

Vi signifies the induced speed of the rotor,

Vw represents wind speed,

V denotes the total induced speed.

Furthermore, the total induced speed V can be transformed as:

$$V = \sqrt{(V_w \cos a_i + v_i)^2 + (V_w \sin a_i)^2}$$

Where,

Vw represents wind speed,

a_i (with $0 \leq a_i \leq \pi$) denotes the angle between the rotor axis and the wind direction.

Clearly, the wind is vertical when $a_i = 0$, and horizontal when $a_i = \pi/2$. According to the momentum theory, power can be expressed as:

$$P = T_i(v_i + V_w \cos a_i)$$

Thus, wind from any direction can be categorized into horizontal and vertical components. For a MAV immersed in a wind field, the horizontal component affects the velocity vector of the rotor, while the vertical component influences the angle of attack of the aircraft. This paper applies both horizontal and vertical wind to investigate the effects of aerodynamic disturbances on the quadrotor's aerodynamic characteristics. The diagram illustrating wind direction is presented in Fig.

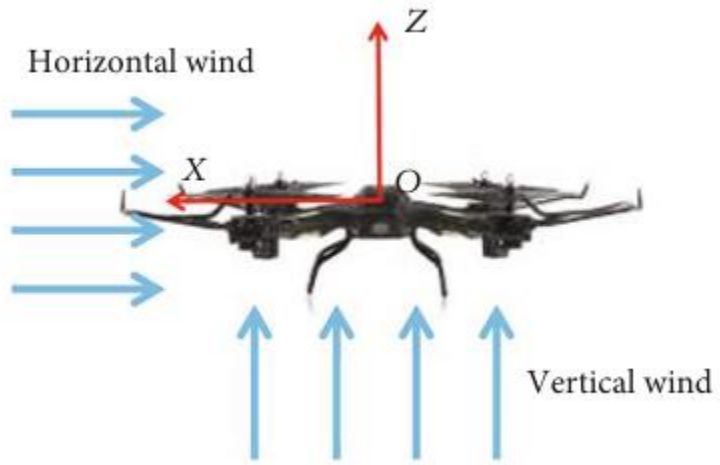


Fig. 5.6 Aerodynamics Direction[18]

5.4.2 Control algorithm based parameter mismatch or Uncontrollability anomaly

In practical applications, motor resistances experience variation with changes in temperature, causing deviations in both stator resistance (Rs) and rotor resistance (Rr) from their nominal or initial values. The accuracy of Rs is particularly critical for the reference model, as precise knowledge of this parameter is essential for obtaining accurate estimations of stator flux components. Similarly, the Field Oriented Control (FOC) technique relies on rotor parameters such as Rr or the rotor time constant (Tr). Any discrepancies between the actual rotor parameters and those used in the controller can result in incorrect estimations of rotor position and rotational speed [19]. This, in turn, can lead to poor control performance and potentially unstable motor operation, particularly at low speeds.

These underlying anomalies disrupt proper functionality, leading to what we term an "Uncontrollability Anomaly." In essence, this anomaly manifests as challenges in effectively controlling the motor's behavior due to discrepancies in critical parameters such as resistance and time constant. To address such issues, it's essential to accurately characterize motor parameters and implement robust control strategies capable of adapting to varying operating conditions.

5.4.3 Mechanical Asymmetry based anomaly

Misalignments within shafts present significant challenges in the operation of rotating machinery, often resulting in heightened dynamic loads and vibrations. These disturbances can lead to premature equipment deterioration and energy inefficiency. Research indicates that poorly aligned machinery can incur downtime costs ranging from 20% to 30% in industrial settings. Moreover, approximately 70% of vibration issues stem from misalignment, exacerbating energy consumption.

Various diagnostic techniques have been employed to detect and monitor shaft misalignments early on, including vibration analysis, wireless sensor technology, model-based fault diagnosis, and motor current signature analysis. However, many of these investigations have focused on motors powered by direct mains or V/Hz drives.

The exploration of fault detection in mechanical systems driven by sensorless voltage source drives has been relatively limited in prior studies. This limitation is primarily attributed to the compensatory actions of the drive's control system, which mitigates dynamic effects resulting from faults while introducing additional noise to the collected data. Consequently, frequency components typically utilized for fault detection may be obscured by drive interventions. Furthermore, the high harmonic content in the terminal output parameters, notably stator current and voltage, complicates the identification of subtle changes in current signatures [20].

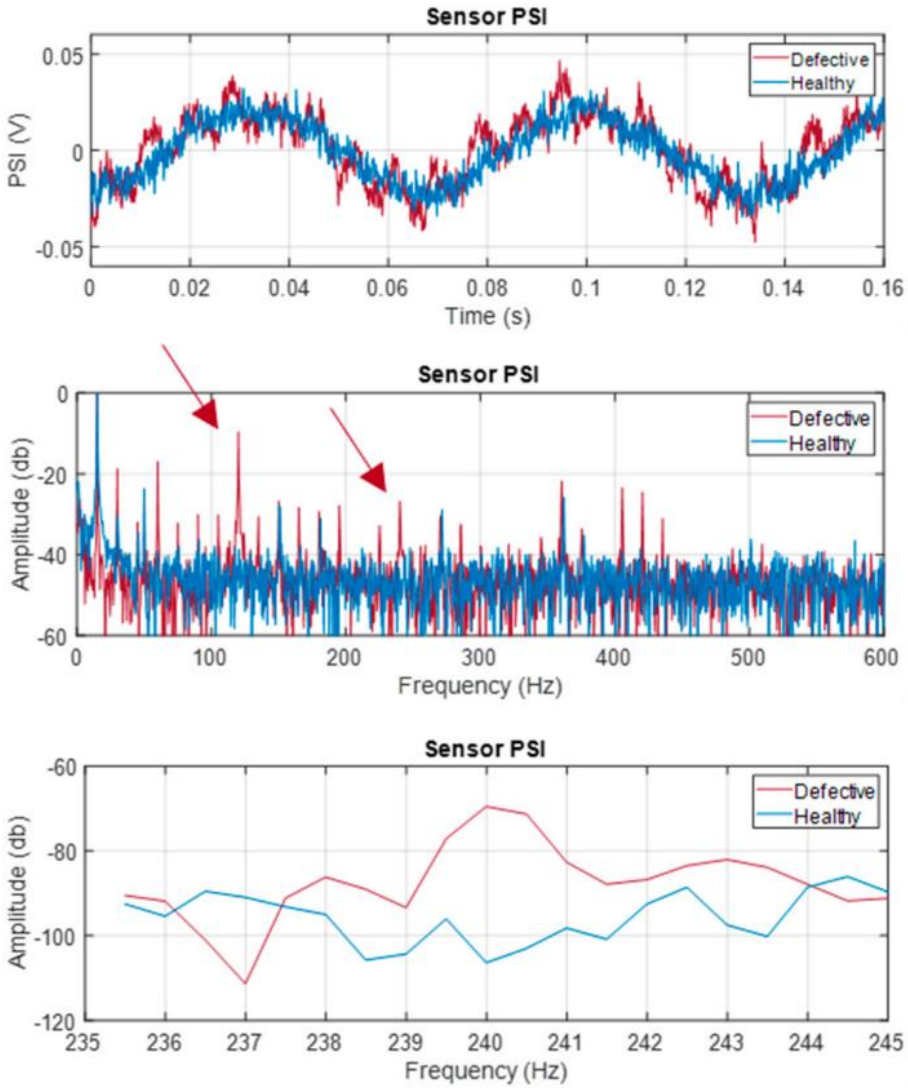


Fig. 5.7 Experimental assessment of position sensor breakdown faults, specifically at no-load operation, utilizing a misaligned piezoelectric sensor. The investigation involved analyzing waveform outputs from the piezoelectric sensor PSI, frequency spectra, and narrow frequency spectra [21].

5.5 Conclusion

In conclusion, Chapter 5 has traversed through the intricate landscape of intelligent anomaly detection for predictive maintenance, shedding light on its significance, evolution, and application within contemporary industrial frameworks. As industries navigate the demands of technological advancement, the quest for efficient maintenance strategies has become paramount. Predictive maintenance (PdM), propelled by Artificial Intelligence (AI) and Industry 4.0, stands as a beacon of optimized operational efficiency, reduced downtime costs, and heightened reliability. The synthesis of predictive maintenance and intelligent anomaly detection heralds a new era of maintenance excellence, where proactive strategies and data-driven insights converge to optimize asset performance and foster operational resilience. From historical maintenance practices to the latest advancements in deep learning algorithms, the chapter has unraveled the evolution and applicability of anomaly detection techniques. Furthermore, specific anomalies targeted within UAV motor control applications, such as aerodynamic disturbances, parameter mismatches, and mechanical asymmetry, have been dissected, offering insights into real-world challenges and mitigation strategies. By addressing these anomalies with precision and foresight, industries can mitigate power consumption, enhance propulsion system efficiency, and prevent major accidents, thereby maximizing resource utilization and minimizing costs.

In essence, the chapter underscores the transformative potential of predictive maintenance and intelligent anomaly detection in ushering in a new era of maintenance excellence, where data-driven insights empower industries to navigate the complexities of modern industrial landscapes with confidence and resilience.

5.6 References

[1] Zonta, T., Da Costa, C. A., Da Rosa Righi, R., De Lima, M. J., Da Trindade, E. S., & Li, G. (2020, December 1). *Predictive maintenance in the Industry 4.0: A systematic literature review*. Computers & Industrial Engineering. https://doi.org/10.1016/j.cie.2020.106889

[2] Bell, V., Moral Arce, I., Mase, J., Rengasamy, D. et al., "Anomaly Detection for Unmanned Aerial Vehicle Sensor Data Using a Stacked Recurrent Autoencoder Method with Dynamic Thresholding," SAE Int. J. Aerosp. 15(2):219-229, 2022, https://doi.org/10.4271/01-15-02-0017

[3] *Proactive/reactive approach for maintenance tasks in time critical systems*. (2005). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/1612626

[4] *Impact of operation mode in terms of reactive power regulations on technical condition and reliability of generating equipment operation*. (2022, March 17). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9731385

[5] *Spotfire | Predictive Maintenance: Enhancing Equipment Reliability and Reducing Downtime*. (n.d.). Spotfire. https://www.spotfire.com/glossary/what-is-predictive-maintenance

[6] *Preventive maintenance! Do we know what it is?* (2000). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/882968

[7] *Machine Learning approach for Predictive Maintenance in Industry 4.0*. (2018, July 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8449150

[8] Rairkar, G. (2024, January 10). *Anomaly Detection for Predictive Maintenance | GAVS Technologies*. GAVS Technologies. https://gavstech.com/blogs/anomaly-detection-for-predictive-maintenance/#:~:text=Anomaly%20Detection%20Machine%20Learning%20Techniques%2FMethods&text=This%20includes%20tasks%20such%20as,a%20suitable%20format%20for%20analysis.&text=In%20this%20step%2C%20relevant%20features,are%20selected%20from%20the%20dataset

[9] *Anomaly Detection with Convolutional Autoencoder for Predictive Maintenance*. (2022, November 26). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10068441

[10] *System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants*. (2014, September 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7005202/similar#similar

[11] Bajaj, A. (2023, August 22). *Anomaly Detection in Time Series*. neptune.ai. https://neptune.ai/blog/anomaly-detection-in-time-series#:~:text=What%20are%20anomalies%2Foutliers%20and,generated%20by%20a%20different%20mechanism.%E2%80%9D

[12] *Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9523565

[13] Yeshchenko, A., & Mendling, J. (2024, February 1). *A survey of approaches for event sequence analysis and visualization*. Information Systems. https://doi.org/10.1016/j.is.2023.102283

[14] *Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9523565

[15] *Unsupervised Anomaly Detection for Self-flying Delivery Drones*. (2020, May 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9197074

[16] *Aerodynamic Curiosities and Anomalies*. (n.d.). https://archive.aoe.vt.edu/mason/Mason_f/Curiosities.html

[17] Khan, S., Liew, C. F., Yairi, T., & McWilliam, R. (2019, October 1). *Unsupervised anomaly detection in unmanned aerial vehicles*. Applied Soft Computing. https://doi.org/10.1016/j.asoc.2019.105650

[18] Lei, Y., Huang, Y., & Wang, H. (2021, April 21). *Effects of Wind Disturbance on the Aerodynamic Performance of a Quadrotor MAV during Hovering*. Journal of Sensors. https://doi.org/10.1155/2021/6681716

[19] *Sensorless Field-Oriented Control for Open-End Winding Five-Phase Induction Motor With Parameters Estimation*. (2021). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9399810

[20] Srinath, R., Sarkar, A., & Sekhar, A. S. (2016, February 1). *Instability of asymmetric shaft system*. Journal of Sound and Vibration. https://doi.org/10.1016/j.jsv.2015.10.008

[21] Papathanasopoulos, D. A., Giannousakis, K., Dermatas, E., & Mitronikas, E. D. (2021, April 16). *Vibration Monitoring for Position Sensor Fault Diagnosis in Brushless DC Motor Drives*. Energies. https://doi.org/10.3390/en14082248

Chapter 6

Experiment - Sensorless Pattern Recognition and Anomaly Detection in the Intelligent Edge Device (seNNsEdge)

6.1 Introduction

This chapter presents an innovative framework tailored for diagnosing electric motor-driven control systems, adept at identifying both external and internal influences without reliance on extra physical sensors. By harnessing operational electrical signatures and cutting-edge deep learning techniques within edge devices, the framework offers a sophisticated approach to condition monitoring and pattern recognition. Through extensive experimentation, pattern recognition has been optimized for microcontroller platforms, enhancing precision and efficiency. This integration of traditional motor control algorithms with advanced AI methodologies not only improves system performance but also opens avenues for electric motors to evolve into intelligent sensors, thereby contributing significantly to the ongoing cyber-physical industrial revolution.

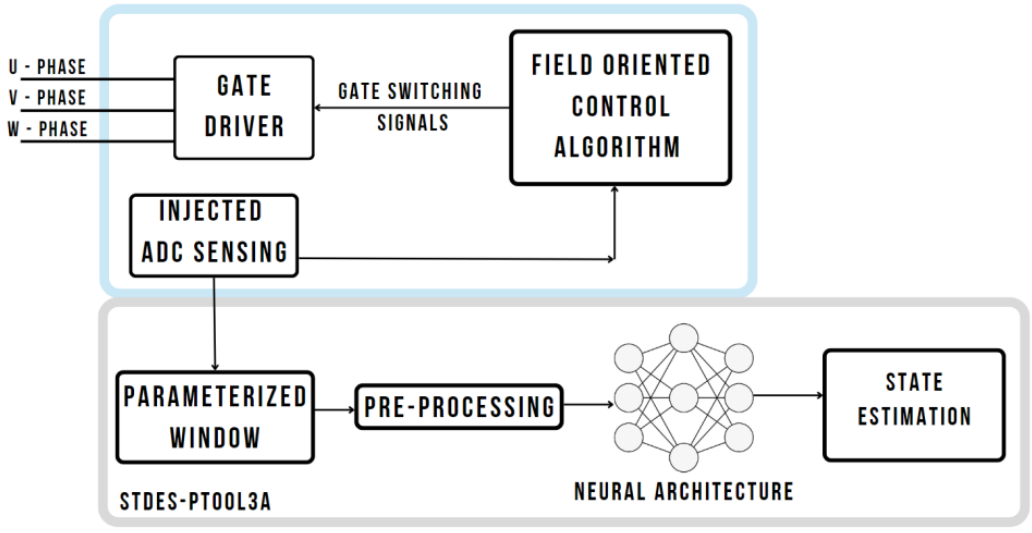


Fig. 6.1 Overview of the framework

6.2 Schematic Summary / Architecture

Figure 6.1 and Figure 6.2 provides a simplified overview of our project's architecture. Throughout the following chapters and experiments, we'll delve deeper into this architecture, breaking down each component and operation. But let's start with a broad overview. The operation of the BLDC/PMSM motor revolves around several key conditions: Uncontrollability, Asymmetrical load, Aerodynamic Disturbance, and Healthy Operation. We manage these conditions through sensorless vector control, specifically employing the field-oriented control (FOC) algorithm with the STEVAL-PTOOL3A Motor Driver. The motor driver's FOC algorithm relies on various electrical signatures, including 3-phase currents (Ia, Ib, Ic) and their corresponding Clarke and Park transformations (I_alpha, I_beta, I_d, I_q), along with similar transformations for back-emf (V_alpha, V_beta, V_d, V_q), Electrical Angle, and Phase current amplitude. These electrical signatures serve as inputs for training a Deep Convolution Neural Network (DCNN) algorithm, crucial for pattern recognition. We train the model with a focus on Quantization Aware techniques to compress the model effectively, using Tensorflow Lite. Once compressed, the model is converted into C-kernel format for execution on an STM32H7 microcontroller, enabling real-time inference. Moreover, communication between the STEVAL-PTOOL3A and the STM32H7 edge device occurs via the I2C protocol, ensuring a rapid 1MHz transmission speed, crucial for real-time anomaly detection.

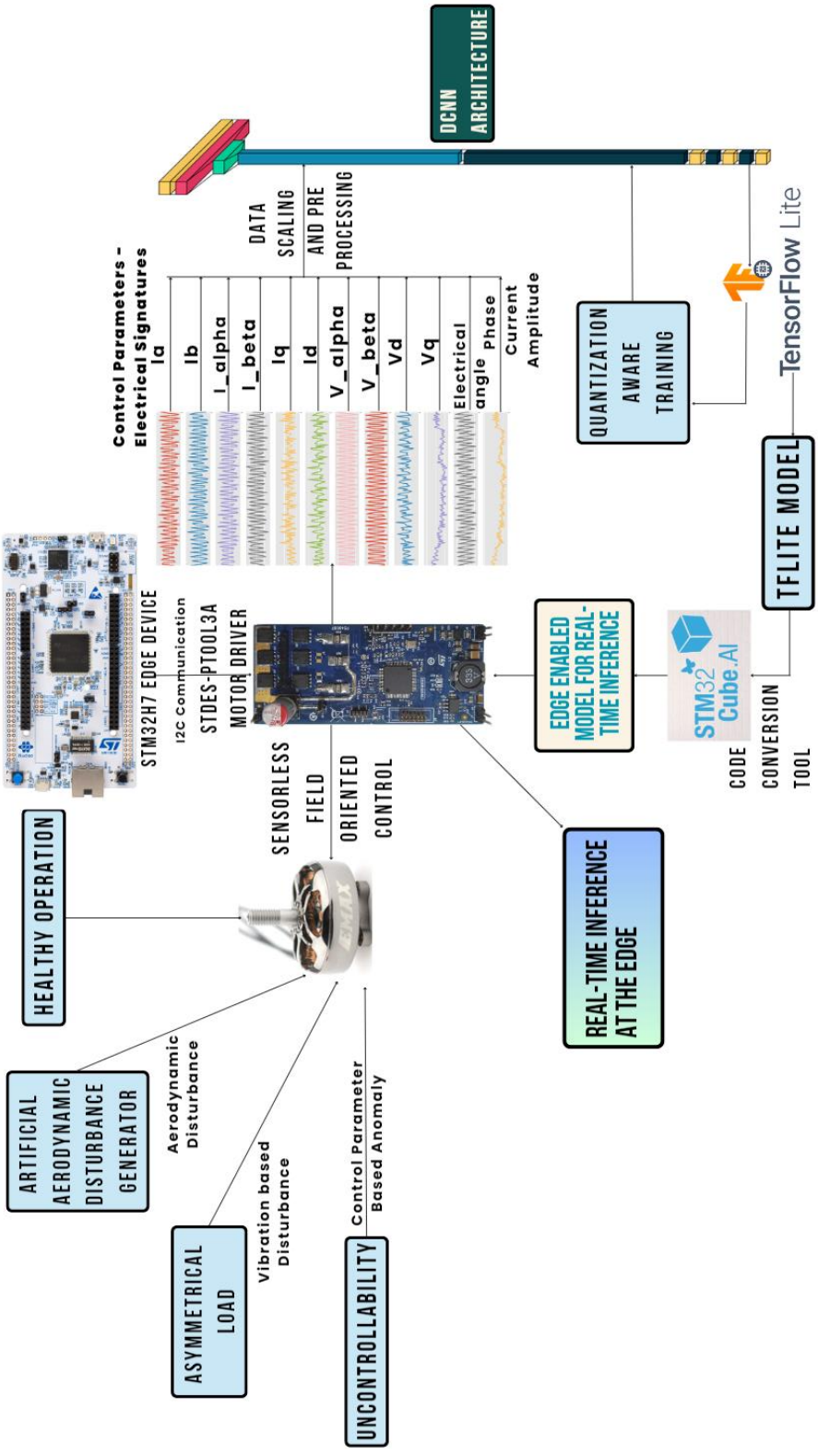


Fig. 6.2 Schematic Architecture of the framework

In the following sections, we will thoroughly examine the entire architecture.

6.3 Experimental Setup

Now, let us delve into the experimental setup, covering the hardware and software frameworks, and explore the system's functionality.

6.3.1 Sensorless control of PMSM/BLDC Motor using Field Oriented Control (FOC) Algorithm

The central component of our system is the Quadcopter motor, controlled using a Voltage Sourced Inverter (VSI), as depicted in Figure 6.3. The system comprise of a Quadcopter Motor - EMAX-ECOII 2400KV Brushless DC motor coupled with 5.1 inch tri-blade flash propellers running at a constant RPM, driven by STMicroelectronics – STDES-PTOOL3A[]. Now let us discuss about the VSI Motor Controller STEVAL-PTOOL3A and its abilities as it is a prime component for our experiment.

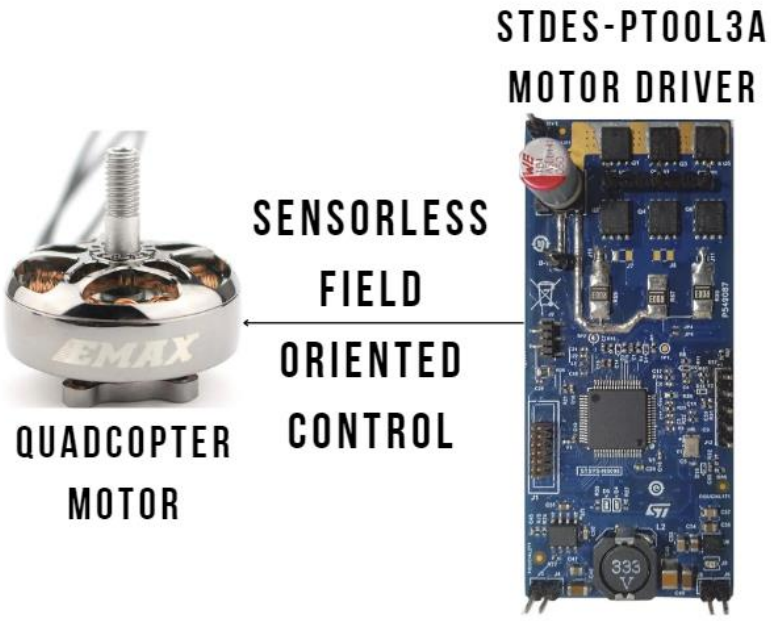


Fig. 6.3

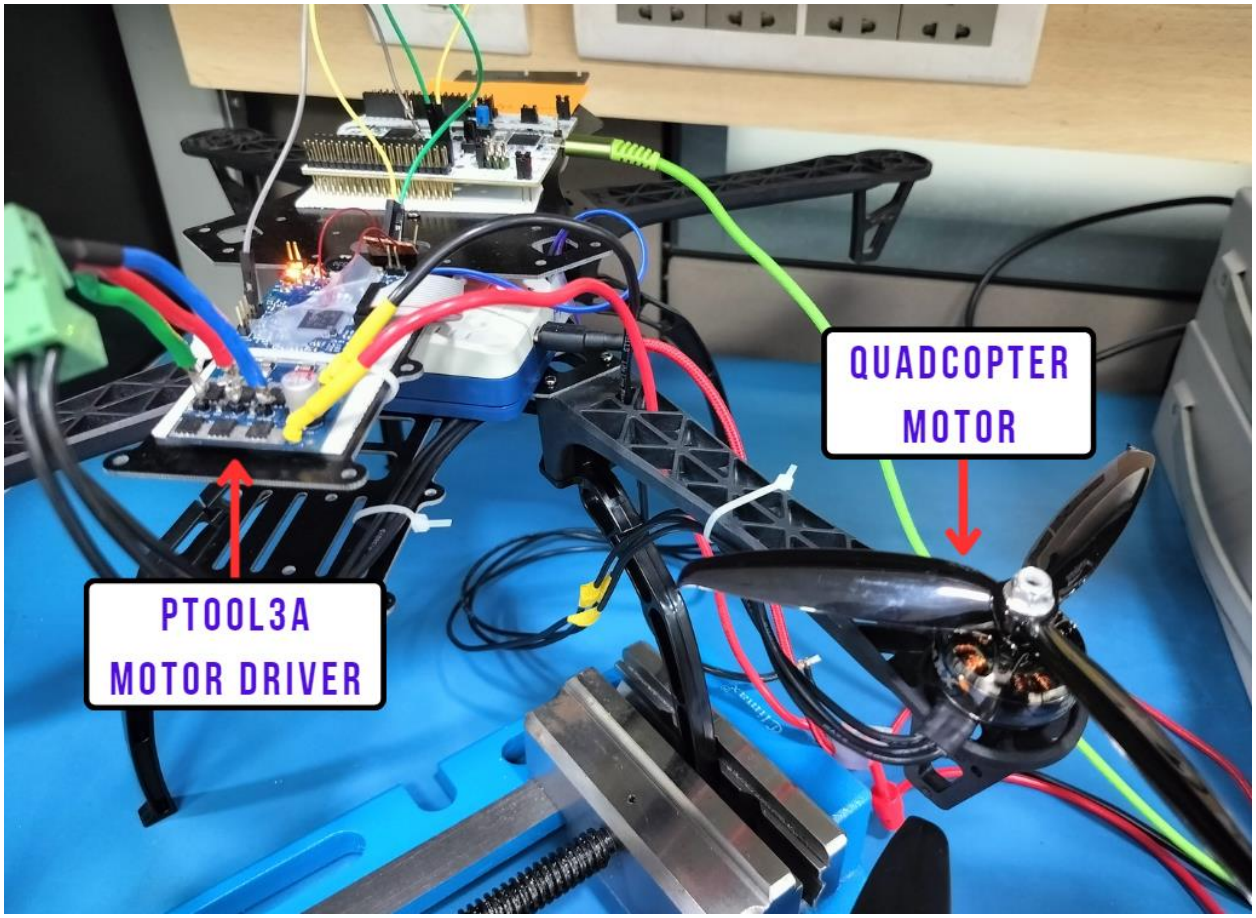


Fig. 6.4 Quadcopter Motor driven by STEVAL-PTOOL3A Motor Driver Board

6.3.2 STEVAL-PTOOL3A

The STEVAL-PTOOL3A exemplar design manifests as a compact configuration purposed for commanding a BLDC/PMSM motor within a handheld battery-operated apparatus, dispensing up to 250W of power. Its functionality hinges on conventional inputs like throttle modulation for motor torque/speed regulation. A compatible 24V/250W BLDC or PMPM motor is interfaceable with the board, capable of yielding heightened power output with either a heatsink attachment or an airflow arrangement to enhance thermal dissipation. This blueprint serves as an optimal solution for robust motor regulation essential in various power tool applications such as drills, grinders, disc cutters, circular saws, leaf blowers, and handheld mowers.

Its compact footprint and computational prowess render it applicable even in diverse settings including drones, wheelchairs, household appliances, and robotic platforms.



Fig. 6.5 STEVAL-PTOOL3A Motor Control Board []

At its core, the board integrates a STM32G431RB microcontroller incorporating integrated operational amplifiers and comparators meticulously tailored for motor control tasks, coupled with a triple half-bridge gate driver commanding six low-voltage high-current MOSFETs. Additionally, the STM32 G4 series MCU accommodates the CORDIC hardware accelerator, engineered to expedite computations pertaining to specific mathematical operations involving vectors and trigonometric functions, proving especially advantageous in motor control contexts. The MCU boasts multiple ADC peripherals, facilitating concurrent sampling of motor phase currents. Furthermore, the board is proficient in utilizing both six-step and FOC algorithms accessible within ST's Motor Control Software Development Kit (MCSDK), ensuring top-tier performance in both six-step and field-oriented motor control scenarios.

6.3.3 STM32-H755ZI-Q (Edge Inference System)

About the IC: The STM32H755xI series leverages the formidable power of the Arm® Cortex®-M7 and Cortex®-M4 32-bit RISC cores, delivering exceptional performance. Operating at speeds of up to 480 MHz for the Cortex®-M7 core and up to 240 MHz for the Cortex®-M4 core, these processors are equipped with a floating-point unit (FPU) supporting both single- and double-precision operations and conversions, in compliance with IEEE 754 standards. Additionally, they boast a comprehensive set of DSP instructions and a memory protection unit (MPU) that fortifies application security.

These devices integrate high-speed embedded memories, featuring a dual-bank Flash memory of 2 Mbytes and up to 1 Mbyte of RAM. This includes 192 Kbytes of TCM RAM, up to 864 Kbytes of user SRAM, and 4 Kbytes of backup SRAM. Furthermore, they offer an expansive array of enhanced I/Os and peripherals linked to APB buses, AHB buses, a 2x32-bit multi-AHB bus matrix, and a multi-layer AXI interconnect that facilitates seamless internal and external memory access.

In terms of functionality, all variants of the series come equipped with three ADCs, two DACs, two ultra-low power comparators, a low-power RTC, a high-resolution timer, and 12 general-purpose 16-bit timers. Moreover, they feature two PWM timers tailored for motor control, five low-power timers, a true random number generator (RNG), and a dedicated cryptographic acceleration cell. Additionally, these devices support four digital filters designed specifically for external sigma-delta modulators (DFSDM) and offer a comprehensive suite of both standard and advanced communication interfaces.

About the Development Board: The STM32 Nucleo-144 board offers an accessible and adaptable platform for users to explore new ideas and create prototypes, leveraging the diverse range of performance and power consumption features inherent to STM32 microcontrollers. With options to tailor configurations according to specific project needs, users can efficiently navigate through different performance-power trade-offs.

Furthermore, for compatible boards, the internal or external SMPS (Switched-Mode Power Supply) significantly reduces power consumption during operation, enhancing overall efficiency.

Facilitating seamless expansion, the board features the ST Zio connector, which extends ARDUINO® Uno V3 connectivity, and ST morpho headers. These interfaces offer a straightforward method for augmenting the functionality of the Nucleo open development platform, supporting a broad spectrum of specialized shields to address diverse application requirement Eliminating the need for additional hardware, the STM32 Nucleo-144 board integrates the ST-LINK debugger/programmer, streamlining the development process and reducing complexity.

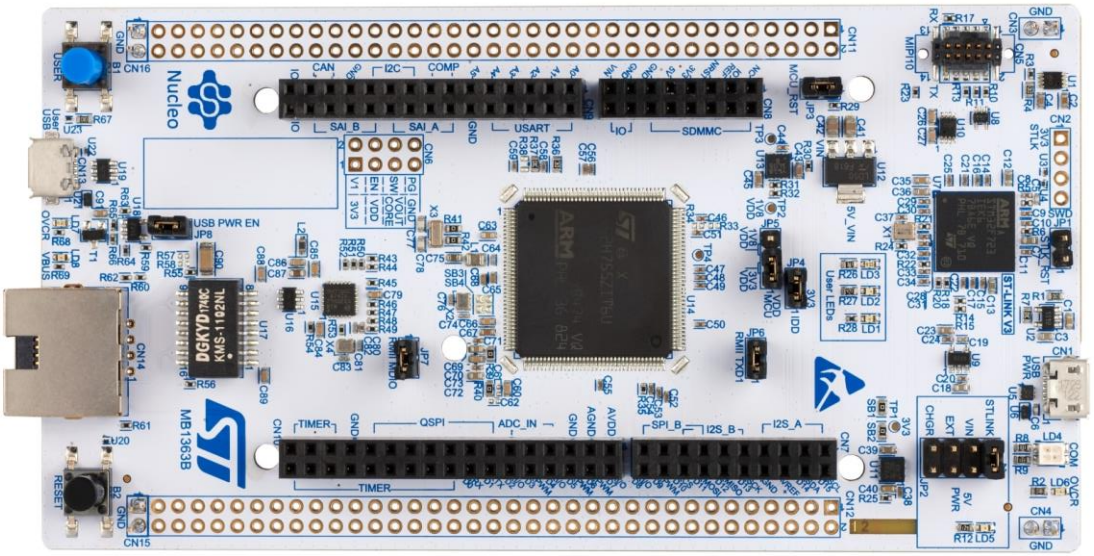


Fig. 6.6 STM32-H755ZI-Q Nucleo Board

Moreover, the board is accompanied by comprehensive free software libraries and examples, accessible through the STM32Cube MCU Package. These resources empower developers with the tools necessary to expedite development cycles and realize innovative solutions efficiently.

6.3.4 Anomaly Generation Experimental Setup

As outlined in the schematic summary, we are exploring **Aerodynamic influences**, **Uncontrollability**, and **Asymmetrical Load-based anomalies**, which will be applied to the Quadcopter motor in action through the PTOOL3A motor driver. This section focuses on detailing the Experimental Setups designed to induce these anomalies and observe their effects on the Quadcopter motor.

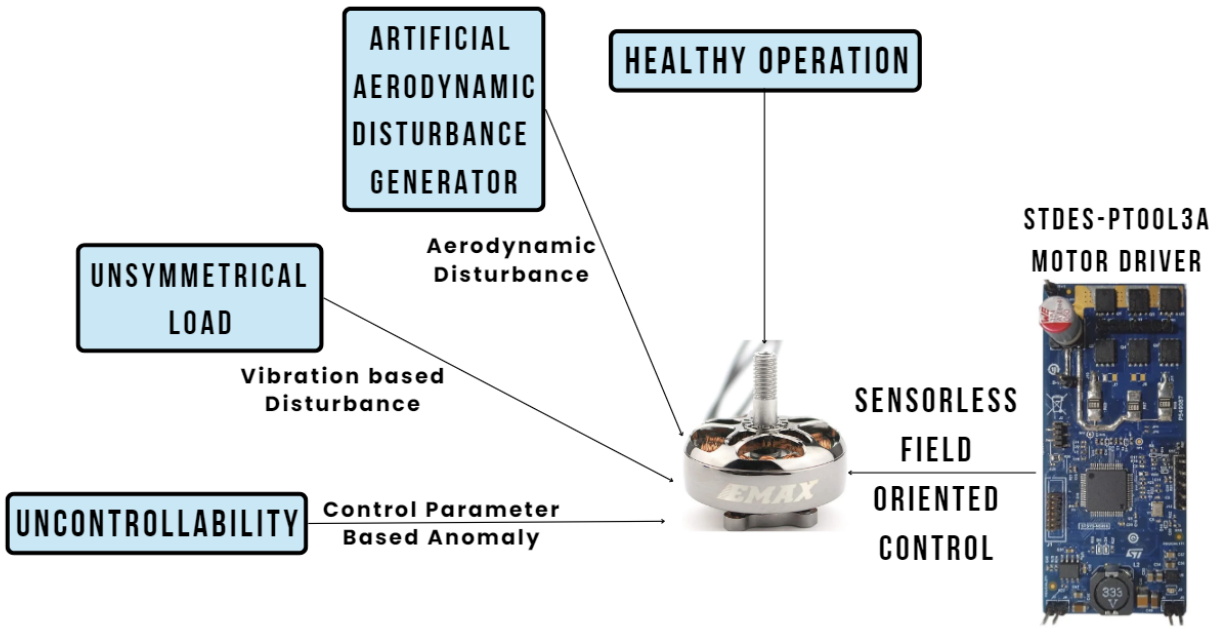


Fig. 6.7 Types of anomalies subjected to the Quadcopter Motor

The rationale behind creating a controlled environment for anomaly detection is twofold: firstly, to comprehend the impact of anomalies on the operational Quadcopter system, and secondly, to gather data on the performance of the system under the influence of these anomalies.

6.3.4.1 Artificial Aerodynamic Disturbance Generator

To facilitate the experiment, we have implemented an Artificial Aerodynamic Disturbance Generator System. Additionally, we have set up a system designed to encounter the aerodynamic disturbance, with its electrical signatures serving as indicators of the disturbance's influence.

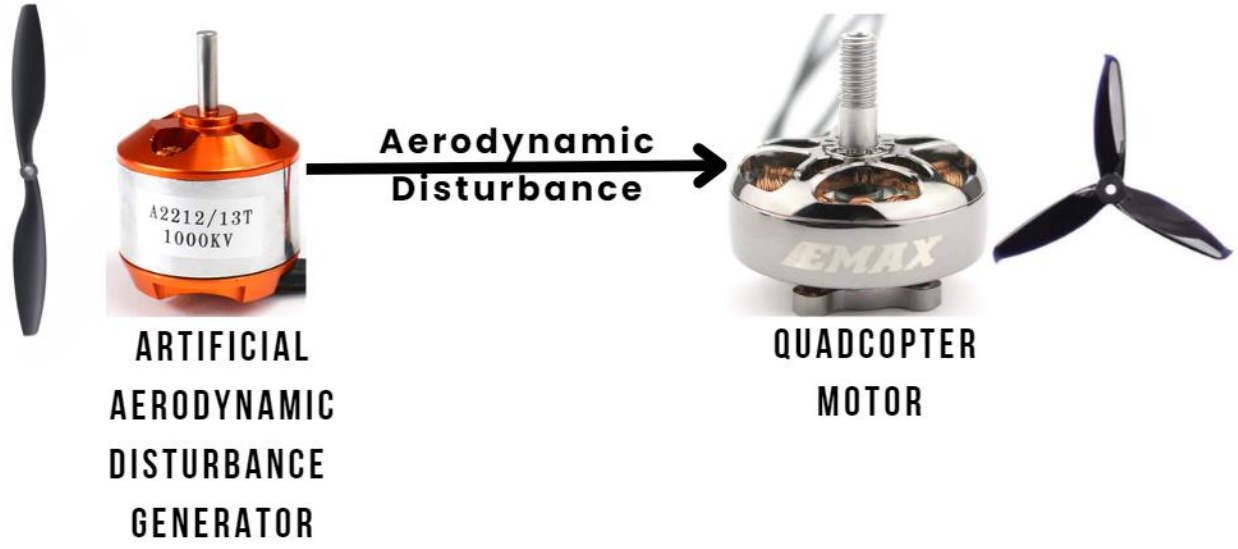


Fig. 6.8 Conjunction between the Artificial Aerodynamic Disturbance Generator and the Quadcopter Motor

The disturbance generator system consists of a 1000KV Brushless DC motor (a generic model) paired with 10-inch aerofoil stable propellers, operating at various controlled RPMs. These components are driven by a generic industrial motor driver.

By adjusting the switching PWM frequency, we can modify the speed of the disturbance motor, generating rotating patterns that impact the operation of the quadcopter motor and disrupt its control loop/algorithm. We meticulously record the data affected by these disturbances for subsequent discussion in later sections.

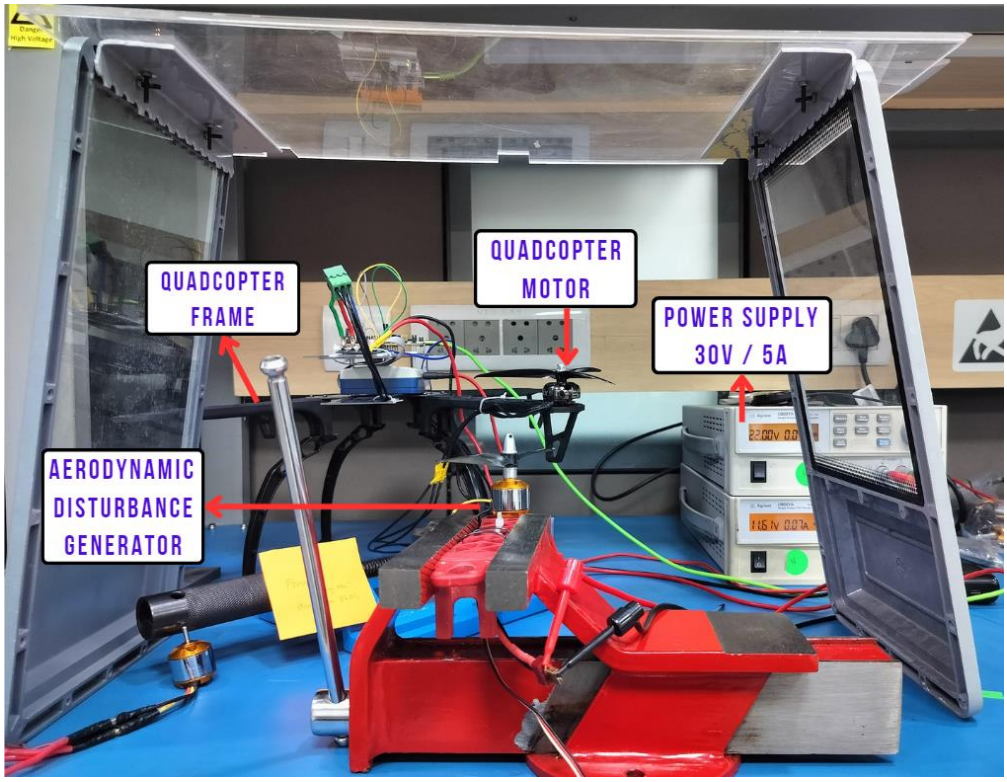


Fig. 6.9 Experimental Setup for Aerodynamic Anomaly Generation

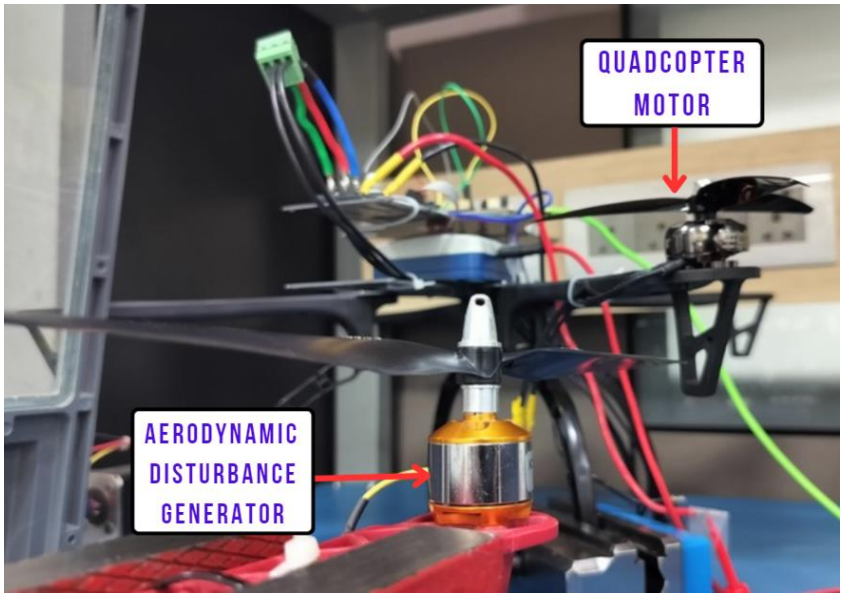


Fig. 6.10 Aerodynamic Disturbance based Anomaly being subjected to the Quadcopter motor

6.3.4.2 Uncontrollability Anomaly

As discussed earlier, we understand motor resistances change with temperature, affecting stator (Rs) and rotor (Rr) resistances. Accurate Rs is crucial for precise stator flux estimation, while FOC relies on rotor parameters like Rr or Tr. Mismatches between actual and controller-used rotor parameters lead to incorrect estimations, causing poor control, especially at low speeds. These deviations, termed "Uncontrollability Anomalies," disrupt motor functionality. Accurate parameter characterization and adaptable control strategies are essential to address these issues. To gather data for this anomaly, we intentionally subjected the Quadcopter motor to mechanical overload and allowed it to run for an extended period. As the motor heated up over time, we observed gradual abnormal operation.

6.3.4.3 Asymmetrical Load-based anomalies

To induce asymmetrical load-based anomalies, our experimental setup incorporates multiple damaged quadcopter propellers, which we mount onto the quadcopter motor. We then drive the motor at various speeds to assess the impact of propeller-induced vibrations on both the quadcopter and the overall system. To comprehensively study the effects of misalignment-related anomalies, we have selected various types of propellers, enabling us to analyze the distinct patterns of influence they exhibit.



Fig. 6.11 Different types of healthy and damaged propellers

6.4 Data Acquisition for acquiring Training Data

The cornerstone of the Sensorless Pattern Recognition and Anomaly Detection in the Intelligent Edge project lies in the acquisition and interpretation of data. This data provides crucial insights into the system's state and its response to external stimuli. In our setup, the primary source of data is the Motor Controller STEVAL-PTOOL3A board. This board implements the control algorithm using Sensorless Field Oriented Control to drive the motor and serves as the interface through which external devices can communicate to ascertain the motor's status. Communication with the PTOOL3A board entails utilizing various communication protocols such as Inter-Integrated Circuit, Serial Peripheral Interface, Universal Synchronous/Asynchronous Receiver Transmitter, among others.

The objective is to extract meaningful information from the PTOOL3A board, which drives the Quadcopter motor. While traditional systems rely on sensors like Hall Sensor, Torque Sensor, or Inertial Measurement Unit to gather sensory data, our system employs Sensorless Field Oriented Control (FOC). This approach allows us to assess the motor's state and health

without the need for additional sensors. Instead, we leverage the motor itself as a sensor. But how does this work? We achieve this by harvesting the internal parameters of the control algorithm or analyzing the electrical signatures to gain insights into the motor's operation. Let's explore these mechanisms visually in Figure 6.12.

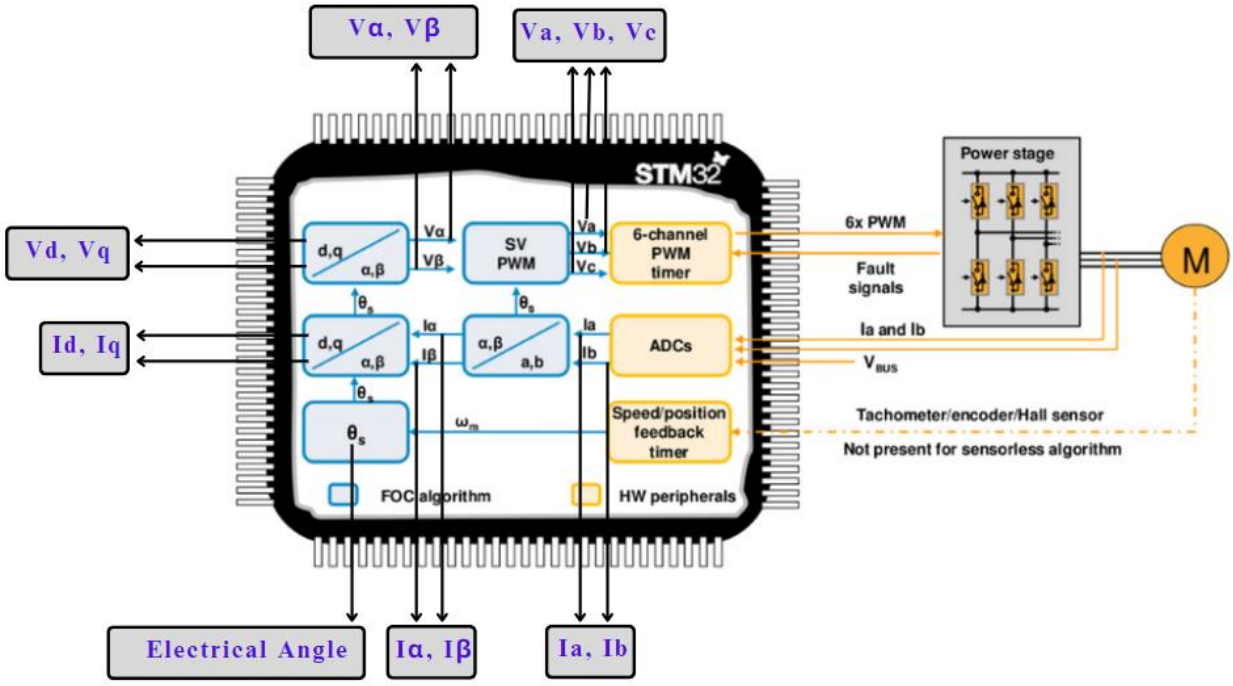


Fig. 6.12 Data Extraction from the FOC Operated Motor Driver



Fig. 6.13 Data buffer extracted from FOC Control Electrical Signatures

As previously discussed in Sensorless Vector Control section, we know that the Field Oriented Control Algorithm relies on shunt currents and executes multiple transformations as part of its operation. Referring to Figure 6.12, we can observe these transformations, represented by the electrical signatures. We extract this data to construct a data buffer, which exhibits distinct properties corresponding to various motor operations.

The data buffer containing electrical signatures derived from the FOC algorithm comprises:

- **Ia, Ib**: These represent the phase currents in the motor windings. In a three-phase motor, Ia, Ib, and Ic correspond to the currents flowing through phases A, B, and C, respectively.

- **Iα, Iβ**: These are the currents represented in a rotating reference frame (dq frame) aligned with the magnetic flux of the motor. Iα represents the component of current aligned with the d-axis (flux-producing axis), while Iβ represents the component of current aligned with the q-axis (torque-producing axis).
- **Id, Iq**: These are the direct and quadrature components of current in the dq frame. Id represents the current component aligned with the d-axis (flux-producing), while Iq represents the current component aligned with the q-axis (torque-producing).

- **Electrical Angle**: This refers to the angle (in electrical degrees) between the rotor position and the reference frame used for control. It is crucial for accurately controlling the motor's behavior.

- **Vd, Vq**: These are the direct and quadrature components of voltage in the dq frame. Vd represents the voltage component aligned with the d-axis (flux-producing), while Vq represents the voltage component aligned with the q-axis (torque-producing).

- **Va, Vb, Vc**: These represent the voltages applied to the motor phases A, B, and C, respectively. They are the input voltages provided by the motor controller to drive the motor.

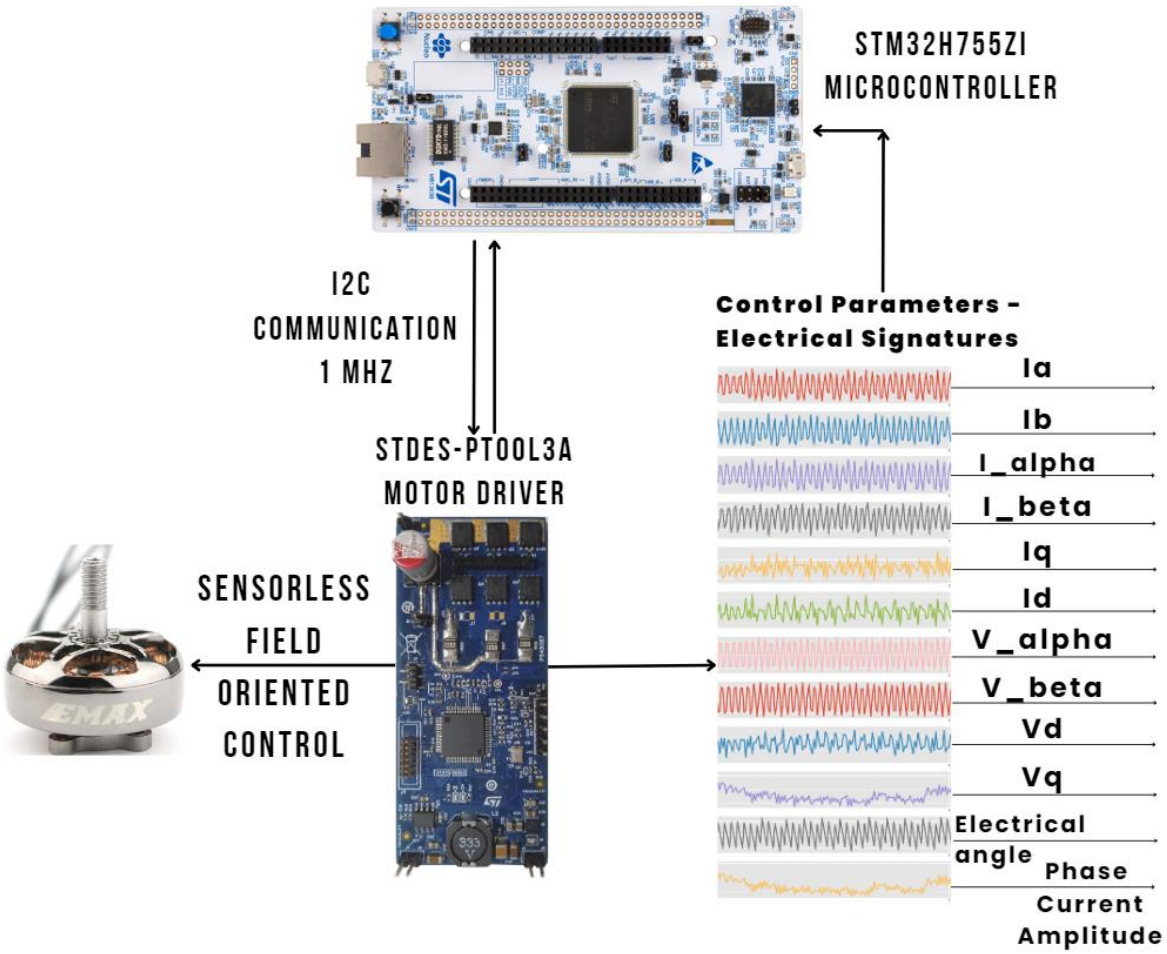


Fig. 6.14 Electrical Signature Extraction from PTOOL3A Board

The STEVAL-PTOOL3A executes the FOC algorithm using the Motor Control Workbench Software Definition Kit developed by STMicroelectronics. This software relies on control parameters/electrical signatures for its operation. The PTOOL3A Motor Driver implements the control algorithm and simultaneously transmits the data buffer, as illustrated in Figure 6.14, to the STM32H755ZI microcontroller at a sampling frequency of 1MHz via Inter Integrated Circuit (I2C) protocol at Fast Mode Plus (1MHz). The PTOOL3A acts a Master and STM32H755ZI acts as slave in the I2C protocol communication.

The STM32H755ZI receives the data buffer and stores it in a circular buffer while concurrently emptying the buffer by writing the data into a Comma Separated Value (CSV) file. This file is transmitted to a computer via USART (Universal Synchronous Asynchronous Protocol) at a baud rate of 921600.

Figure 6.15 presents an outlook of the CSV file structure. The data buffer can be customized to include additional information such as timestamps and signal names, enhancing the depth of insights gained from the data.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	EMAX_3kSPD	60535	285	-264	285	140	312	-54	3398	2090	-295	3979	-6525	3989
2	EMAX_3kSPD	60536	-227	232	-227	-137	159	211	-3557	1756	-312	3954	-28805	3966
3	EMAX_3kSPD	60537	61	24	61	-63	73	47	473	-3931	-307	3947	14323	3959
4	EMAX_3kSPD	60538	269	-280	269	168	312	-53	3154	2407	-292	3958	-7567	3967
5	EMAX_3kSPD	60539	-307	264	-307	-128	266	199	-3775	1279	-319	3973	-30195	3985
6	EMAX_3kSPD	60540	109	-56	109	1	41	100	1257	-3785	-313	3976	12219	3988
7	EMAX_3kSPD	60541	253	-248	253	140	256	-133	2445	3150	-290	3978	-10257	3987
8	EMAX_3kSPD	60542	-307	200	-307	-54	310	31	-3970	15	-306	3958	32004	3970
9	EMAX_3kSPD	60543	-35	88	-35	-82	43	-78	2184	-3321	-313	3962	9492	3974
10	EMAX_3kSPD	60544	381	-280	381	103	224	-325	1593	3630	-287	3955	-12825	3964
11	EMAX_3kSPD	60545	-371	184	-371	1	365	-66	-3968	-385	-308	3974	30953	3986
12	EMAX_3kSPD	60546	-99	88	-99	-45	-27	-106	2219	-3284	-298	3952	9399	3963
13	EMAX_3kSPD	60547	365	-312	365	149	250	-305	1452	3709	-289	3973	-13249	3983
14	EMAX_3kSPD	60548	-355	360	-355	-211	401	95	-3868	-889	-301	3957	29621	3968
15	EMAX_3kSPD	60549	93	-120	93	84	19	123	2897	-2736	-319	3972	7056	3984
16	EMAX_3kSPD	60550	141	-120	141	57	70	-135	684	3908	-298	3957	-15360	3967
17	EMAX_3kSPD	60551	-211	472	-211	-424	388	270	-3634	-1630	-315	3970	27545	3982
18	EMAX_3kSPD	60552	221	-328	221	251	51	330	3155	-2381	-313	3940	5914	3952
19	EMAX_3kSPD	60553	-115	8	-115	57	61	112	462	3980	-315	3995	-15997	4006
20	EMAX_3kSPD	60554	-147	440	-147	-424	365	260	-3420	-2052	-312	3975	26315	3988
21	EMAX_3kSPD	60555	413	-568	413	417	233	538	3562	-1730	-318	3947	3875	3959

Fig. 6.15 Databuffer acquired from source PTOOL3A Motor Driver to PC as a CSV file

6.5 Comprehensive Data Acquisition for Aerodynamic, Uncontrollability, and Asymmetry-Based Anomalies

In this section, we'll explore the acquired data buffer, specifically the electrical signatures corresponding to various anomalies as waveforms.

6.5.1 Acquiring Data for Healthy Operation

Prior to delving into anomaly-induced waveforms, it's crucial to establish a reference waveform database for the healthy operation of the Quadcopter motor. We collected a dataset under optimal conditions, ensuring the absence of any anomalies, allowing the motor to operate smoothly. Figure 6.16 illustrates the waveforms of the electrical signatures during the healthy condition.

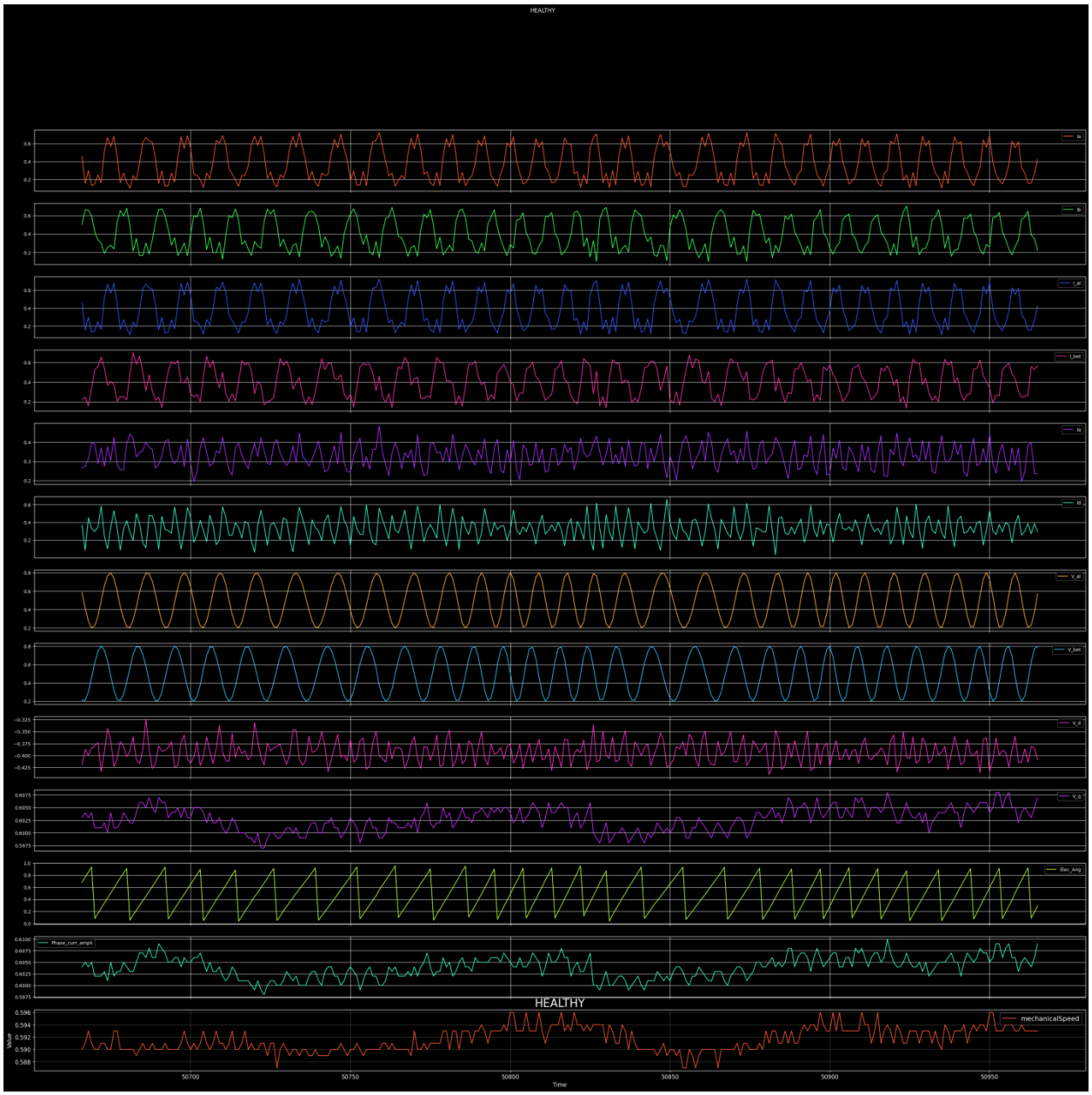


Fig. 6.16 Healthy Waveform Waveforms

6.5.2 Acquiring Data for Aerodynamic Anomaly

Figure 6.17 illustrates the waveforms of the electrical signatures during the aerodynamic anomaly effect.

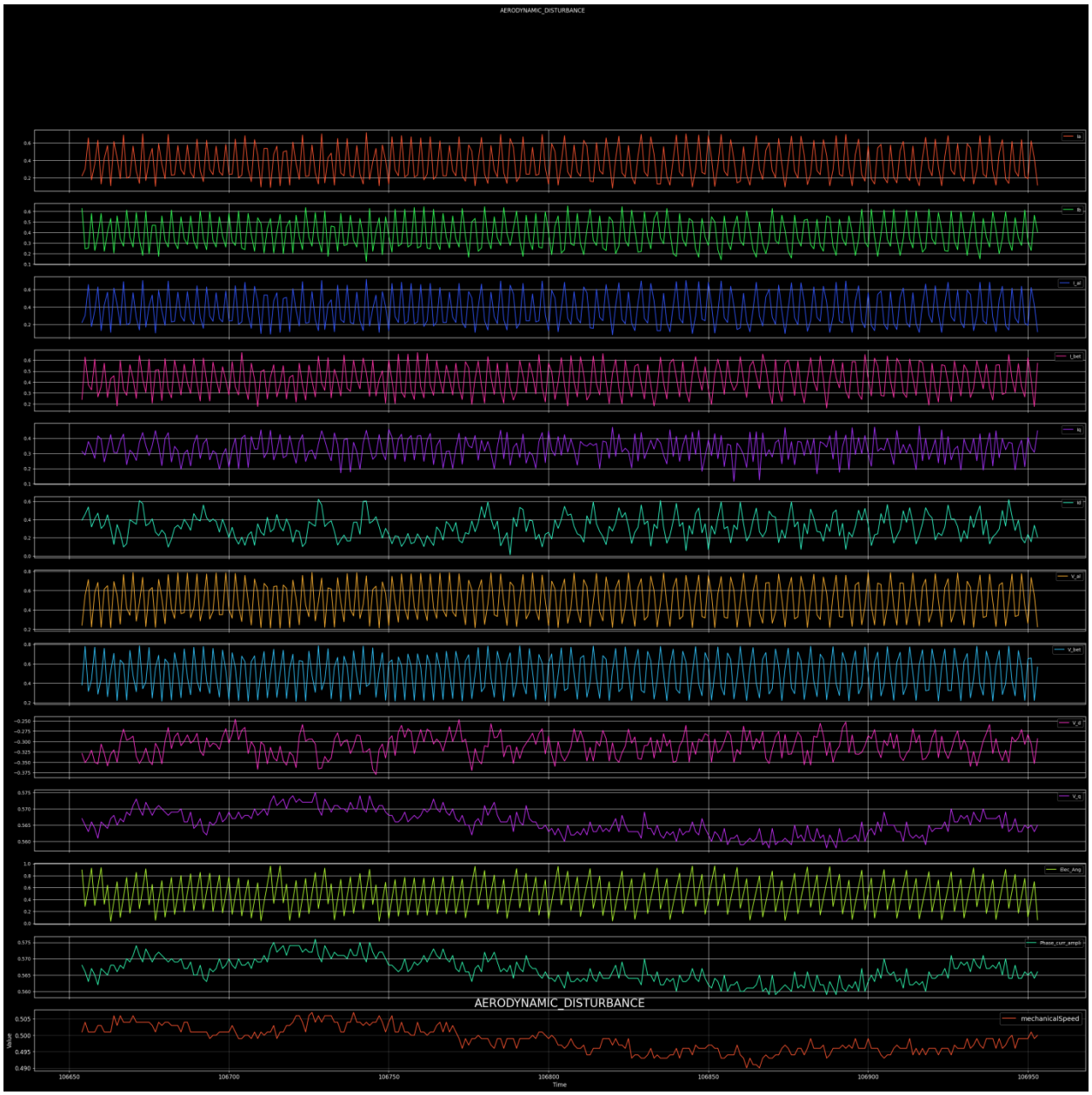


Fig. 6.17 Aerodynamic induced anomaly waveform

6.5.3 Acquiring Data for Uncontrollability Anomaly

Figure 6.18 illustrates the waveforms of the electrical signatures during the uncontrollability anomaly effect.

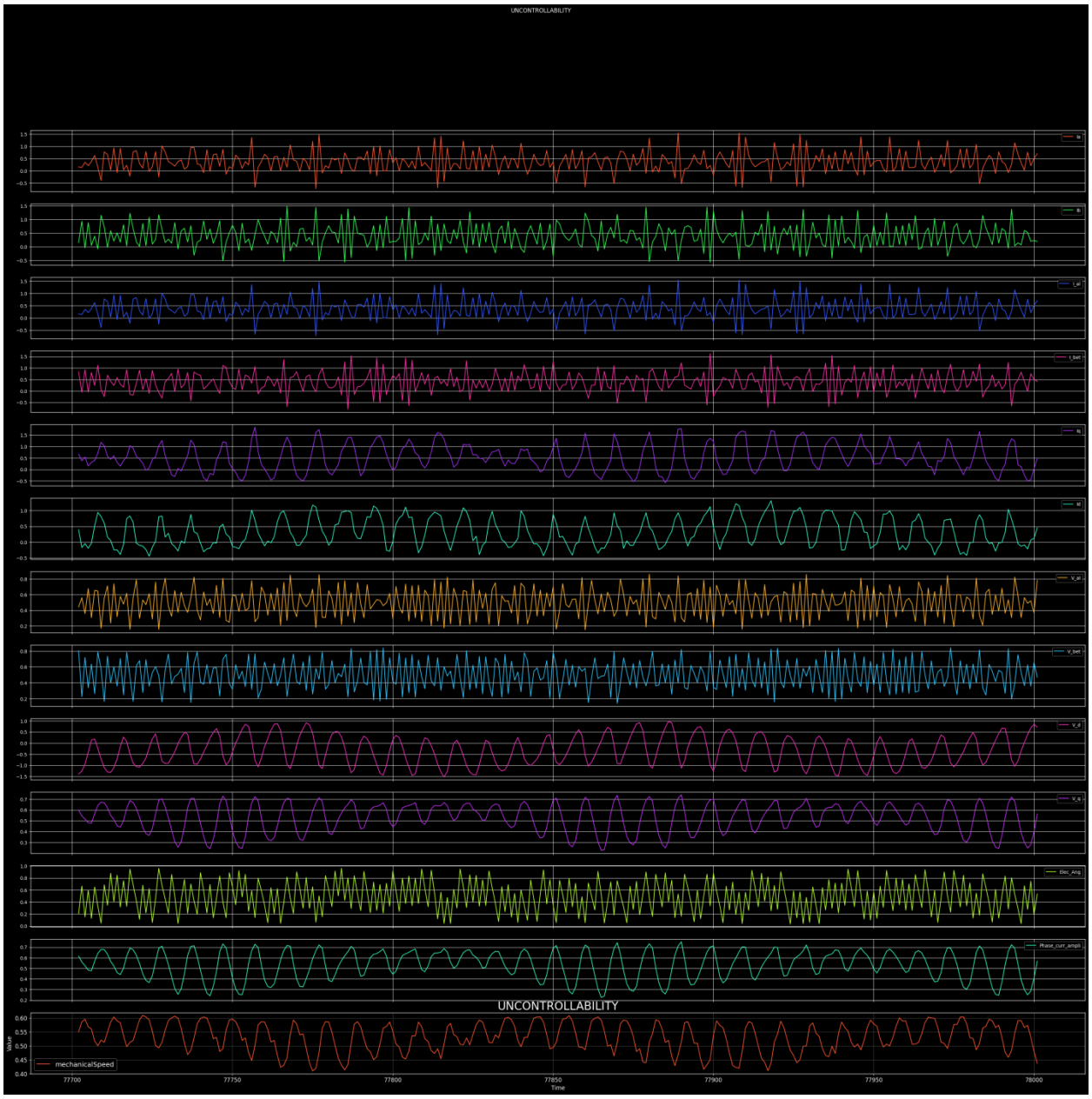


Fig. 6.18 Uncontrollability induced anomaly waveform

6.5.4 Acquiring Data for Asymmetric Broken Propeller Anomaly

Figure 6.19 illustrates the waveforms of the electrical signatures during the broken propeller anomaly effect.

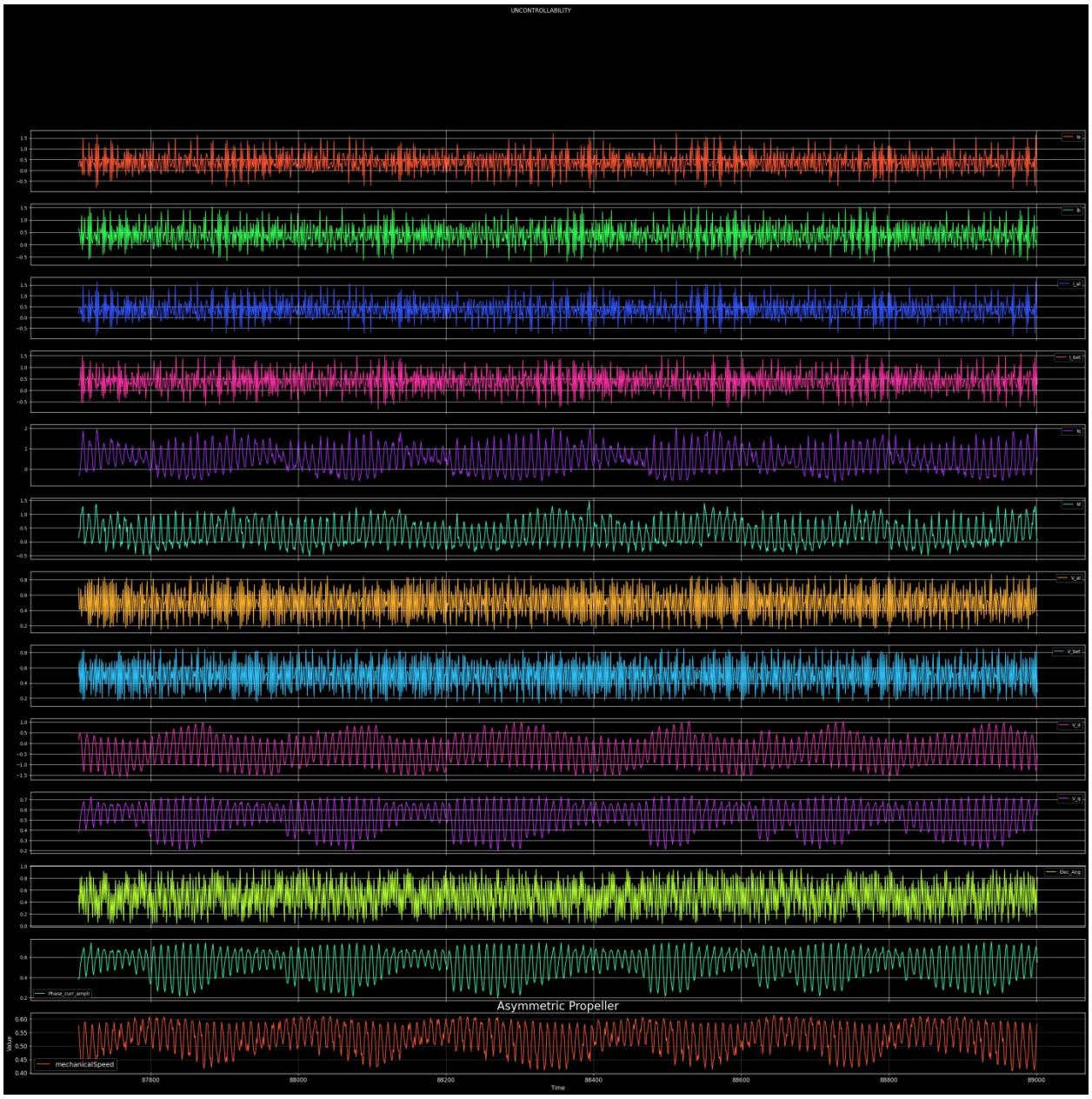


Fig. 6.19 Asymmetric propeller / broken propeller induced anomaly

6.6 Neural Network Architecture for Anomaly Detection

In the previous section, we observed the waveforms associated with various anomalies, determining whether their distinctive patterns were visually identifiable. However, even if these patterns weren't immediately discernible, advanced neural algorithms possess the capability to extract nuanced features from the waveforms and precisely classify each anomaly. What surpasses Convolutional Neural Networks in mimicking the capabilities of the visual cortex? With this question in mind, we engineered a CNN algorithm that not only effectively classifies anomalies but is also optimized for execution on edge devices. In this section, we will delve into the architecture of the CNN developed. Refer to Fig [] for the CNN architecture.

6.6.1 Input Layer:

In the CNN architecture, input data is sourced from the previously discussed databuffer. The input layer serves as the entry point for this raw data into the neural network. It is structured as a 2D array with dimensions (Number_Of_Rows=50, Number_Of_Columns=13), representing the spatial dimensions of the input signal, with a single channel (e.g., grayscale).

The number of rows, 50, denotes the number of samples considered from successive timestamps, while the number of columns signifies the features, i.e., the electrical signatures.

6.6.2 Feature Extractor:

The feature extractor is the core of the network, responsible for learning meaningful representations from the input data. Each convolutional layer (Conv2D) applies a set of learnable filters to the input data, convolving them across the input spatial dimensions to extract spatial patterns. Batch normalization normalizes the activations of each layer, which helps with faster convergence during training and regularization. Here, ReLU activation introduces non-linearity, enabling the network to learn complex relationships within the data.

Table. 6.1 CNN architecture for Motor Anomaly Detection

Layer (type)	Output Shape	Param #	Mathematical Formula
inputLayer (InputLayer)	(None, 50, 13, 1)	0	Input Data
conv2d	(None, 49, 12, 24)	120	$H_{\text{out}} = \frac{(H_{\text{in}} - F + 2\times P)}{S} + 1$
batch_normalization	(None, 49, 12, 24)	96	$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ $x' = \frac{x-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}$ $y = \gamma x' + \beta$
activation	(None, 49, 12, 24)	0	$\text{ReLU}(x) = \max(0, x)$
max_pooling2d	(None, 46, 11, 24)	0	$H_{\text{out}} = \frac{(H_{\text{in}} - F)}{S} + 1$
conv2d_1	(None, 45, 10, 16)	1552	$H_{\text{out}} = \frac{(H_{\text{in}} - F + 2\times P)}{S} + 1$
batch_normalization_1	(None, 45, 10, 16)	64	Same as above
activation_1	(None, 45, 10, 16)	0	$\text{ReLU}(x) = \max(0, x)$
max_pooling2d_1	(None, 42, 9, 16)	0	$H_{\text{out}} = \frac{(H_{\text{in}} - F)}{S} + 1$
conv2d_2	(None, 41, 8, 8)	520	$H_{\text{out}} = \frac{(H_{\text{in}} - F + 2\times P)}{S} + 1$
batch_normalization_2	(None, 41, 8, 8)	32	Same as above
activation_2	(None, 41, 8, 8)	0	$\text{ReLU}(x) = \max(0, x)$
max_pooling2d_2	(None, 38, 7, 8)	0	$H_{\text{out}} = \frac{(H_{\text{in}} - F)}{S} + 1$
conv2d_3	(None, 37, 6, 4)	132	$H_{\text{out}} = \frac{(H_{\text{in}} - F + 2\times P)}{S} + 1$
batch_normalization_3	(None, 37, 6, 4)	16	Same as above
activation_3	(None, 37, 6, 4)	0	$\text{ReLU}(x) = \max(0, x)$
max_pooling2d_3	(None, 34, 5, 4)	0	$H_{\text{out}} = \frac{(H_{\text{in}} - F)}{S} + 1$
flatten	(None, 680)	0	Reshaping as a vector
dense	(None, 12)	8172	$Z = WX + b$
dense_1	(None, 6)	78	$Z = WX + b$
dense_2	(None, 4)	28	$Z = WX + b$
Total params: 10810 (42.23 KB)		Trainable params: 10706 (41.82 KB)	
Non-trainable params: 104 (416.00 Byte)			

Table. 6.2 Mathematical Operations and Variables Explained

Layer Type	Mathematical Operation and Variables
Convolution	Output shape: $H_{\text{out}} = \frac{(H_{\text{in}} - F + 2 \times P)}{S} + 1$ Where: H_{out} = Output height, H_{in} = Input height, F = Filter size, P = Padding size, S = Stride length
Batch Normalization	$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i$ $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$ $x' = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ $y = \gamma x' + \beta$ Where: x = Input, m = Batch size, μ_B = Batch mean, σ_B^2 = Batch variance, ϵ = Small constant (for numerical stability), γ = Scale parameter, β = Shift parameter
ReLU Activation	$\text{ReLU}(x) = \max(0, x)$ Where: x = Input
Dense (Fully Connected)	$Z = WX + b$ Where: Z = Output, W = Weight matrix, X = Input matrix, b = Bias vector

6.6.3 Max Pooling Layer:

Max-pooling reduces the spatial dimensions of the feature maps while retaining the most salient features, aiding in translation invariance and reducing computational complexity.

6.6.4 Flatten Layer:

The output of the convolutional layers is flattened into a one-dimensional array, collapsing the spatial dimensions into a single vector. This prepares the data for processing by the fully connected layers, which require one-dimensional input.

6.6.5 Dense / Fully Connected Layers:

Fully connected layers process the flattened feature vector, performing high-level reasoning and decision-making based on the extracted features.

These layers learn complex combinations of features and their relationships across the entire input space. The number of neurons in these layers determines the capacity of the network to capture and represent different patterns in the data.

6.6.6 Rectified Linear Unit:

ReLU activation is commonly used in fully connected layers to introduce non-linearity.

6.6.7 Output Layer:

The output layer is typically a dense layer with a softmax activation function, which outputs a probability distribution over the different classes.

In motor signal anomaly detection, the number of classes (Number_Of_Classes) represents different types of anomalies or normal conditions. Here, Number of classes are HEALTHY, AERODYNAMIC DISTURBANCE, UNCONTROLLABILITY AND ASYMMETRIC LOAD.

6.6.8 Softmax Activation Function

The softmax function ensures that the output probabilities sum up to 1, making it interpretable as class probabilities.

6.7 Training the Neural Net for Anomaly Classification / Motor Health Estimation / Diagnosis

6.7.1 Training Methodology

For training the CNN, we utilize the capabilities of TensorFlow and Keras, two powerful tools for building and training neural network models. TensorFlow, developed by Google Brain, offers scalability, flexibility, and high performance, making it ideal for a variety of machine learning tasks. Its comprehensive ecosystem includes support for distributed computing, hardware acceleration, and extensive documentation. Keras, a high-level neural network API, provides a user-friendly interface for model development, allowing for rapid prototyping and experimentation. With its emphasis on ease of use, modularity, and extensibility, Keras seamlessly integrates with TensorFlow, enabling developers to leverage the best of both worlds. Together, TensorFlow and Keras form a robust framework for training neural networks, empowering users to tackle challenging tasks such as anomaly classification, motor health estimation, and diagnosis with confidence and efficiency.

6.7.2 Defining Callbacks

Callbacks are functions that are executed during training at specific stages, allowing for dynamic adjustments and interventions. In the code snippet, several callbacks are defined:

- **Reduce Learning Rate on Plateau**: This callback dynamically adjusts the learning rate when the validation loss stops improving, ensuring smoother convergence during training.

- **Early Stopping**: This callback halts the training process early if the validation loss fails to improve over a certain number of epochs, preventing overfitting and conserving computational resources.

- **Model Checkpoint**: This callback saves the model's weights at specified intervals, typically based on a chosen metric such as validation loss. This ensures that the best-performing model weights are preserved throughout the training process.

6.7.3 Compiling the Model

After importing dependencies and defining callbacks, the next step is compiling the neural network model. Compilation involves configuring the model with essential components such as the loss function, optimizer, and evaluation metrics. In the provided code, the model is compiled with the following configurations:

- **Loss Function**: Categorical cross-entropy loss is used, which is suitable for multi-class classification tasks. It quantifies the discrepancy between the model's predictions and the actual target values during training.

- **Optimizer**: The Adam optimizer is chosen for its efficiency and effectiveness in optimizing neural network models. It adapts the learning rate during training, making it well-suited for a wide range of applications.

- **Metrics**: The model is evaluated using the accuracy metric, which measures the proportion of correctly classified samples. Additionally, other metrics such as precision, recall, and F1-score can be used to assess the model's performance comprehensively.

6.7.4 Training the Model

Once the model is compiled, it is ready to be trained using the available training data. The training process involves presenting batches of training samples to the model, adjusting its weights through backpropagation, and optimizing the chosen objective function. Key parameters such as the number of epochs, batch size, and callbacks are specified during the training phase:

- Epochs: The model is trained for a specified number of epochs, with each epoch representing one complete pass through the entire training dataset. Training over multiple epochs allows the model to learn from the data iteratively.

- Batch Size: During each epoch, the training data is divided into batches, with the model updating its weights after processing each batch. The batch size determines the number of samples processed before the model's parameters are updated.

- Callbacks: Callback functions such as ReduceLROnPlateau, EarlyStopping, and ModelCheckpoint are utilized to monitor the training process and make dynamic adjustments as needed. These callbacks help improve training efficiency and prevent overfitting.

6.7.5 Training Hardware

The CNN algorithm was trained on a Nvidia Tesla T4 GPU, made available through Google Colab. The Tesla T4 is a GPU card built on the Turing architecture, specifically optimized for accelerating deep learning model inference.

Our Tesla T4 card boasts 40 streaming multiprocessors (SM) and a 6MB L2 cache shared among all SMs. Additionally, it features 16GB of high-bandwidth memory (GDDR6) directly connected to the processor. The overall architecture is depicted in Figure 6.20.

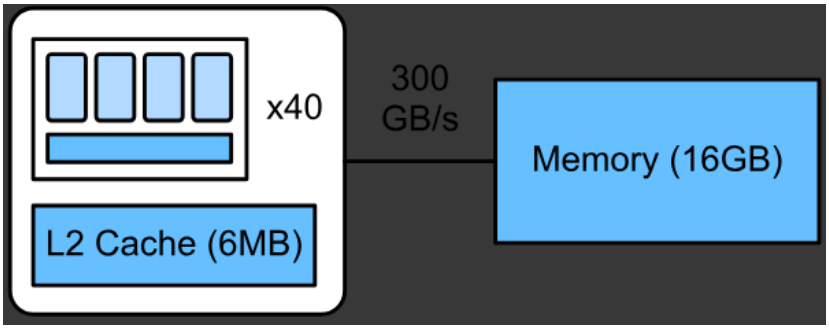


Fig. 6.20 Tesla T4 GPU architecture

The NVIDIA System Management Interface (nvidia-smi) is a command-line utility built upon the NVIDIA Management Library (NVML), designed to facilitate the management and monitoring of NVIDIA GPU devices.

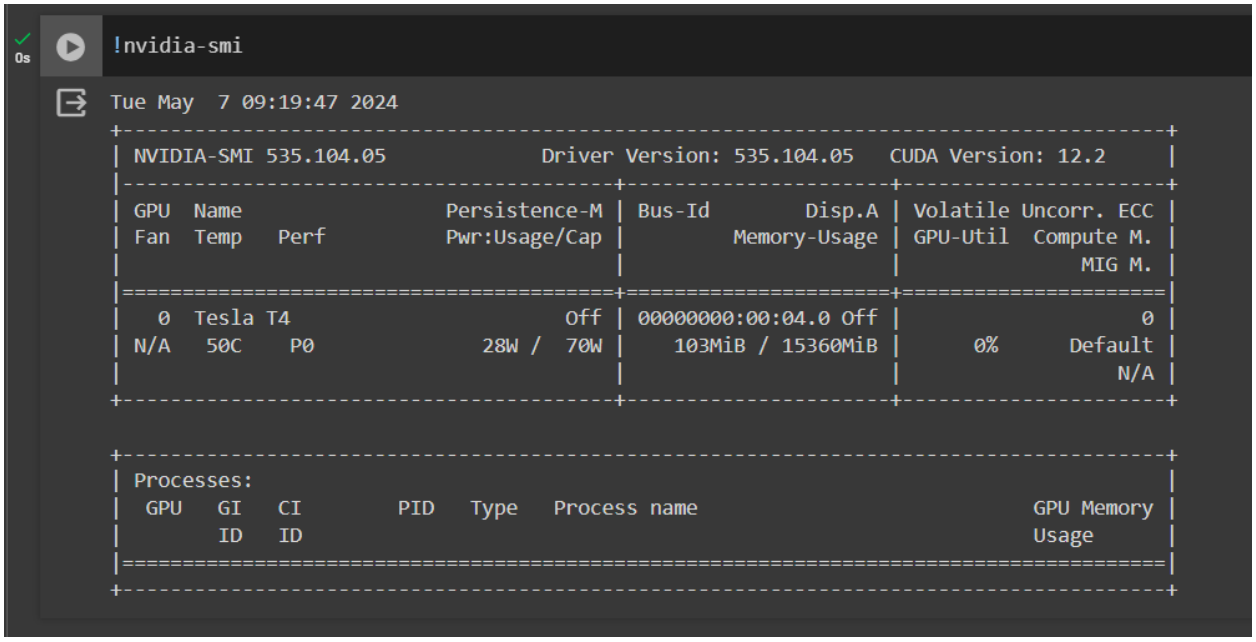


Fig. 6.21 Training hardware details using !nvidia-smi command

This utility empowers administrators to inquire about GPU device status and, with appropriate privileges, to alter GPU device state. Initially focused on the TeslaTM, GRIDTM, QuadroTM, and Titan X products, it also offers limited support for other NVIDIA GPUs.

6.7.6 Training Metrics – Training Loss and Accuracy

The following figures illustrate the training and validation loss, as well as accuracy graphs.

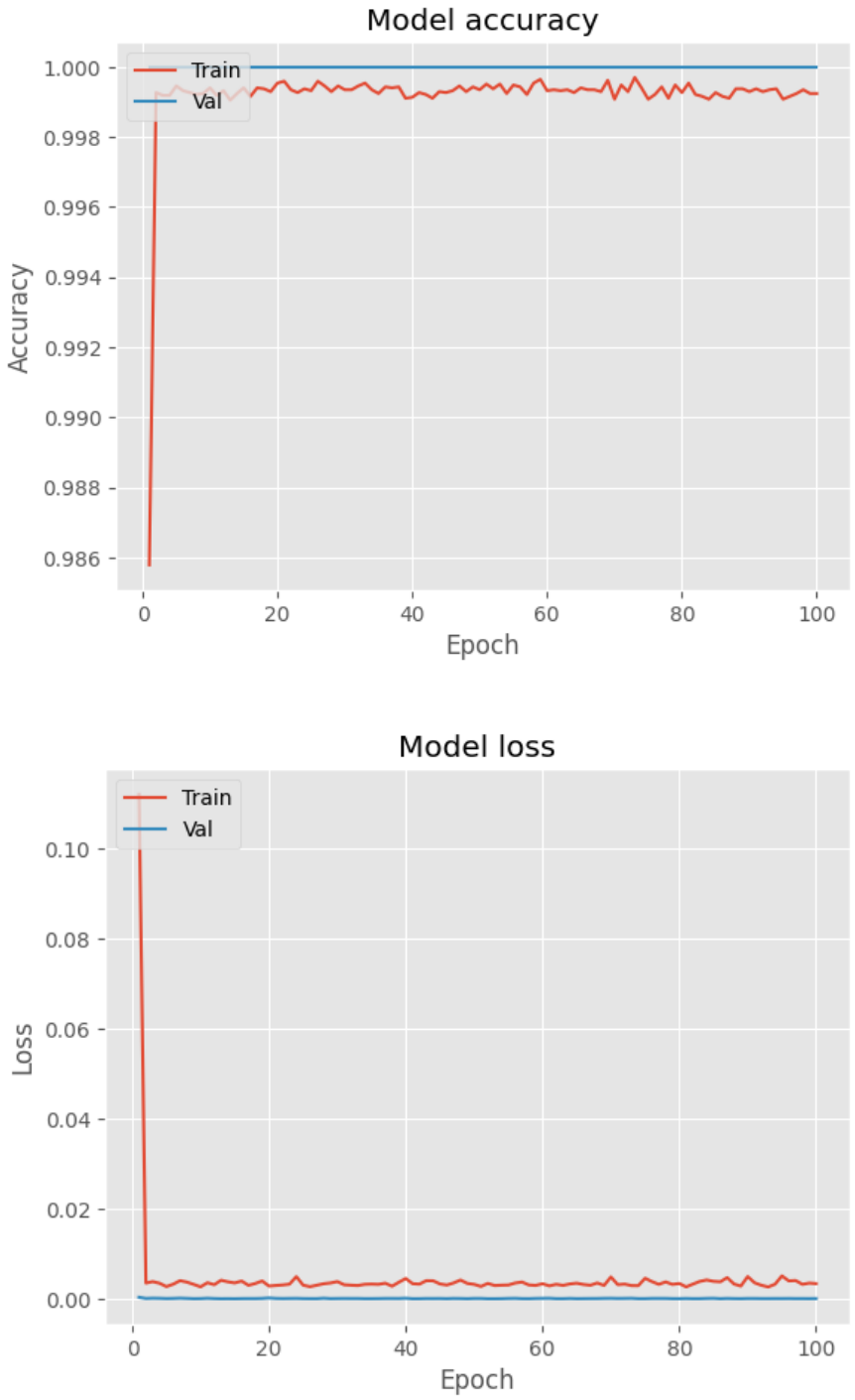


Fig. 6.22 Training Accuracy and Loss Waveform respectively

6.8 Results – CNN Model Performance Validation

In this section, we'll explore the anomaly classification results, employing form validation metrics to evaluate the performance.

6.8.1 Classification Metrics

- **TP (True Positives)**: Number of correct positive predictions.
- **FN (False Negatives)**: Number of actual positive instances incorrectly predicted as negative.
- **FP (False Positives)**: Number of actual negative instances incorrectly predicted as positive.
- **TN (True Negatives)**: Number of correct negative predictions.

- **Accuracy**: Accuracy measures the proportion of correctly classified instances out of the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: Precision measures the proportion of correctly identified positive cases out of all cases predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity)**: Recall measures the proportion of correctly identified positive cases out of all actual positive cases.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**: F1 Score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Matthews Correlation Coefficient (MCC)**: MCC takes into account true and false positives and negatives and is generally regarded as a balanced measure even if the classes are of very different sizes.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

- **Confusion Matrix**

The confusion matrix is a table used to evaluate the performance of a classification model. It compares the actual values of the target variable with the values predicted by the model.

Table. 6.3 Confusion Matrix

		Predicted	
		Positive	Negative
2∗Actual	Positive	TP	FN
	Negative	FP	TN

- **ROC Curve**

The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It shows the performance of a classification model at all classification thresholds.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- **Precision-Recall Curve**

The Precision-Recall curve is a graphical plot that shows the trade-off between precision and recall for different threshold values. It is particularly useful when dealing with imbalanced datasets, where one class is more frequent than the other.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

After validating the CNN architecture across various anomaly classes, Figure 6.23 illustrates the scores of different classification metrics. The results signify the neural net classified every anomaly correctly.

Metric	Score
Accuracy	1
Precision	1
Recall	1
F1 Score	1
Matthews Corr. Coeff. (MCC)	1

Fig. 6.23 Validation metrics results

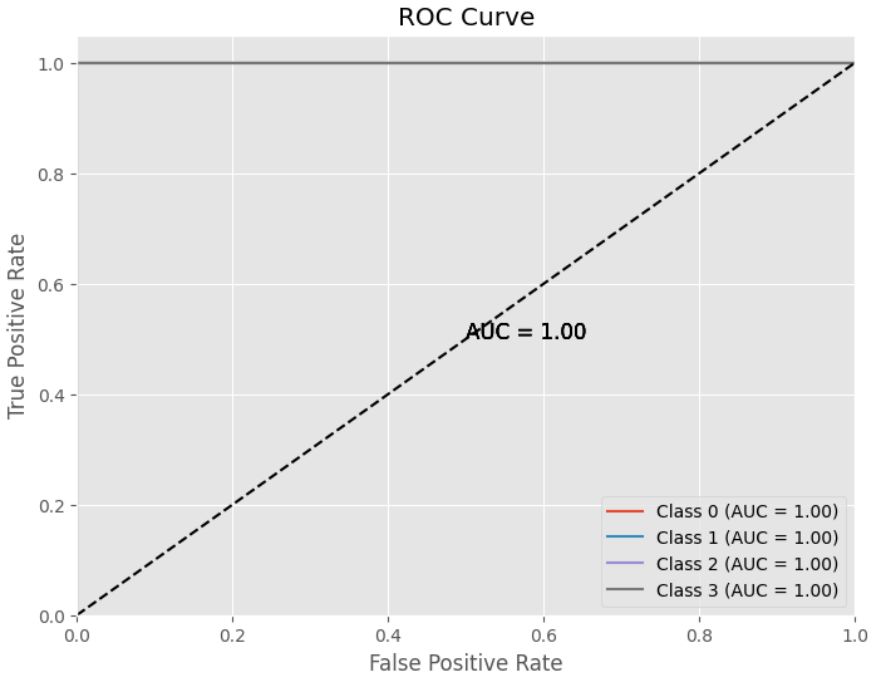


Fig. 6.24 ROC curve as validation metric

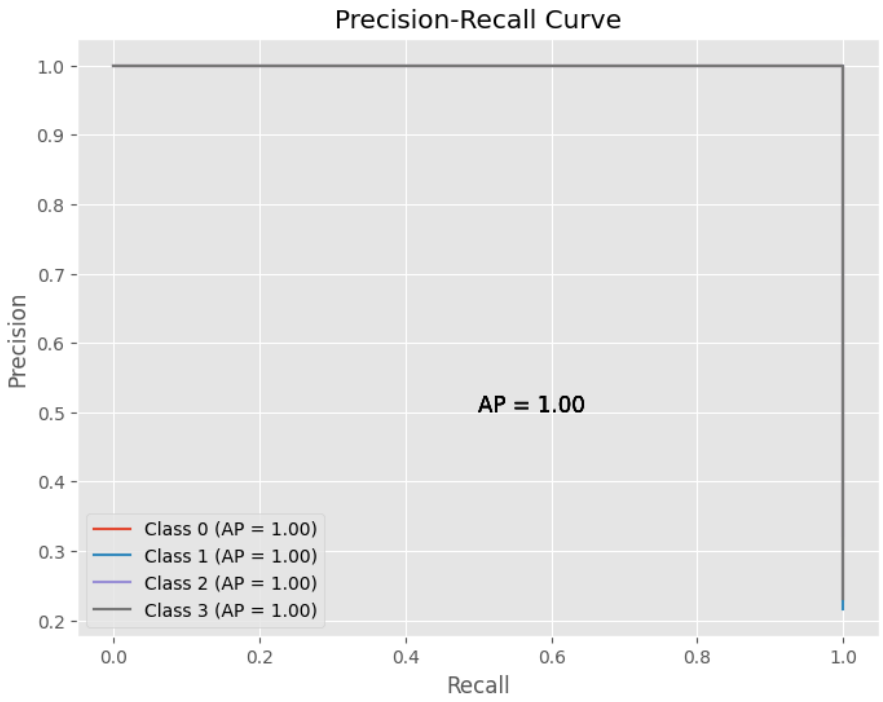


Fig. 6.25 Precision-Recall Curve as validation metric

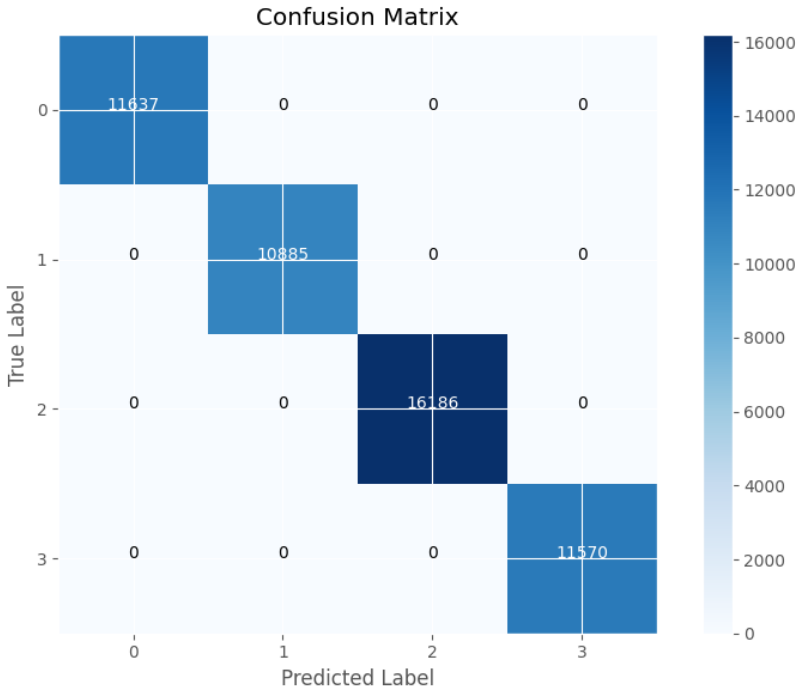


Fig. 6.26 Confusion Matrix as validation metric

Here, the 0-3 represents the anomaly classes respectively.

The results demonstrate that the CNN algorithm not only achieved perfect classification but also showcased its robustness for applications in power electronics and motor control. Next, we will assess the neural net's performance as a real-time inference system at the edge.

6.9 Realization of the Trained Neural Net on the Edge Device

In this section we realized the trained neural net into the edge device to perform the inference in real-time. We used STM32H755ZI-Q to acquire data from PTOOL3A via Inter Integrated Protocol in real-time motor operation and concurrently feed the pre-trained convolution neural net with real-time data for pattern recognition / anomaly detection.

6.9.1 Model Compression – Tensorflow Quantization

Edge devices often grapple with limited memory and computational resources, necessitating various optimizations to ensure models can operate within these constraints. Moreover, certain optimizations enable the utilization of specialized hardware for accelerated inference.

TensorFlow Lite, along with the TensorFlow Model Optimization Toolkit, furnishes tools tailored to streamline the complexity of optimizing inference. These tools offer several avenues through which model optimization can bolster application development.

Size Reduction:

Optimization techniques can trim the size of a model, offering several advantages:

- **Reduced Storage Size**: Smaller models occupy less space on users' devices, translating to more storage availability. For instance, an Android app utilizing a compact model consumes less storage on mobile devices.
- **Decreased Download Size**: Compact models require less bandwidth and time to download onto users' devices.
- **Lower Memory Usage**: Smaller models demand less RAM during execution, freeing up memory for other application components and potentially enhancing overall performance and stability. Techniques such as quantization, pruning, and clustering contribute to size reduction while balancing accuracy considerations.

Latency Reduction:

Latency denotes the time taken to execute a single inference with a given model. Certain optimization techniques curtail the computational overhead associated with inference, leading to reduced latency. Lower latency can also positively impact power consumption. Presently, quantization stands out as a method to mitigate latency by streamlining inference calculations, albeit with a potential trade-off in accuracy.

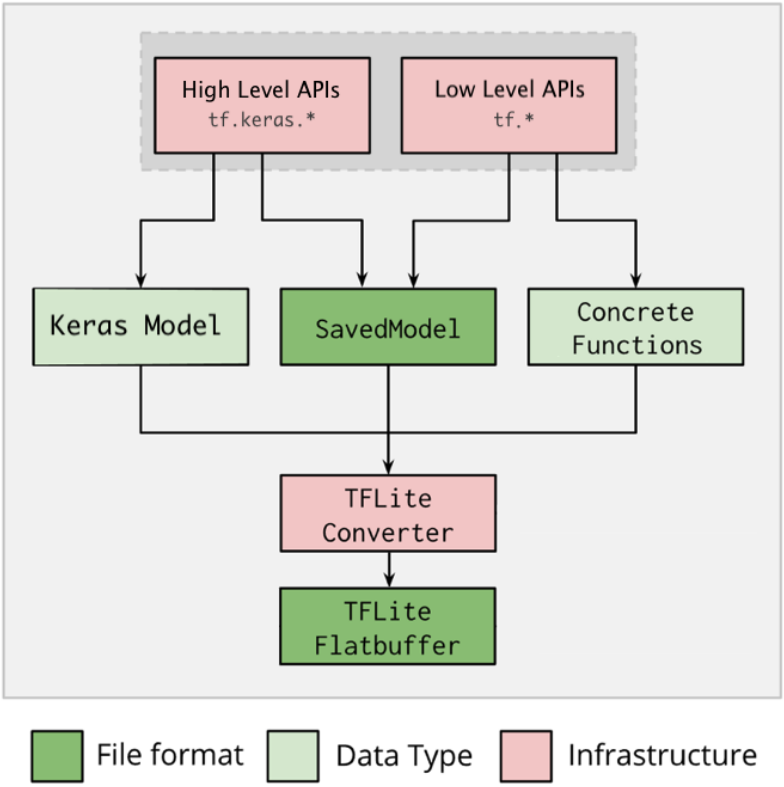


Fig. 6.27 Tensorflow to Tensorflow Lite Conversion [2]

6.9.2 Optimized C-kernel Conversion – STM32 AI Tools

The Arm® Cortex®-M instructions and data manipulations, like packing and unpacking during memory transfers, pose challenges for efficient implementation across all data type combinations. The stm32ai application prioritizes optimizations to enhance memory usage (flash memory, RAM, or both) and reduce latency (execution time), rather than focusing solely on size optimization. Consequently, the code generator considers only specific data types—namely, 32-bit float, 8-bit signed, and binary signed (1-bit)—to deploy optimized C-

kernels, as detailed in the upcoming "Optimized C-kernels" section.

In scenarios where these specific data types are not applicable, the code generator resorts to using a fallback strategy, employing a 32-bit floating-point C-kernel with pre- or post-quantization or dequantization operations whenever feasible. This approach ensures compatibility and performance across various data types while adhering to memory and latency optimization goals.

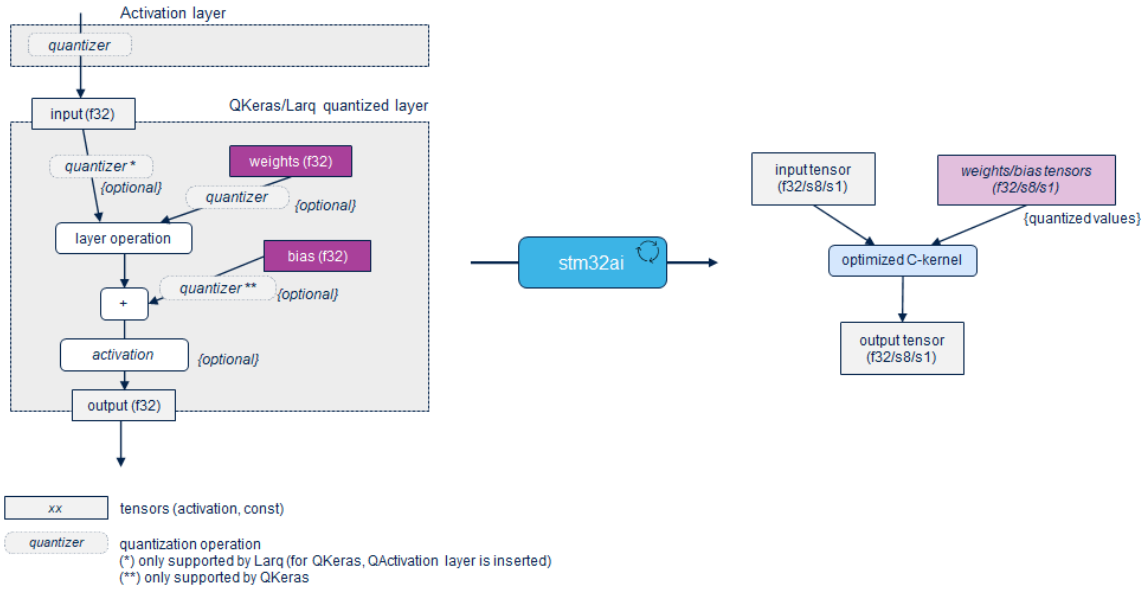


Fig. 6.28 Representation from Quantization layer to Deployed Layer [3]

6.9.3 Model metrics on the Edge Device

We utilized STM32AI tools to implement the pre-trained CNN model for the STM32H755 microcontroller architecture. Figure 6.28 illustrates the memory disparity between the pre-compressed and compressed network files. The Keras non-compressed file, named "seNNsEdge_Model_H5.h5," has a size of 234.31 KB. Meanwhile, the TensorFlow Lite compressed file, named "seNNsEdge_Model_tflite.tflite," has a size of 19.17 KB. Consequently, we observe a compression percentage of 91.822%.

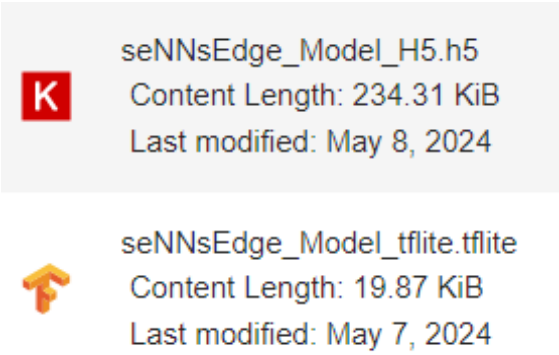


Fig. 6.29 Figure displays the Model size – Content Length

The tflite model was converted to C-kernel by STM32Cube-AI for real-time execution in the microcontroller. The memory usage pertaining to neural network parameters and operations is depicted in Figure 6.29, generated by the STM32AI tool.

Table. 6.4 Memory consumption for CNN C-kernel

Model Parameters	10,596 Params (12.98 KiB)
input	2.54 KBytes
output	16 Bytes
macc	1,145,208
weights	42,384 B (41.39 KiB)
activations	59,048 B (57.66 KiB)
ram (total)	59,048 B (57.66 KiB)

The Total Multiply and Accumulate Complexity (MACC), as measured by the tool, amounts to 1,145,208 MACC. In real-time inference tests for anomaly detection within our setup, the system exhibited remarkable detection capabilities, achieving an impressively low latency of 6.9 milliseconds per decision iteration.

6.10 Conclusion

In conclusion, the chapter details the rigorous process of training a Convolutional Neural Network (CNN) for anomaly classification, motor health estimation, and diagnosis, utilizing TensorFlow and Keras. Various training methodologies including defining callbacks, compiling the model, and executing the training process are explained in depth. The hardware setup, featuring a Nvidia Tesla T4 GPU, is described alongside training metrics such as loss and accuracy. The CNN model's performance validation is comprehensively assessed through classification metrics, confusion matrix, ROC curve, and precision-recall curve, demonstrating impeccable classification across anomaly classes.

Furthermore, the trained neural network is successfully deployed onto an edge device, specifically the STM32H755 microcontroller architecture, for real-time inference. Model compression techniques, leveraging TensorFlow Lite and STM32AI tools, significantly reduce model size while maintaining high performance. Optimized C-kernel conversion enhances memory usage and reduces latency, ensuring efficient deployment on resource-constrained devices. Real-time inference tests showcase the system's exceptional anomaly detection capabilities, with low latency of 6.9 milliseconds per decision iteration, affirming the robustness and effectiveness of the trained neural network in practical applications.

6.11 References

[1] *STDES-PTOOL3A - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/evaluation-tools/stdes-ptool3a.html#

[2] *Introduction to TensorFlow Lite – Study Machine Learning*. (n.d.). https://studymachinelearning.com/introduction-to-tensorflow-lite/

[3] *File:DQNN 1 QuantizeToOperator.png - stm32mcu*. (n.d.). https://wiki.st.com/stm32mcu/wiki/File:DQNN_1_QuantizeToOperator.png

Chapter 7

Expanding the Horizons of Edge Computing: Stacked Convolutional Autoencoder for 3-phase Shunt Current Measurement Denoising

7.1 Introduction

In this section, we delve into an innovative application of edge computing, expanding its scope beyond anomaly detection. In the domain of Sensorless Motor Control and Field-Oriented Control (FOC), achieving precise measurement of 3-phase currents is fundamental. These currents play a critical role in Sensorless Vector Control of Motors and are typically assessed using Shunt Resistors and Op-Amps before being processed through Analog-to-Digital Converters (ADCs), as illustrated in Figure 1. The accuracy of ADC measurements is pivotal for the effectiveness of control algorithms [1].

Traditionally, a combination of hardware and software filters has been utilized to attenuate noise in feedback shunt current signals [2]. However, in this chapter, we present an innovative methodology employing a neural approach to denoise shunt currents. Specifically, we harness the power of Stacked Convolutional Neural Networks (CNNs) to construct a Denoising Autoencoder [3] framework named, Convolutional Autoencoder. This approach not only diminishes the need for hardware filters but also capitalizes on the robustness and fault tolerance characteristics inherent in neural networks.

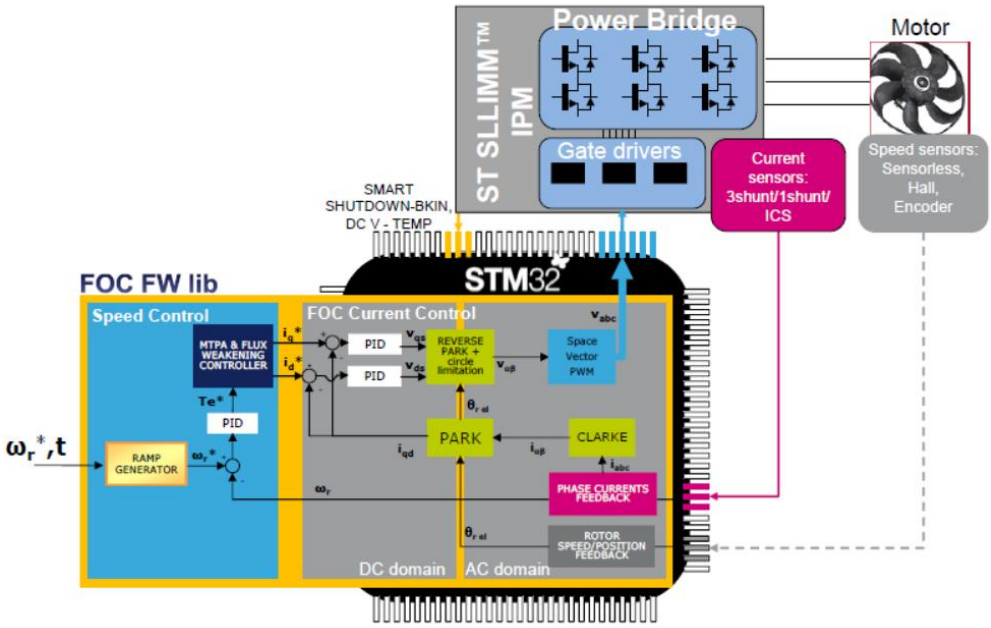


Fig. 7.1 Field Oriented Control of PMSM/BLDM Motor Control Loop [4]

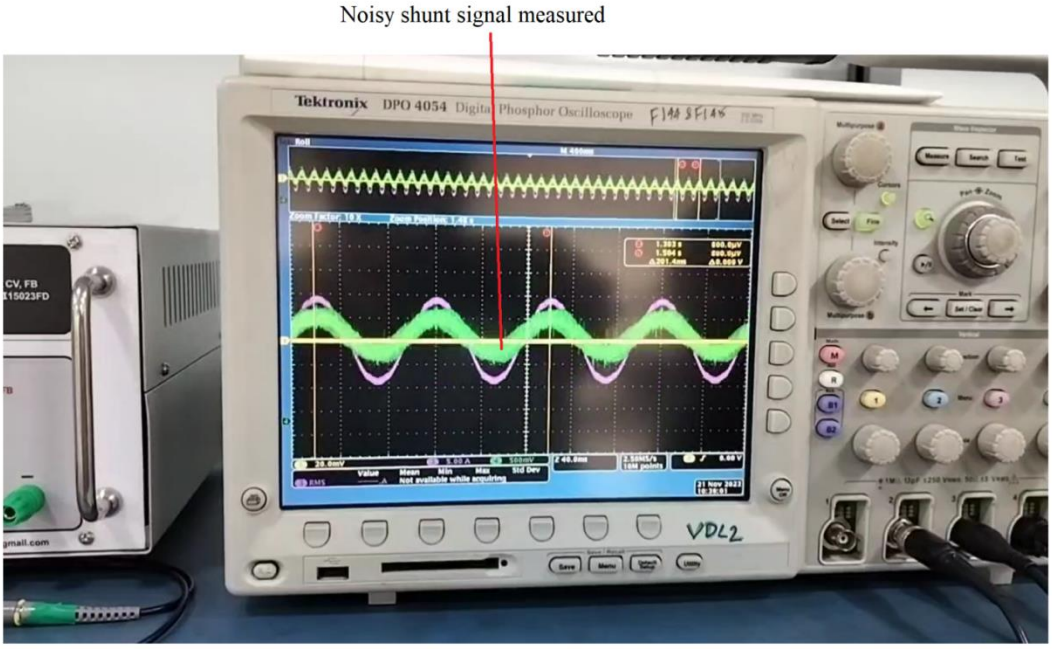


Fig. 7.2 Green colored waveform displays the noisy shunt current which is eventually fed into ADC

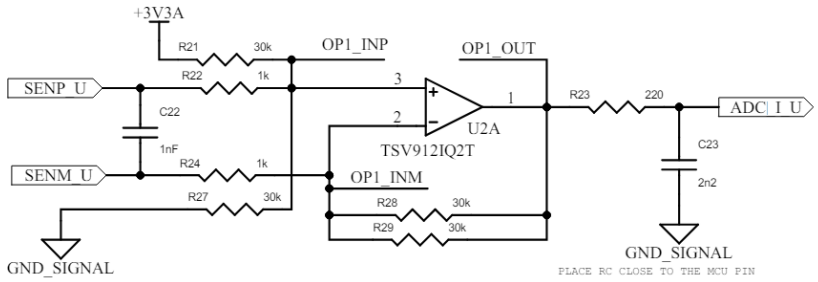


Fig. 7.3 ADC Sensing Circuit in PMSM/BLDM Motor Controller – STEVAL PTOOL4A [5]

7.2 Introduction to Auto Associative Neural Networks or Autoencoders

The primary objective of deep learning is to discern meaningful patterns within data that are pertinent to specific application tasks. Among the well-established neural algorithms, the autoencoder, also known as an auto-associative neural network, stands out. The architecture of an autoencoder typically comprises an equal number of input (x) and output (y) layers, where the network is trained to produce an output closely resembling the input. Fundamentally, an autoencoder consists of an Encoder-Decoder pair: the encoder maps the input into a latent vector or hidden representation, denoted as z(x), while the decoder maps the latent vector back into the output, represented as y(z). However, the purpose of the network transcends mere input-output mapping; rather, it incorporates constraints aimed at generating non-trivial solutions. Various methodologies exist for imposing constraints tailored to specific applications. For instance, constraints can be formulated to restrict the latent vector z relative to the input x or enforce sparsity in z, resulting in sparse encoding.

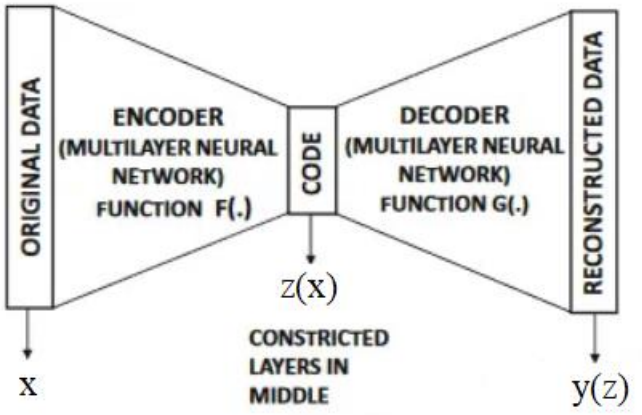


Fig. 7.4 Autoencoder graphical representation

This implies that high-dimensional signals exhibit a low-dimensional structure. Alternatively, by adapting and refining the training procedure, the network can be coerced into generating non-trivial solutions, as exemplified by applications such as signal reconstruction or denoising, which will be elucidated in the subsequent section.

7.3 Denoising Autoencoder Algorithm

We understand the crucial significance of limiting the dimensionality of the latent vector to prevent the model to replicate the input to output by learning identity mapping. The concept of Denoising Autoencoders [6] forces the neural net architecture to learn to de-corrupt the data yielding the desired denoised data. The notion is to corrupt the input vector x_n modifying the vector as x` and emit the reconstructed output y(x`, w), where the network learns denoising ability upon training and reducing the error function such as sum-of-squares:

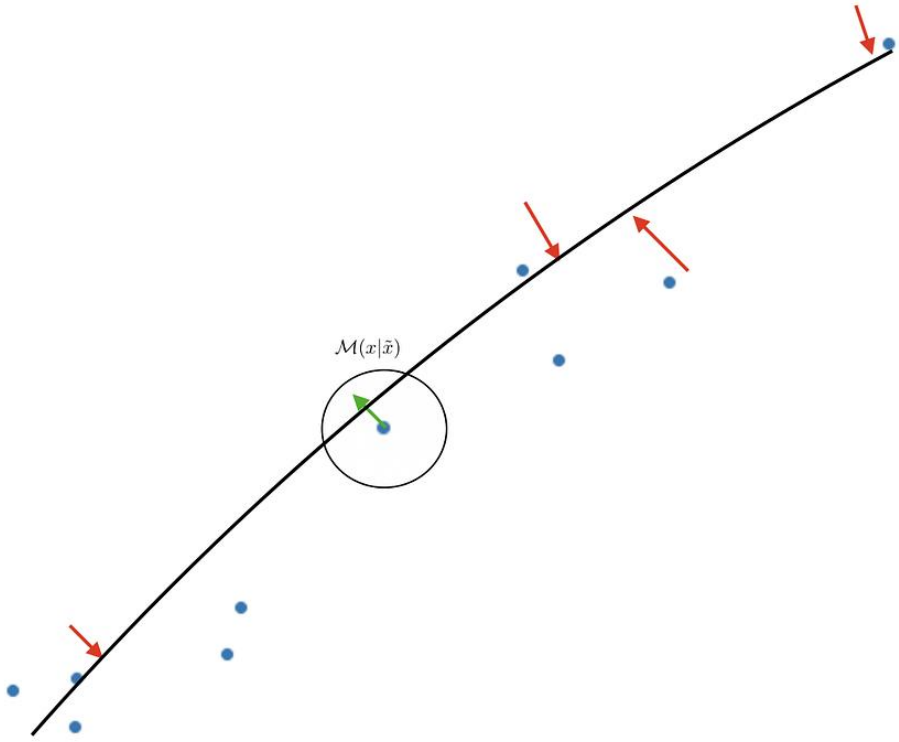


Fig. 7.5 Denoising Autoencoder - I. Goodfellow, Y. Bengio, A. Courville [7]

7.4 Experiment and Results – Training a Convolution Neural Network based Stacked Autoencoder to Denoise Shunt Sensing like Simulation Data

Our aim is to train a Stacked Convolutional Autoencoder (Convolutional Autoencoder) model to remove noise from Permanent Magnet Synchronous Motor (PMSM) or Brushless DC (BLDC) motor shunt currents. This task utilizes a synthetic dataset generated through simulation, comprising both noise-free and noisy signals. The noisy signal encompasses a diverse spectrum of noise categories, reflecting our aspiration to develop a model with robust denoising capabilities.

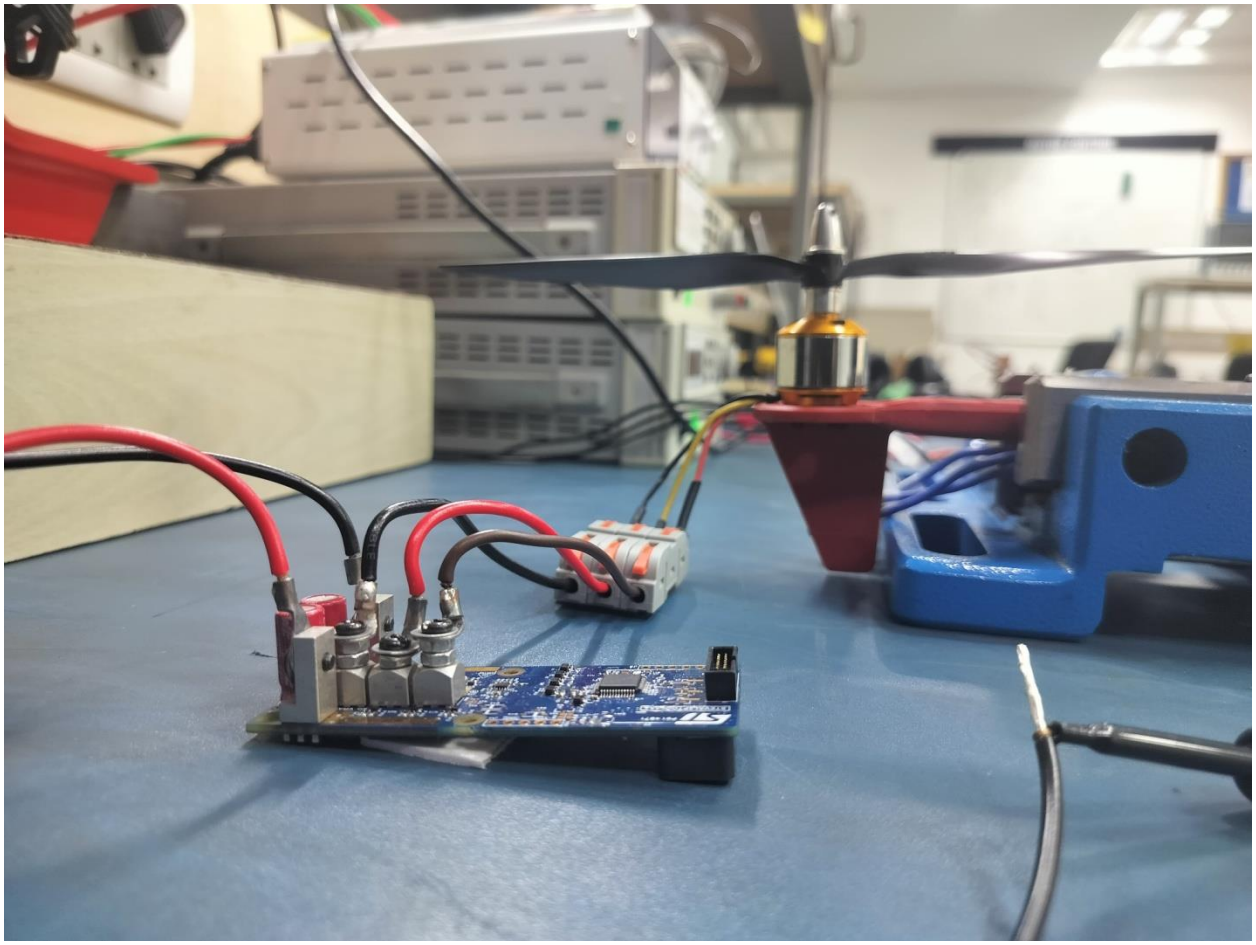


Fig. 7.6 PMSM Motor driven by STEVAL-PTOOL4A Motor Driver Board

The synthetic simulation dataset consists of a 50Hz Pure Signal resembling Shunt Current and its corresponding Noisy version. Gaussian noise has been employed to introduce noise into the signal. Mathematically, Gaussian Noise is represented by:

$$f(x) = \left[\frac{1}{\sigma \times \sqrt{2\pi}}\right] \times e^{\left[-\frac{(x-\mu)^2}{2\times\sigma^2}\right]} \quad (1)$$

$x = value\ at\ which\ the\ Gaussian\ noise\ is\ being\ evaluated$

$\sigma^2 = variance\ \left(the\ square\ of\ the\ standard\ deviation\right)\ of\ the\ distribution$

$\mu = mean\ \left(average\right)\ value\ of\ the\ distribution$

$e = base\ of\ the\ natural\ logarithm,\ approximately\ 2.71828$

We have generated simulation data for pure and noisy signal as shown in Fig. 7.7

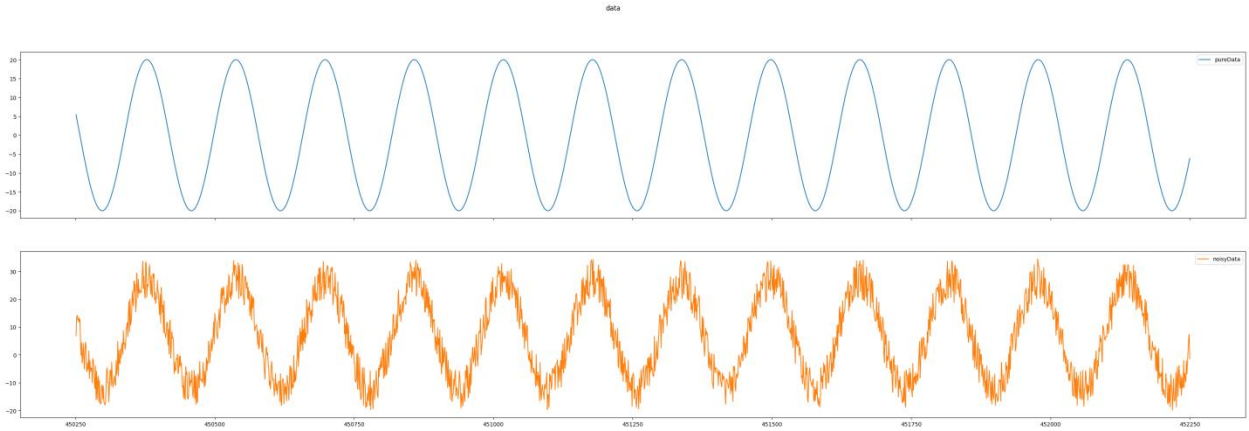


Fig. 7.7 Waveform of Dataset Pure (Up) vs Gaussian Noise induced Noisy Signal (Down)

The pure signal serves as the desired noiseless output emitted by our PMSM/BLDC motor during shunt sensing, as depicted in Fig. 7.7 Gaussian noise has been deliberately introduced into the clean signal, though any application-specific noise can be incorporated to train the algorithm as preferred. The choice of Gaussian noise arises from two primary considerations. Firstly, visual inspection using a digital oscilloscope, as illustrated in Fig. [7.7], revealed Gaussian noise closely resembling the observed noise pattern. Secondly, our initial focus was primarily on testing the algorithm's functionality and real-time inference on the edge, rather than tailoring it to a specific application.

We have meticulously generated approximately 380 hours of simulation data, comprising 1,371,753 data points represented in seconds. As depicted in Fig. 7.7, the x-axis denotes Time in seconds, while the y-axis represents Amplitude. The dataset has been resized to include 109,763 1-D matrices of dimension (100, 1), corresponding to the input and output layers of the neural network.

Fig. 7.9 visually elucidates the training architecture for the autoencoder algorithm. The encoder-decoder pair receives the noisy signal and adjusts its weights to effectively denoise and reconstruct it to resemble the clean version. Additionally, Fig. [7.8] provides a detailed depiction of the architecture.

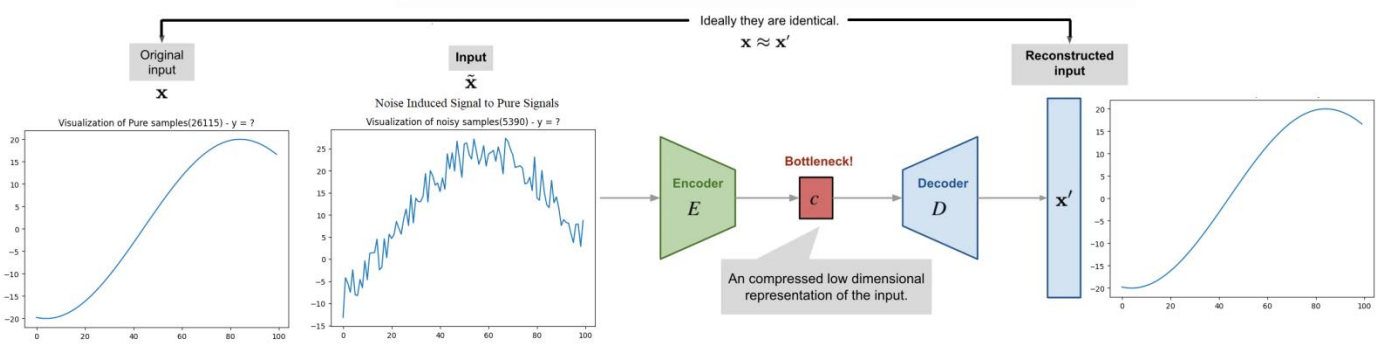


Fig. 7.8 Training Architecture for Convolutional Autoencoder

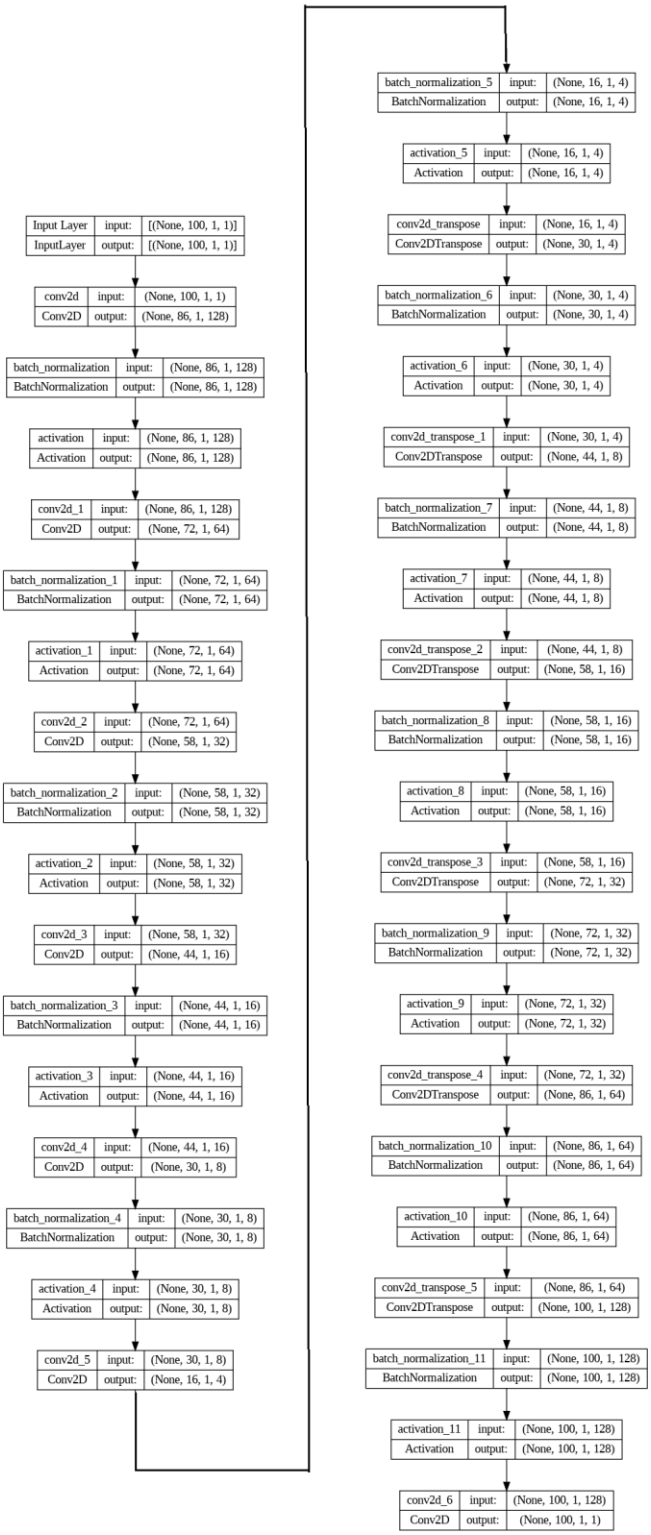


Fig. 7.9 Architecture of the Convolutional Autoencoder Neural Network

The training comprised 100 epochs; however, due to a pre-set callback function designed to trigger a pause in training upon stagnation in Loss, the training halted at 85 epochs. Fig. 7.10 illustrates the Model Loss graph, depicting the training performance.

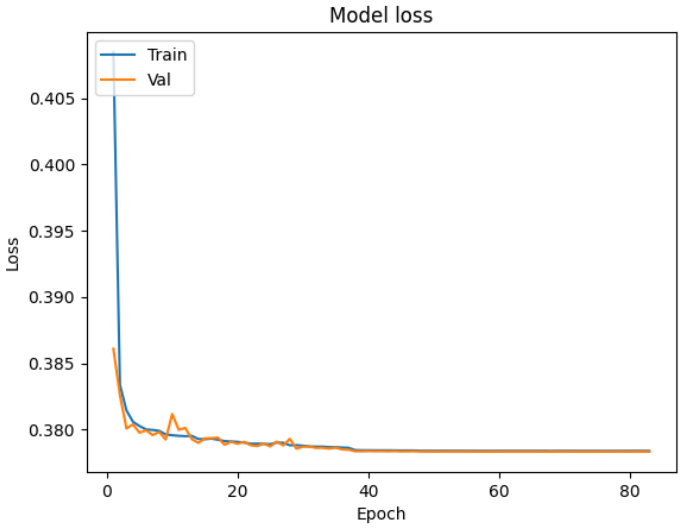


Fig. 7.10 Training Loss Waveform for 86 Epochs

7.4.1 Results - Denoising Reconstruction

Fig. demonstrates the waveforms for reconstruction performance.

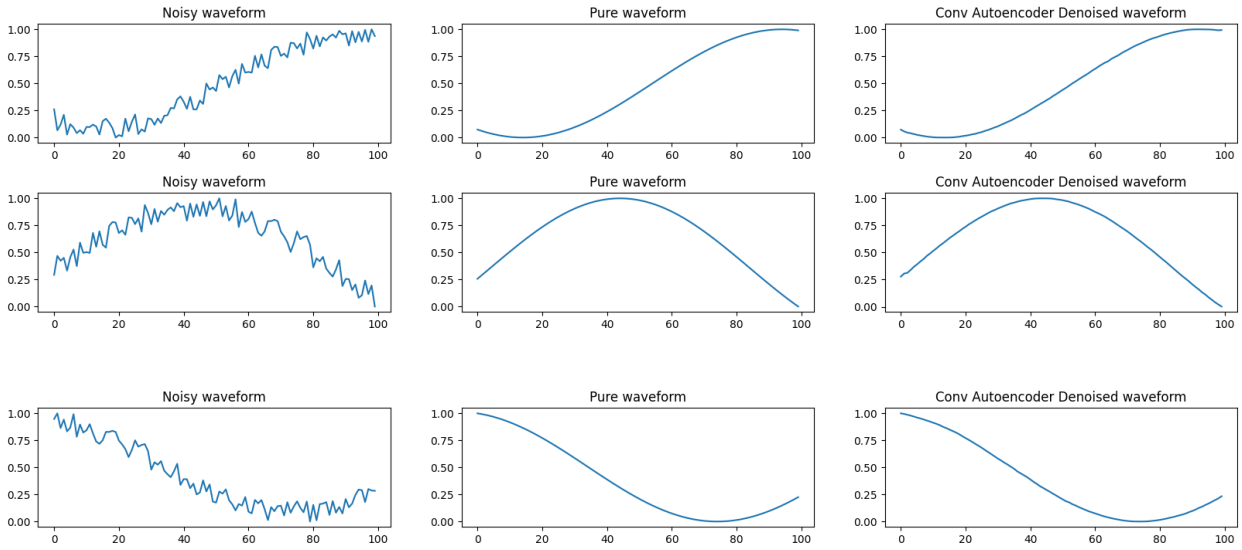


Fig. 7.11 Reconstruction performance of Convolutional Autoencoder

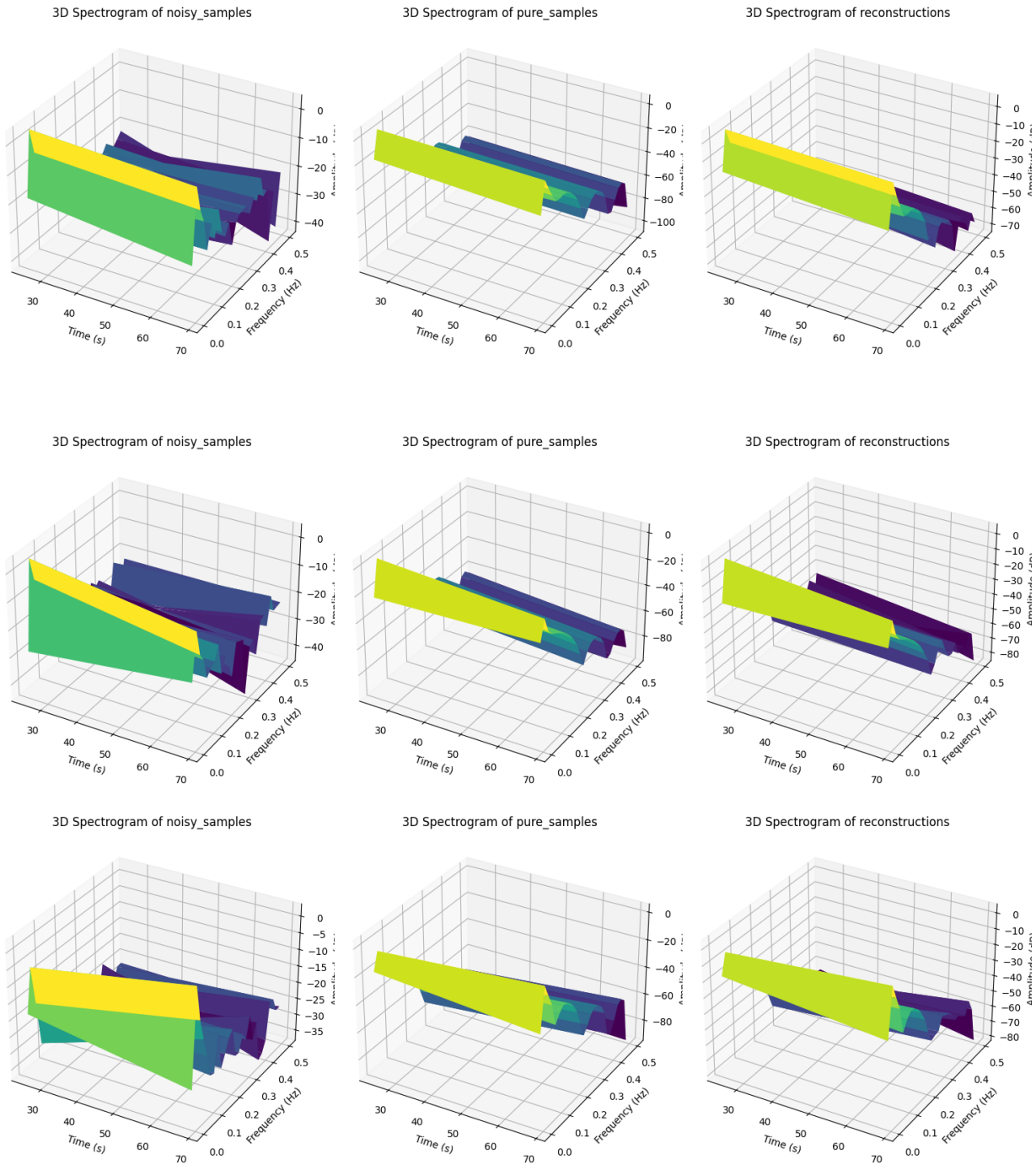


Fig. 7.12 Reconstruction performance of Convolutional Autoencoder via Spectogram

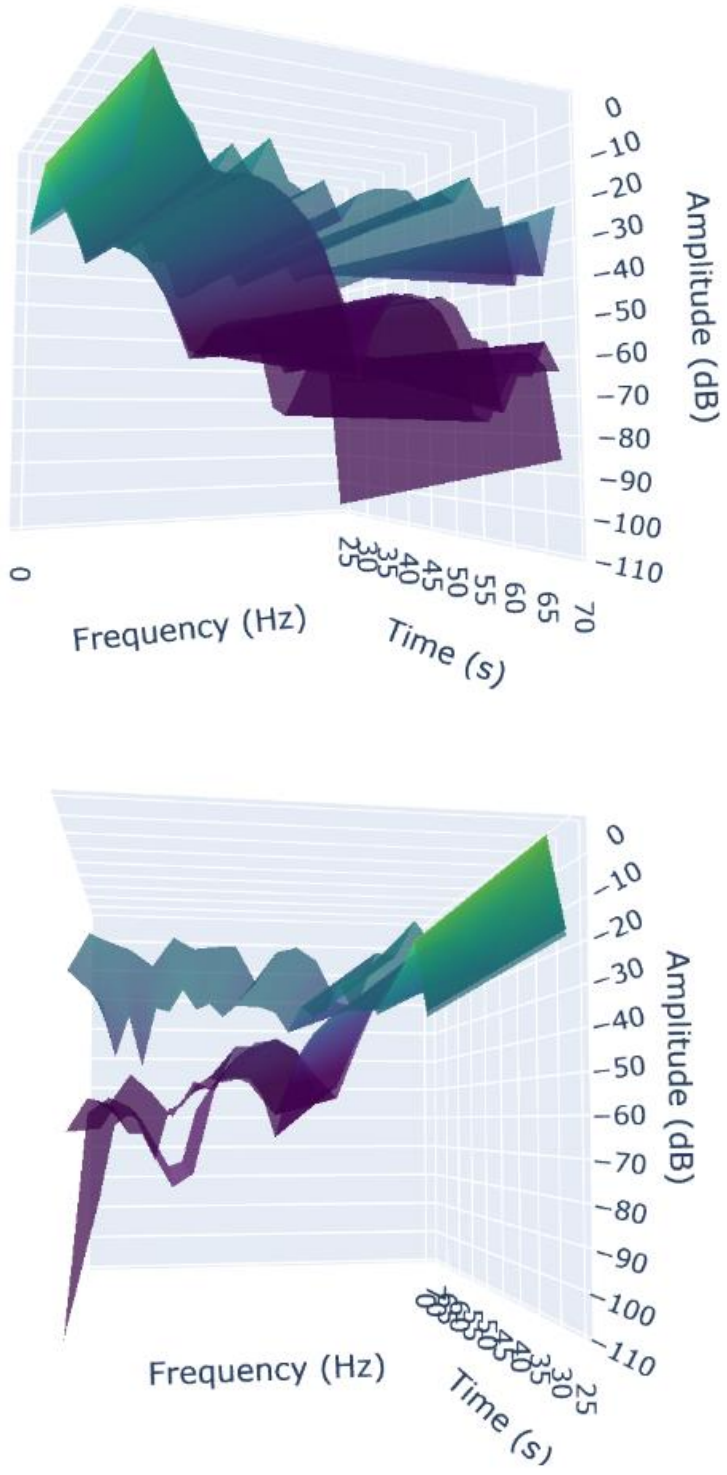


Fig. 7.13 3D Spectogram Representing Results

7.5 Performance Metrics and Interpretation of the Convolutional Autoencoder Model

Performance analysis was conducted between the Pure and Reconstructed Signals using various metrics: Mean Squared Error, Peak Signal-to-Noise Ratio, Mean Absolute Error, Structural Similarity Index, Cross-Correlation, Signal-to-Noise Ratio, Pearson Correlation Coefficient, and Kullback-Leibler Divergence. Fig. 7.14 presents the Results and Interpretation Remarks.

Metric	Value	Interpretation
Mean Squared Error	2.83083646113627e-05	Lower values indicate better performance, as it means the denoised signal is closer to the pure signal.
Peak Signal-to-Noise Ratio	45.48085219323559	Higher values indicate better performance, as it means the denoised signal has higher fidelity to the pure signal relative to the noise level.
Mean Absolute Error	0.004015823807711367	Lower values indicate better performance, similar to MSE.
Structural Similarity Index	0.9998862261489707	Values close to 1 indicate better performance, as it means the denoised signal retains the structural characteristics of the pure signal.
Cross-Correlation	0.9999249917483131	Values close to 1 indicate better performance, as it means the denoised signal closely follows the variations of the pure signal.
Signal-to-Noise Ratio	12692.738733482258	Higher values indicate better performance, as it means the signal power is much higher compared to the noise.
Pearson Correlation Coefficient	0.9999249917483126	Values close to 1 indicate better performance, indicating a strong linear relationship between the denoised and pure signals.
Kullback-Leibler Divergence	0.0440039203191674	Lower values indicate better performance, as it means the distributions of the denoised and pure signals are closer to each other.

Fig. 7.14 Results and Interpretation results

7.6 Experiment and Results – Real time Inference in the Edge Device STM32H755 Microcontroller to Denoise Noisy Signal

The trained model underwent conversion into an edge-executable model through model compression techniques detailed in Chapter [Specify Chapter]. The edge device utilized is the STMicroelectronics STM32H755ZI-Q, featuring Dual 32-bit Arm® Cortex®-M7 up to 480MHz and -M4 MCUs, 2MB flash, 1MB RAM, 46 communication and analog interfaces, as well as SMPS crypto capabilities [8]. Fig. [Enter Figure Number] illustrates the real-time inference on the edge module.

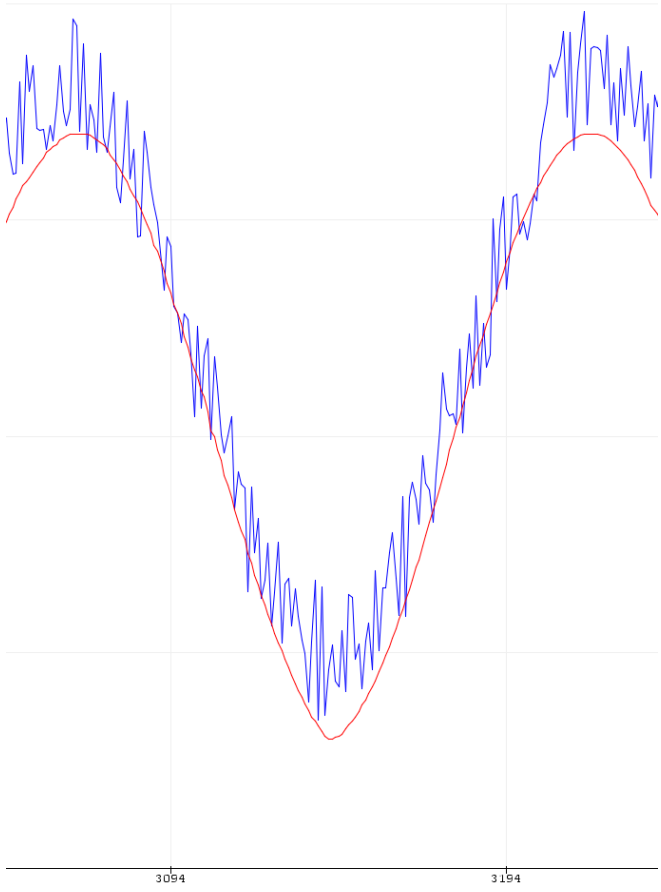


Fig. 7.15 Real Inference Convolutional Autoencoder running on STM32H755 MCU

7.7 Conclusion

In summary, Chapter 6 delves into the innovative application of Stacked Convolutional Auto Associative Neural Networks for denoising 3-phase shunt current measurements, a critical aspect in Sensorless Motor Control and Field-Oriented Control (FOC) systems. Traditionally, hardware and software filters have been employed for noise mitigation in shunt current signals routed through Analog-to-Digital Converters (ADCs). However, this chapter introduces a novel approach utilizing neural networks to denoise these signals, leveraging the robustness and fault tolerance behavior of deep learning models.

The chapter commences with an introduction to Auto Associative Neural Networks or Autoencoders, elucidating their fundamental architecture and purpose in discerning meaningful patterns within data. Specifically, Denoising Autoencoders are explored as a methodology to learn denoising abilities, wherein the input vector is deliberately corrupted to facilitate the learning process. Subsequently, the Denoising Autoencoder algorithm is detailed, emphasizing its role in mitigating noise from shunt current measurements. Synthetic simulation data, comprising noise-free and noisy signals, is generated to train a Stacked Convolutional Autoencoder model. The architecture and training process of the Convolutional Autoencoder are meticulously explained, along with the reconstruction performance results showcased through waveforms and spectrograms. Performance metrics including Mean Squared Error, Peak Signal-to-Noise Ratio, and Structural Similarity Index are employed to evaluate the model's denoising efficacy. Furthermore, real-time inference on an edge device, specifically the STM32H755 microcontroller, is demonstrated, highlighting the practical implementation of the trained model in real-world scenarios.

Overall, the chapter underscores the effectiveness of deep learning techniques, particularly Convolutional Autoencoders, in denoising shunt current measurements, thereby enhancing the accuracy and efficacy of Sensorless Motor Control and FOC systems.

7.8 References

[1] *High Precision Voltage Reference for ADC: Review and Design Methodology*. (2022, October 28). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10011299

[2] *Reconfigurable and low complexity variable digital filter design*. (2014, April 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/6950053

[3] *Improved Denoising Auto-Encoders for Image Denoising*. (2018, October 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8633143

[4] *X-CUBE-MCSDK - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/embedded-software/x-cube-mcsdk.html

[5] *STEVAL-PTOOL4A - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/evaluation-tools/steval-ptool4a.html#overview

[6] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, January 1). *Extracting and composing robust features with denoising autoencoders*. https://doi.org/10.1145/1390156.1390294

[7] *Deep Learning*. (n.d.). https://www.deeplearningbook.org/

[8] *STM32H745/755 - STMicroelectronics*. (n.d.). STMicroelectronics. https://www.st.com/en/microcontrollers-microprocessors/stm32h745-755.html

Chapter 8

Conclusion

In this thesis, we have presented an innovative framework that harnesses the power of deep learning and traditional motor control methodologies for sensorless diagnosis and anomaly detection in electric motor-driven control systems. Our framework addresses the pressing need for efficient monitoring, predictive maintenance, and anomaly detection in industrial applications where electric motors play a vital role.

Contributions and Innovations:

- **Framework Development:** We developed a comprehensive framework that utilizes the intrinsic electrical signatures generated by the Field-Oriented Control (FOC) algorithm for real-time diagnosis and classification of motor operating conditions and anomalies.
- **Integration of Deep Learning in the Edge:** By integrating deep learning algorithms, particularly Convolutional Neural Networks (CNNs), at the edge, we enabled accurate detection of abnormal patterns indicative of motor degradation or malfunction, thereby enhancing predictive maintenance capabilities.
- **Utilization of Motor as Sensor:** Our approach explores the potential for electric motors themselves to function as sensors, enabling early detection of anomalies without the need for additional physical sensors through the concurrent operation of neural algorithms at the edge device alongside the control algorithm.
- **Model Compression:** We addressed the challenge of deploying complex neural networks on resource-constrained platforms by significantly reducing the memory footprint through compression techniques, paving the way for widespread adoption of deep learning techniques in edge computing applications.
-

- **Signal Denoising:** We explored the application of neural algorithms in edge computing for signal denoising, replacing hardware filters with intelligent neural algorithms to enhance signal processing efficiency in the edge device.

Key Findings from Individual Chapters:

Chapter 2: Provided a foundational understanding of Sensorless Field-Oriented Control (FOC) mechanisms, highlighting its role in revolutionizing AC motor control paradigms and enabling precise control over torque and flux components.

Chapter 3: Explored Convolutional Neural Networks (CNNs) and their role in pattern recognition, showcasing their effectiveness in tasks such as image classification and signal data analysis.

Chapter 4: Investigated neural network model compression techniques, particularly focusing on quantization, and demonstrated their practical application for efficient deployment on edge devices.

Chapter 5: Examined intelligent anomaly detection techniques for predictive maintenance, emphasizing their significance in optimizing asset performance and preventing major accidents.

Chapter 6: Detailed the process of training a CNN for anomaly classification and motor health estimation, showcasing the model's performance validation and real-time deployment on an edge device.

Chapter 7: Introduced the innovative application of Convolutional Autoencoders for denoising shunt current measurements, enhancing the accuracy and efficacy of Sensorless Motor Control and FOC systems.

Chapter 9

Future Scope of Work

Moving forward, our research lays the groundwork for ongoing exploration and innovation in the domain of intelligent electric motor control systems. As we look ahead, several avenues emerge for further advancement and refinement of our framework, as well as for the exploration of new frontiers in sensorless diagnosis and anomaly detection methodologies. The future scope of our work encompasses the following key areas:

- **Innovation in Neural Net Architectures and Algorithms:** Future research efforts could focus on exploring novel neural network architectures tailored specifically for electric motor control systems. Additionally, there is a need for research into advanced deep learning algorithms that can further enhance the accuracy and efficiency of anomaly detection and predictive maintenance capabilities.

- **Model Compression Algorithm Innovations:** The development of more efficient model compression algorithms will be crucial for deploying complex neural networks on resource-constrained edge devices. Innovations in this area will enable the deployment of sophisticated deep learning models in real-world industrial settings, thereby enhancing the scalability and accessibility of our framework.

- **Data Augmentation and Synthetic Simulation Data Generation Techniques**: Research into data augmentation techniques and synthetic simulation data generation methods will enable the acquisition of diverse datasets without the need for extensive hardware experimentation. These techniques will facilitate the training of more robust and generalized neural network models, thereby improving the reliability and accuracy of anomaly detection algorithms.

- **Exploration of Other Hardware Architectures:** Research efforts should also be directed towards exploring alternative hardware architectures that can efficiently support neural network algorithms. Investigating the compatibility and performance of different hardware platforms with respect to deep learning inference will broaden the applicability of our framework across diverse industrial settings.

- **Research in More Anomaly Patterns:** There is a need for comprehensive research into additional anomaly patterns that may occur in electric motor-driven control systems. By expanding the scope of anomaly detection algorithms to encompass a broader range of potential anomalies, we can further enhance the predictive maintenance capabilities of our framework and ensure the early detection of critical issues.

- **On-Device Learning for Extreme Edge Computing:** Exploring the feasibility of on-device learning techniques for extreme edge computing environments represents a promising avenue for future research. By enabling edge devices to adapt and learn from real-time data without relying on centralized processing, on-device learning can significantly enhance the autonomy and responsiveness of electric motor control systems.

- **Prescriptive Maintenance Strategies:** The integration of machine learning algorithms for prescriptive maintenance holds immense potential for optimizing asset performance and minimizing downtime costs. Future research efforts should focus on developing intelligent algorithms that can adjust operating conditions in real-time based on predictive maintenance insights, thereby maximizing asset efficiency and longevity.

In conclusion, the thesis contributes significantly to bridging the gap between traditional motor control methodologies and cutting-edge deep learning techniques. By embracing the future scope outlined above, we can continue to push the boundaries of innovation in electric motor control systems, ultimately enhancing their reliability, efficiency, and predictive maintenance capabilities across various industrial applications.