ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

REAL-TIME ANOMALY DETECTION IN UAV SYSTEMS USING TINYML ON ARM CORTEX-M MICROCONTROLLERS

M.Sc. THESIS

Mehmet Alperen BAKICI

Department of Defense Technologies

Defense Technologies Program

NOVEMBER 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

REAL-TIME ANOMALY DETECTION IN UAV SYSTEMS USING TINYML ON ARM CORTEX-M MICROCONTROLLERS

M.Sc. THESIS

Mehmet Alperen BAKICI
(514211014)

Department of Defense Technologies

Defense Technologies Program

Thesis Advisor: Prof. Dr. Ece Olcay GÜNEŞ

NOVEMBER 2024

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ARM CORTEX-M MİKRODENETLEYİCİLERDE GÖMÜLÜ MAKİNE ÖĞRENMESİ KULLANARAK İHA SİSTEMLERİNDE GERÇEK ZAMANLI ANOMALİ TESPİTİ

YÜKSEK LİSANS TEZİ

Mehmet Alperen BAKICI
(514211014)

Savunma Teknolojileri Anabilim Dalı

Savunma Teknolojileri Programı

Tez Danışmanı: Prof. Dr. Ece Olcay GÜNEŞ

KASIM 2024

Mehmet Alperen BAKICI, a M.Sc. student of İTU Graduate School student ID 514211014, successfully defended the thesis/dissertation entitled “REAL-TIME ANOMALY DETECTION IN UAV SYSTEMS USING TINYML ON ARM CORTEX-M MICROCONTROLLERS”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Ece Olcay GÜNEŞ** İstanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Revna ACAR VURAL** Yildiz Technical University	.....
	Assistant Prof. Dr. Metin HÜNER İstanbul Technical University	.....

Date of Submission : 09 Oct 2024
Date of Defense : 15 Nov 2024

To my spouse and family,

FOREWORD

First of all, I would like to thank my advisor, Prof. Dr. Ece Olcay GÜNEŞ, for her patience, support, encouragement, and guidance throughout this research.

I am also deeply grateful to Prof. Dr. Mustafa DOĞAN, Prof. Dr. Sıtkı Çağdaş İNAM, and Assoc. Prof. Dr. Selda GÜNEY for their invaluable guidance and support in helping me pursue my master's degree.

I would like to express my gratitude to Mr. Selçuk BAYRAKTAR, Board Chairman of Baykar Technology and CTO, for his support during my academic education. I am also thankful to my employer, Baykar Technology, for providing me with the support and opportunities necessary to complete my master's degree, and for enabling me to conduct research, development, and projects. Additionally, I would like to thank Mr. Bedreddin KARAKUŞ, Head of the Embedded Software Technologies Unit, and my colleagues in the Power Systems Software Development Team for their patience and support.

I want to express my deepest gratitude to my beloved spouse, Tuba ŞENGÜN BAKICI, for her unwavering support, patience, and encouragement throughout my academic journey. Your love and belief in me have been my greatest source of strength. Thank you for always being by my side and for your endless sacrifices.

Finally, I wish to thank my family, Asiye, Bülent, and Aslıhan BAKICI, for their constant support throughout my life.

October 2024

Mehmet Alperen BAKICI
Electrical and Electronics Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

AD : Anomaly Detection
AI : Artificial Intelligence
ALFA : AirLab Failure and Anomaly
ARM : Advanced RISC Machines
AUC : Area Under the Curve
BMS : Battery Management System
CAN : Control Area Network
CFD : Computational Fluid Dynamics
CFS : Correlation-based Feature Selection
CMSIS : Cortex Microcontroller Software Interface Standard
CNN : Convolutional Neural Network
CPU : Central Processing Unit
CSV : Comma-Seperated Values
DL : Deep Learning
EHV : Extra High Voltage
EMC : Electromagnetic Compatibility
EMI : Electromagnetic Interference
FCU : Flight Control Unit
FDI : Fault Detection and Isolation
FPR : False Positive Rate
FPU : Floating Point Unit
GNSS : Global Navigation Satellite Systems
GPS : Global Positioning System
GUI : Graphical User Interface
HVAC : Heating, Ventilation and Air Conditioning
I^2C : Inter-Integrated Circuit
IOT : Internet of Things
KB : Kilobyte
Li-ion : Lithium-ion
LSTM : Long Short-Term Memory
MAC : Multiply-accumulate
MB : Megabyte
MCU : Microcontroller Unit

ML : Machine Learning
MRI : Magnetic Resonance Imaging
MSE : Mean Squared Error
NaN : Not a Number
NCP : Neural Co-processor
NN : Neural Network
QNN : Q-bit Quantized Neural Networks
RAM : Random Access Memory
RISC : Reduced Instruction Set Computing
RNN : Recurrent Neural Network
ROC : Receiver Operating Characteristic
ROS : Robot Operationg Sytems
RTOS : Real-Time Operating System
SIMD : Single Instruction, Multiple Data
SMOTE : Synthetic Minority Over-sampling Technique
SNR : Signal to Noise Ratio
SoC : State of Charge
SoH : State of Health
SPI : Serial Peripheral Interface
SVD : Singular Value Decomposition
TFLite : TensorFlow Lite
TFLM : Tensorflow Lite for Microcontrollers
TinyML : Tiny Machine Learning
TPR : True Positive Rate
UART : Universal Asynchronous Receiver-Transmitter
UAV : Unmanned Aerial Vehicle
USB : Universal Serial Bus
UV : Ultraviolet
VAE : Variational Autoencoders
XAI : Explainable Artificial Intelligence

LIST OF TABLES

Page

LIST OF FIGURES

Page

REAL-TIME ANOMALY DETECTION IN UAV SYSTEMS USING TINYML ON ARM CORTEX-M MICROCONTROLLERS

SUMMARY

In recent years, the significance of machine learning (ML) within microcontroller-centric systems has expanded, particularly within the realms of defense technology and UAV (Unmanned Aerial Vehicle) implementations. These systems often encounter difficulties due to limited computing resources and power constraints. Tiny Machine Learning (TinyML) is emerging as a promising solution to these issues, enabling efficient ML implementation in environments with limited resources. TinyML enables low-power microcontrollers to process machine learning tasks, facilitating enhanced and instantaneous decision-making capabilities directly on the device. This not only boosts the capabilities of UAVs but also extends the use of advanced ML in resource-constrained settings.

This research delves into integrating machine learning functionalities into Unmanned Aerial Vehicle (UAV) systems, leveraging the ARM Cortex-M microprocessor architecture via TinyML. The principal aim is to facilitate real-time anomaly detection, thereby bolstering UAV system security and imbuing it with greater dynamism and autonomy, reducing reliance on pilot interventions. In order to achieve the aforementioned objective of enhancing anomaly detection, a unique and specialized autoencoder model was conceptualized and developed. This particular model was intricately designed with the specific purpose of identifying and flagging anomalies within a given dataset Autoencoders, a class of neural networks, demonstrate remarkable proficiency in discerning and comprehending data patterns. They acquire an understanding of prevalent data patterns, enabling the identification of any data instances diverging from these norms as anomalies.

The dataset used in this research is the ALFA (AirLab Failure and Anomaly) dataset, which contains data from various autonomous UAV flights specifically gathered for failure and anomaly detection purposes. Utilizing this data set, the autoencoder architecture can be efficiently educated and assessed. This ensures that the model can recognize typical UAV operational patterns and accurately detect anomalies. Nevertheless, the deployment of the model on ARM Cortex-M Series Microcontroller Units (MCUs) presents considerable obstacles attributable to the constrained memory and computational capacities of such devices.

To address these challenges, the research utilizes the Common Microcontroller Software Interface Standard for Neural Network (CMSIS-NN) and TensorFlow Lite for model optimization. CMSIS-NN, a library developed by Arm, is optimized for ARM Cortex-M microcontrollers, aiding in neural network operations within the MCU's resource limitations. TensorFlow Lite is specifically tailored for running machine learning models on devices with restricted resources, enabling model conversion and compression while maintaining performance. These enhancements make the model suitable for deployment on the MCU, facilitating real-time anomaly detection within strict memory and processing constraints.

The successful implementation of the autoencoder algorithm on ARM Cortex-M microcontrollers exemplifies the viability and efficiency of utilizing TinyML for instantaneous anomaly identification in Unmanned Aerial Vehicle (UAV) platforms. This approach enhances the UAV's ability to continuously learn and adapt to new conditions, improving its anomaly detection capabilities in various operational scenarios. As a result, UAVs become more self-learning and agile in responding to unforeseen situations, significantly enhancing their operational reliability and security.

This research adds to the body of knowledge on UAV system security by showcasing the efficient application of TinyML for executing real-time anomaly detection on microcontrollers with limited resources. The integration of CMSIS-NN and TensorFlow Lite optimizations ensures that even with limited computational resources, sophisticated ML models can be deployed to enhance the functionality and security of UAV systems.

ARM CORTEX-M MİKRODENETLEYİCİLERDE GÖMÜLÜ MAKİNE ÖĞRENMESİ KULLANARAK İHA SİSTEMLERİNDE GERÇEK ZAMANLI ANOMALİ TESPİTİ

ÖZET

Son yıllarda, mikrodenetleyici merkezli sistemlerde makine öğreniminin (ML) önemi, özellikle savunma teknolojisi ve İnsansız Hava Aracı (İHA) uygulamaları alanlarında büyük bir artış göstermiştir. Bu sistemler, genellikle sınırlı işlem gücü, bellek kapasitesi ve enerji kaynakları nedeniyle geleneksel makine öğrenimi yöntemlerinin uygulanmasında zorluklarla karşılaşmaktadır. Bu nedenle, daha hafif, enerji verimli ve kaynak dostu çözümler sunabilen yaklaşımlara olan ihtiyaç giderek artmaktadır. Gömülü makine öğrenimi (Tiny Machine Learning), bu sınırlamaları aşmak için yenilikçi bir çözüm olarak ortaya çıkmıştır. Bu teknoloji, düşük güçlü mikrodenetleyiciler üzerinde makine öğrenimi modellerinin çalıştırılmasını mümkün kılarak, cihaz üzerinde gerçek zamanlı ve gelişmiş karar verme süreçlerini desteklemektedir. Ayrıca bulut tabanlı veri işleme gereksinimlerini ortadan kaldırarak, veri aktarımından kaynaklanan gecikmeleri minimize etmekte ve sistem güvenliğini artırmaktadır. Özellikle, İHA gibi zaman kritik uygulamalarda bu özellikler operasyonel başarı açısından hayati bir öneme sahiptir.

İnsansız hava araçlarında hata ve anomali tespitinin zorlukları, sistemin karmaşık yapısından ve uçuş sırasında oluşabilecek çok sayıda değişkenin kontrol edilmesi gerekliliğinden kaynaklanmaktadır. Bir İHA'nın uçuş güvenliğini sağlamak, motor ve itki sistemlerinin performansı, navigasyon sistemlerinin doğruluğu, güç sistemi ve batarya sağlığı ile uçuş kontrol yüzeylerinin durumu gibi kritik aviyonik sistemlerin sürekli olarak izlenmesini gerektirir. Bu durum, manuel veya kural tabanlı yöntemlerle gerçekleştirildiğinde hem zaman alıcı hem de hata yapmaya açık bir süreç haline gelir. Bu bağlamda, gömülü makine öğrenimi teknolojilerinin uygulanması, hata durumlarının tespit edilmesini çok daha hızlı ve otonom bir şekilde gerçekleştirme potansiyeli sunmaktadır. Gömülü makine öğrenimi ile İHA sistemleri, uçuş sırasında oluşabilecek anormallikleri gerçek zamanlı olarak tespit edebilir ve gerekli önlemleri hızlı bir şekilde alabilir. Bu durum, hem uçuş güvenliğini artırmakta hem de İHA'nın

operasyonel etkinliğini ve bağımsızlığını desteklemektedir. Gömülü makine öğreniminin en büyük avantajları enerji verimliliği, düşük maliyetli donanımlarda çalışabilme yeteneği ve sınırlı bellek kaynaklarını optimize edebilme kapasitesidir. Bu özellikler, İHA'ların uzun süreli operasyonlarını desteklerken, daha güvenilir bir sistem tasarımına olanak tanır.

Gömülü makine öğreniminin sunduğu enerji verimliliği, düşük maliyetli donanımlarda çalışabilme kapasitesi ve yüksek doğruluk oranlarını koruyabilme yeteneği, İHA sistemleri için umut verici bir çözüm sunmaktadır. Ancak, bu teknolojinin uygulanmasında çeşitli teknik zorluklar bulunmaktadır ve bu zorlukların üstesinden gelmek için etkili stratejiler geliştirilmesi gerekmektedir. Gömülü sistemlerde makine öğrenimi uygulamaları, sınırlı işlem gücü, bellek kapasitesi ve enerji kaynakları gibi kısıtlamalar nedeniyle çeşitli teknik zorluklar barındırmaktadır. Özellikle mikrodenetleyiciler, geleneksel makine öğrenimi modellerini çalıştırmak için yeterli donanım kapasitesine sahip değildir. Bu cihazlar genellikle birkaç yüz kilobyte bellek ve düşük işlemci hızları ile sınırlıdır. Bu durum, karmaşık yapay zeka modellerinin doğrudan mikrodenetleyiciler üzerinde çalıştırılmasını oldukça zorlaştırmaktadır. Özellikle derin öğrenme tabanlı modeller, büyük tensor işlemleri ve yüksek düzeyde paralel işlem gücü gerektirdiğinden, gömülü sistemlerde verimli bir şekilde çalıştırılmaları için optimize edilmelidir. Gömülü makine öğrenim teknolojisi bu noktada çözüm sunarak model sıkıştırma, kuantizasyon ve donanım hızlandırma gibi tekniklerle bu zorlukların üstesinden gelmeyi mümkün kılmaktadır. Ayrıca gömülü sistemlere makine öğrenim modellerinin entegrasyonunu kolaylaştırmakta hem de yüksek doğruluk oranlarını koruyarak güçlü bir anomali tespit sistemi geliştirilmesine imkan tanımaktadır. Bununla birlikte, gömülü sistemlerin sınırlamaları sadece donanım kaynaklarıyla sınırlı kalmaz. Bu tür sistemlerin genellikle batarya ile çalışması, enerji tüketiminin de kritik bir parametre olmasını gerektirir. Özellikle İHA gibi uygulamalarda, enerji verimliliği sağlanamadığında operasyonel süreler ciddi şekilde kısalabilir. Gömülü makine öğrenimi tabanlı çözümler, enerji tüketimini minimize ederek sistemin verimli bir şekilde çalışmasını mümkün kılmaktadır. Bunun yanı sıra, sınırlı bellek kaynaklarıyla başa çıkmak için kullanılan model sıkıştırma teknikleri, hem bellek kullanımını optimize etmekte hem de çıkarım sürelerini kısaltmaktadır.

Gömülü makine öğreniminin gömülü sistemlerdeki bir diğer avantajı, uçta çıkarım yapma yeteneğidir. Uçta çıkarım, verinin merkezi bir sunucuya aktarılmadan doğrudan cihaz üzerinde işlenmesini ifade eder. Bu yaklaşım, özellikle veri gizliliği ve güvenliği açısından büyük avantajlar sunar. İHA sistemlerinde uçuş sırasında toplanan veriler, genellikle hassas operasyonel bilgiler içerir. Bu verilerin merkezi bir sisteme aktarılması, hem veri gizliliği ihlallerine yol açabilir hem de iletişim hatlarında ek bir yük oluşturabilir. Gömülü makine öğrenimi, verilerin cihaz üzerinde işlenmesini mümkün kılarak bu tür riskleri en aza indirmektedir. Ayrıca, uçta çıkarım, geniş bant iletişim hatlarına olan ihtiyacı da azaltır. Veri aktarımı gereksiniminin azalması, hem iletişim gecikmelerini minimize eder hem de bant genişliği kullanımını optimize eder. Bu durum, özellikle geniş bir İHA filosunun aynı anda çalıştığı operasyonlarda büyük avantaj sağlar. Uçta çıkarım aynı zamanda gerçek zamanlı karar alma süreçlerini hızlandırır. Merkezi sistemlerde işlenen verilerde, veri aktarımı sırasında oluşabilecek gecikmeler kritik kararların zamanında alınmasını engelleyebilir. Gömülü makine öğrenimi destekli sistemlerde ise veri işleme cihaz üzerinde gerçekleştirildiği için, anomali tespiti ve gerekli müdahaleler çok daha hızlı bir şekilde uygulanabilir. Bu durum, yalnızca operasyonel verimliliği artırmakla kalmaz, aynı zamanda sistem güvenilirliğini ve uçuş güvenliğini de önemli ölçüde artırır. Gömülü makine öğreniminin sunduğu çözümler, gömülü sistemlerin sınırlamalarını aşarak hem veri gizliliği ve güvenliği sağlamada hem de enerji verimliliği ve gerçek zamanlı performans artırmada kritik bir rol oynamaktadır. Bu avantajlar, İHA sistemlerinin daha bağımsız, güvenli ve etkili bir şekilde çalışmasına olanak tanımaktadır. Ancak, bu faydaların elde edilebilmesi için makine öğrenim modellerinin mikrodenetleyicilere uygun şekilde optimize edilmesi ve donanım kısıtlamaları göz önünde bulundurularak uygulanması gerekmektedir.

Bu araştırmanın temel amacı, gömülü makine öğrenimi teknolojisinin gerektirdiği optimizasyon yöntemlerini inceleyerek ARM Cortex-M mikrodenetleyiciler üzerinde çalışabilen bir anomali tespit sistemi geliştirmektir. Geliştirilen sistem, İHA'ların uçuş sırasında karşılaşabileceği anomali durumlarını gerçek zamanlı olarak tespit ederek, uçuş güvenliğini artırmayı ve sistemin daha bağımsız çalışmasını sağlamayı hedeflemektedir. Bu bağlamda, araştırmamızda kullanılan ALFA (AirLab Failure and Anomaly) veri kümesi, farklı uçuş senaryolarını ve anomali durumlarını içeren zengin bir veri kaynağı olarak öne çıkmaktadır. Araştırmada kullanılan veri kümesi, Carnegie

Mellon Üniversitesi'ne bağlı AirLab tarafından sunulan ALFA veri kümesidir. Bu veri kümesi, motor gücü kaybı, dümenin veya kanatçıkların sabit pozisyonda kalması gibi kritik uçuş senaryolarını kapsayarak, uçuş performansı ve güvenliğini değerlendirmek için bir kaynak sunmaktadır. Veri setinin geniş kapsamı sayesinde, farklı uçuş koşullarını ve senaryolarını modellemek mümkün olmuştur. Örneğin, motor arızaları sırasında hızlanma, yükseklik ve manyetik alan ölçümlerindeki değişimler gibi kritik parametreler, sistemin anormallikleri algılama yeteneğini geliştirmek için model eğitiminde önemli bir rol oynamıştır. Veri kümesi, hem bireysel uçuş verileri hem de birleştirilmiş uçuş senaryoları üzerinde modellenmiştir. Bireysel uçuşlar üzerinde yapılan eğitimler, belirli bir senaryoya özgü performansı artırırken, birleştirilmiş uçuş verileriyle yapılan eğitimler, modelin genelleme yeteneğini güçlendirmiştir. Özellikle, birleşik uçuş verileri ile oluşturulan model, %76'lık bir doğruluk oranına ulaşarak genel bir arıza tespit sistemi geliştirilmiştir. Bu model, ARM Cortex-M mikrodenetleyiciler üzerinde gerçek zamanlı olarak çalışabilmesi için CMSIS-NN ve TensorFlow Lite optimizasyonlarıyla entegre edilmiştir. Araştırma sonucunda, geliştirilen autoencoder modelinin doğruluk oranlarının %71 ile %93 arasında değiştiği gözlemlenmiştir. Model, yalnızca enerji verimliliği sağlamakla kalmayıp, aynı zamanda diğer kritik görevler için sistem kaynaklarını serbest bırakmıştır. TensorFlow Lite ve CMSIS-NN gibi araçlar sayesinde, modelin bellek kullanımı 200 KB'ye düşürülmüş ve çıkarım süresi 1-2 milisaniye arasında gerçekleşmiştir. Bu optimizasyonlar, mikrodenetleyicinin sınırlı kaynakları içinde yüksek performans sağlamış ve modelin gerçek zamanlı uygulamalar için uygunluğunu kanıtlamıştır.

Bu çalışma, gömülü makine öğrenim teknolojisinin yalnızca akademik bir kavram olmaktan çıkarak gerçek uygulamalarda nasıl kullanılabileceğini göstermektedir. Geliştirilen sistem, İHA'ların uçuş güvenliğini artırmak, enerji tasarrufu sağlamak ve operasyonel maliyetleri düşürmek için etkili bir çözüm sunmaktadır. Bunun yanı sıra, savunma teknolojileri alanında da önemli bir etki yaratma potansiyeline sahiptir. Gömülü makine öğrenim teknolojisinin sunduğu gerçek zamanlı veri işleme yetenekleri, İHA'ların operasyonel kullanım ortamlarında daha bağımsız ve güvenilir bir şekilde çalışmasını sağlayarak savunma operasyonlarının başarı oranını artırabilir. Gelecek çalışmalarda, daha karmaşık veri kümeleri üzerinde yapılan testler ve hibrit öğrenme yöntemleriyle modelin performansının daha da artırılması hedeflenmektedir.

1. INTRODUCTION

The application of machine learning (ML) in microcontroller-based systems has brought about a revolution in numerous technological sectors, notably in defense and autonomous systems. Unmanned Aerial Vehicles (UAVs) play a crucial role in contemporary defense strategies due to their adaptability, effectiveness, and capacity to operate in hazardous environments while ensuring human safety. However, UAV systems face significant challenges primarily due to their limited computational resources and power constraints, which hinder the incorporation of advanced ML techniques necessary for enhancing the independent capabilities and operational efficiency of UAVs.

Tiny Machine Learning (TinyML) is seen as a hopeful answer to these obstacles. It allows ML models to be used on low-power microcontrollers, enabling instant decision-making and improving the intelligence of embedded systems. The key benefit of TinyML is its capacity to carry out intricate calculations on limited microcontroller resources, making it well-suited for tasks like UAVs, where power efficiency and lightweight processing are vital.

Within the realm of UAV systems, real-time anomaly detection stands out as a crucial function that can substantially elevate operational safety and security. Anomalies in UAV operations might arise from different factors, like sensor malfunctions, environmental interferences, and system failures. Timely detection of these anomalies is vital for averting potential failures and ensuring the dependability of UAV missions. Conventional anomaly detection methods often depend on extensive data processing capabilities, which are unfeasible for resource-limited UAV systems. This is where TinyML, with its ability to execute ML algorithms on microcontrollers, offers a noteworthy advantage.

This research focuses on the integration of real-time anomaly detection into UAV systems using TinyML on ARM Cortex-M microcontrollers. ARM Cortex-M series is widely used in embedded systems due to its performance and power efficiency

features. Applying machine learning models to ARM processors poses various challenges. The research discusses the applicability of machine learning models to ARM processor structures and the advantages and disadvantages encountered during this application.

In this research, an autoencoder model, a type of neural network known for its effectiveness in pattern recognition, was applied for anomaly detection. The model is trained using the ALFA (AirLab Failure and Anomaly) dataset, which furnishes comprehensive data from diverse autonomous UAV flights specifically collected for failure and anomaly detection purposes. This dataset enables the autoencoder to grasp the typical operational patterns of UAVs and pinpoint deviations from these patterns as anomalies.

The utilization of the autoencoder model on ARM Cortex-M microcontrollers gives rise to various challenges, mainly due to the restricted memory and processing capacities of these devices. To tackle these challenges, this research employs the Common Microcontroller Software Interface Standard for Neural Network (CMSIS-NN) and TensorFlow Lite for model optimization. CMSIS-NN, devised by Arm, is tailored for neural network operations on ARM Cortex-M microcontrollers, while TensorFlow Lite facilitates model conversion and compression, ensuring efficient performance within resource constraints.

1.1 Purpose of Thesis

The main goal of this thesis is to show how machine learning models can be successfully used in microcontroller units (MCUs). In the research carried out for this purpose, more effective, flexible and efficient software applications have been investigated in depth using Tiny Machine Learning capabilities. It is aimed to increase the operational functionality of the system by detecting real-time anomalies in unmanned aerial systems with TinyML capabilities.

Incorporating machine learning models into MCU-based UAV systems will facilitate the swift and dynamic identification of anomalies. This enhancement is pivotal for boosting the security and dependability of UAV operations, enabling timely detection and response to potential issues without human intervention. By reducing the need for

human supervision, the UAV can function with greater autonomy, thereby heightening its awareness of the surroundings and overall flight safety.

Moreover, this research strives to tackle the obstacles linked to deploying advanced machine learning models on devices with limited resources. The utilization of ARM Cortex-M microcontrollers, along with optimization techniques like CMSIS-NN and TensorFlow Lite, will guarantee the efficient operation of the models within the restricted memory and processing capacities of MCUs. This strategy will not only enhance the anomaly detection capabilities of UAVs but also open up possibilities for broader integration of TinyML in various embedded systems.

With this research, it is aimed to contribute to the field of embedded machine learning by presenting practical methods for implementing real-time anomaly detection on low-power microcontrollers. The knowledge gained from this research is expected to increase the independence and resilience of UAV systems, allowing them to better adapt to different operational scenarios and cope effectively with unpredictable conditions. This increased level of independence and reliability will significantly increase the overall safety and efficiency of UAV operations in the civil and defense sectors.

1.2 Literature Review

Tiny Machine Learning, commonly known as TinyML, is an innovative field at the intersection of machine learning and embedded systems. It focuses on bringing the power of machine learning to resource-constrained microcontroller units (MCUs) used in edge devices. The essence of TinyML lies in its ability to deploy machine learning models onto small hardware platforms, enabling efficient inference and real-time application in a variety of settings. This approach tackles several challenges prevalent in edge computing, such as the complexity of the deployment environment, data heterogeneity, and constraints related to computation and communication [1].

At the core of TinyML systems are MCUs, which are often integrated with neural co-processors (NCPs). This integration is pivotal in optimizing power consumption and reducing latency, crucial factors in edge computing. TinyML harnesses lightweight artificial neural network models, which are trained on raw data and subsequently

deployed on microcontrollers. These models are primarily used for classification tasks, a fundamental aspect of many machine learning applications [2].

To enhance the efficiency of TinyML systems, specialized deployment frameworks have been developed. One such framework is MinUn, which focuses on optimizing memory usage. This is achieved through innovative techniques, including the utilization of emerging number representations like posits. Such advancements are instrumental in generating efficient code tailored for the unique requirements of TinyML [3].

The progress in TinyML signifies a notable shift in machine learning, bringing its capabilities to low-power, cost-effective intelligent devices. The advancements allow for the deployment of machine learning inference on minuscule devices, greatly improving efficiency and performance. Overall, TinyML presents a compelling approach to democratize machine learning, enabling its integration into a myriad of low-power and intelligent edge devices.

1.2.1 Implementing TinyML on ARM Cortex MCUs

For the implementation of TinyML on ARM Cortex microcontroller units (MCUs), CMSIS-NN, an abbreviation for Cortex Microcontroller Software Interface Standard for Neural Networks, is a specialized software library developed by ARM. It is specifically designed for the implementation of neural networks on ARM Cortex-M processors, which are prevalently used in microcontroller-based embedded systems.

The strength of CMSIS-NN lies in its collection of efficient neural network kernels, which are meticulously crafted for ARM Cortex-M CPUs. These kernels play a pivotal role in enhancing the performance and reducing the memory footprint of neural network applications, particularly in the domain of intelligent IoT edge devices [5]. The design of CMSIS-NN is aligned with the operational capabilities of ARM Cortex-M processors, focusing on memory efficiency and computational speed. This is evident in its optimization strategies for 2D convolutions and fully connected layers, which are critical components of many neural network architectures [6].

A notable implementation of CMSIS-NN is observed in its deployment on the STM32F779I-EVAL board, which is equipped with an ARM Cortex-M7 core. This board exemplifies a balance of low cost and energy efficiency, making it an ideal candidate for evaluating the efficacy of CMSIS-NN in real-world scenarios [7].

Additionally, emerging research and work-in-progress results are indicative of the potential advancements with CMSIS-NN. For instance, Q-bit Quantized Neural Networks (QNNs) implemented on Cortex-M7 microcontrollers using CMSIS-NN have shown promising results in achieving a reduced memory footprint and improved energy efficiency. These findings suggest the possibility of further enhancements in performance, making CMSIS-NN a key player in the evolution of TinyML applications [8].

1.2.2 Related works

The field of Tiny Machine Learning has seen various applications and research dedicated to utilizing machine learning in resource-limited settings. This section examines important works demonstrating the versatility and promise of TinyML in diverse domains.

1.2.2.1 Adaptive traffic control with TinyML

Automatic traffic scheduling can improve traffic system efficiency by reducing delays and congestion. The proposed algorithm utilizes ML and TinyML to automate traffic scheduling based on vehicle density. Piezoelectric sensors embedded across each lane are used to detect vehicles and classify them based on pick-up speed. A TinyML model is employed to predict green signal timings based on the identified traffic density [9]. The combination of vehicle mass and axle details is used for accurate vehicle classification. The system measures vehicular density accurately and adapts traffic control accordingly. The use of TinyML makes the approach cost-effective and suitable for real-time deployment. TinyML was used in the related study, and the TinyML model estimates the green signal time required for vehicles to clear traffic during the red signal [9].

Overall, the proposed algorithm combines ML and TinyML to automate traffic control based on vehicle density, improving efficiency and reducing delays.

1.2.2.2 Predictive machine maintenance using TinyML

Anomaly detection is a key component of Tiny Machine Learning for predictive machine maintenance in an industrial environment. TinyML utilizes unsupervised learning algorithms to identify patterns in data that deviate from expected behavior. TensorFlow Lite Micro is used to train the TinyML model for anomaly detection. The

implementation of TinyML in predictive maintenance allows for the detection of anomalies in equipment before critical failures occur, enabling scheduled maintenance and optimization of part inventory. Predictive machine maintenance using TinyML has the potential to improve equipment uptime and reduce maintenance costs in various industries, such as Heating, Ventilation and Air Conditioning (HVAC) systems [10].

1.2.2.3 TinyOL: TinyML with online learning on microcontrollers

Current TinyML solutions are based on batch/offline settings, where the neural network is trained using a large amount of pre-collected data on a powerful machine and then flashed onto microcontrollers. This results in a static model that is hard to adapt to new data and adjust for different scenarios. The lack of on-device training capabilities in current TinyML solutions hinders the flexibility of the Internet of Things (IoT). The inability to perform incremental on-device training on streaming data limits the model's ability to continuously learn and adapt in real-time. Current TinyML solutions do not support online learning, which is crucial for constrained IoT devices [11]. The main purpose of TinyOL is to address the limitations of current TinyML solutions by enabling incremental on-device training on microcontrollers. It aims to overcome the static nature of current TinyML models, which are trained offline on powerful machines and then flashed onto MCUs, making them hard to adapt to new data and adjust for different scenarios. TinyOL utilizes the concept of online learning to enable on-device training on streaming data, allowing the model to continuously learn and adapt to new information. By providing on-device training capabilities, TinyOL enhances the flexibility of the Internet of Things by allowing the model to adapt to changing environments and scenarios in real-time. The system is designed specifically for constrained IoT devices, taking into account the limitations of power, memory, and computation [11].

1.2.2.4 Anomaly detection for unmanned aerial vehicle sensor data using a stacked recurrent autoencoder method with dynamic thresholding

The detection of anomalies in sensor data from Unmanned Aerial Vehicles is crucial for ensuring their safety and reliability. A new approach involves using a Stacked Recurrent Autoencoder Method with Dynamic Thresholding to capture complex temporal patterns. This method adjusts the anomaly detection threshold in real-time,

thus strengthening anomaly detection in UAV applications. These techniques could greatly improve the safety and reliability of UAV operations. The main focus of this study is on a specialized neural network model known as the Long Short-Term Memory (LSTM) Autoencoder. LSTM, categorized as a form of recurrent neural network (RNN), excels in capturing extended dependencies, which are critical for scrutinizing time series data like sensor recordings. An autoencoder, serving as an unsupervised learning technique, generates a condensed portrayal of input data and is frequently utilized for dimensionality reduction or feature acquisition. Within this framework, the LSTM autoencoder foretells data values and juxtaposes them with the input to detect anomalies. Once the deviation surpasses a prearranged threshold, the data point is flagged as anomalous [12]. Through the implementation of these sophisticated methodologies, the tenacity and credibility of UAV operations can be significantly fortified. This strategy capitalizes on the capabilities of LSTM networks in managing sequential data and the effectiveness of autoencoders in acquiring concise data representations. Consequently, the approach furnishes a sturdy structure for real-time anomaly detection, assuring prompt identification and alleviation of potential concerns in UAV systems.

The dataset utilized in the research is the ALFA dataset, which is specifically designed for fault detection in UAVs. This dataset comprises 10 pre-processed non-faulty flights, which are used to train the anomaly detection model.

The paper concludes that the proposed Stacked LSTM Autoencoder-based anomaly detection method, when combined with dynamic thresholding and a weighted loss function, demonstrates a strong performance in detecting anomalies in UAV sensor data, particularly during engine-failure scenarios. The method significantly reduces the delay in detecting true anomalous behavior and is not limited to the specific UAV dataset used in the study, indicating its potential applicability to a wide range of aircraft with varying specifications [12].

1.3 Original Contribution of the Research

The research highlights the originality of TinyML, which emerges as a new paradigm and offers opportunities for use in real-time anomaly detection for UAV systems. More specifically, the research leads the way in integrating Machine Learning models on ARM Cortex microcontroller units designed specifically for UAV systems. By

utilizing the efficiency and low energy usage of ARM Cortex MCUs, the research proves that it is feasible to implement advanced machine learning models on devices with limited resources.

This innovative implementation enhances the capabilities of real-time anomaly detection, allowing UAVs to promptly and accurately identify anomalies, thereby enhancing operational safety and efficiency without the need for continuous connectivity to central servers. The incorporation of TinyML facilitates efficient anomaly detection, consequently improving UAV security and operational efficiency. The optimization strategies investigated in this study, such as model quantization and the utilization of CMSIS-NN, play a crucial role in reducing the computational and memory requirements of neural network models. These optimizations enable the deployment of complex models on low-power MCUs, rendering the proposed solutions highly suitable for real-world applications. The study also conducts thorough experiments and assessments using real-world data obtained from UAV operations. This pragmatic approach ensures that the proposed solutions are not only theoretically sound but also practical in real-world settings, offering valuable insights for future implementations. The comprehensive evaluation of real-world data underscores the practicality and efficacy of the proposed framework.

In addition to contributing to the broader goal of improving UAV safety and operational efficiency, this research proves the applicability of machine learning models to MCUs with limited memory space and limited processing power used in embedded systems. With this research, it is likely to increase the applicability of machine learning models to embedded systems in automotive, defense, healthcare and other industrial fields.

2. TINY MACHINE LEARNING

In this section, the concepts of Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL) are briefly explained. The concept of Tiny Machine Learning will be mentioned. The contributions of TinyML and the challenges it will bring will be discussed.

Nowadays, AI, ML and DL concepts appear frequently. It is known that there are deep learning studies in various research fields such as natural language processing, speech recognition, medical applications, and computer vision. These research and developments are used in many areas such as Defense technologies, finance, industrial automation and health technologies [13].

The field of AI is concerned with not just understanding but also building intelligent entities, machines that can compute how to act effectively and safely in a wide variety of novel situations. AI includes different subfields, such as Machine Learning, which focuses on enhancing algorithms' performance through experience and data.

Machine Learning is a subset of AI that examines how to improve performance by utilizing experience and data. It revolves around creating models capable of learning from data to make forecasts or decisions. ML uses various techniques like supervised learning, unsupervised learning, and reinforcement learning [13], [14].

Deep Learning is a subset of ML and utilizes artificial neural networks with many layers (hence "deep") to learn from vast amounts of data. DL algorithms have achieved notable progress in domains like image recognition, speech recognition, and natural language processing by discerning complex patterns within data [13].

In this research, in-depth studies on Machine Learning have been carried out. In Figure 2.1, provide a comprehensive overview of the relationships between AI, ML, DL, and the different types of learning methodologies.

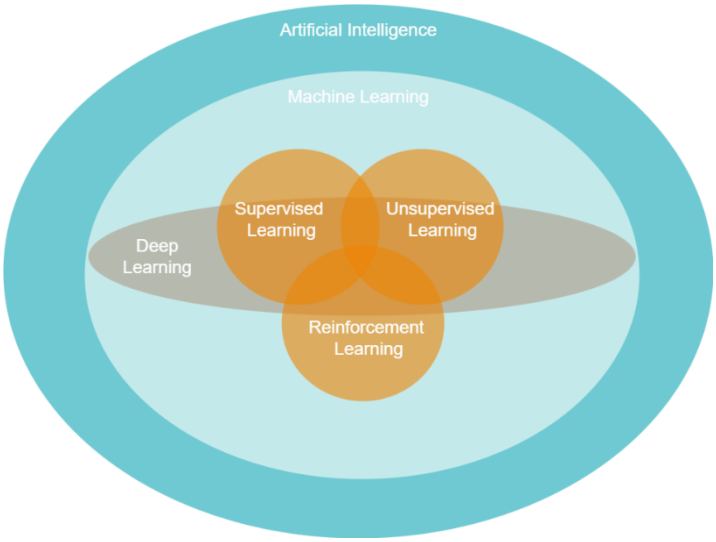

Figure 2.1 : Relationships between learning methodologies.

2.1 An Emerging Paradigm TinyML

Tiny Machine Learning combines Machine Learning power with Embedded Systems efficiency as shown in Figure 2.2, enabling machine learning algorithms to run on resource-constrained devices like microcontrollers. This fusion opens up opportunities for intelligent applications at the edge, defense industries.

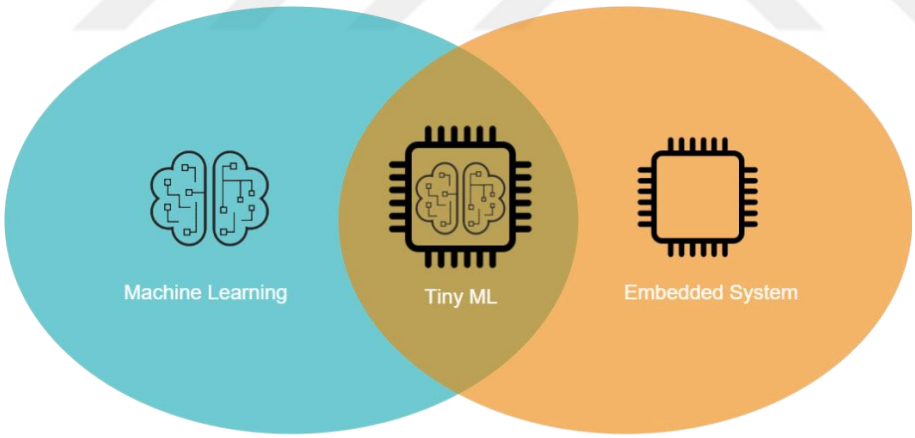

Figure 2.2 : TinyML diagram.

Tiny Machine Learning denotes the intersection between machine learning algorithms and extremely low-power embedded systems. It offers a platform for implementing smart applications on microcontrollers. By enabling devices to handle data processing on-site, TinyML decreases delays and reliance on cloud services, ultimately improving the effectiveness and speed of systems such as UAVs [1].

Specialized frameworks have contributed to the development of efficient ML models suitable for deployment on microcontrollers. CMSIS-NN is one of these frameworks,

consisting of effective neural network kernels that enhance the performance of machine learning algorithms on ARM Cortex-M CPUs. These kernels are specifically designed to enhance the execution of small-scale neural networks, which are crucial for the limited computational resources present on microcontrollers [5].

TinyML's capability to function on low-energy, limited-resource hardware has paved the way for its utilization in real-time systems. For instance, in UAVs, TinyML can process sensor data locally to swiftly identify unusual patterns or problems, allowing for immediate necessary actions crucial for flight safety and dependability. This feature is crucial in situations where transmitting data to a central server for analysis is not feasible due to latency or connectivity limitations [1], [15]. Nevertheless, integrating TinyML into microcontrollers presents several challenges. The primary issue is the limited memory and processing capabilities of these devices, which restrict the complexity of models that can be run. Additionally, power usage remains a notable concern, particularly for battery-operated embedded systems that must operate for long durations without being recharged.

Moving ahead, TinyML research is concentrating on several key areas: boosting model efficiency, decreasing power usage, and enhancing systems' adaptability to real-time learning. Researchers are investigating methods such as federated learning, enabling models to be trained directly on devices in a decentralized manner, thus improving performance and adaptability without compromising user privacy. Furthermore, advancements in hardware-specific optimizations and compiler technologies are poised to further lower the resource demands for running complex models on microcontrollers [1], [15], [16].

2.2 Technical Foundations of TinyML

2.2.1 Hardware considerations

When deploying TinyML on microcontrollers, especially for UAV systems that prioritize performance and reliability, it is important to take into account different hardware aspects. These aspects are vital in guaranteeing that the system fulfills operational needs and also sustains energy efficiency and the capacity to process real-time data efficiently.

2.2.1.1 Microcontroller selection

The choice of MCU is foundational in TinyML applications. ARM Cortex-M series MCUs are widely preferred due to their balance of power efficiency and processing capability. These MCUs feature low-power operation modes and efficient instruction sets that are optimized for computational tasks at minimal energy cost. The specific model selection, such as Cortex-M4 or Cortex-M7, depends on the required computational power, available memory, and peripheral integration necessary for the application [5].

In this thesis, the STM32H745 microcontroller has been selected as the core hardware platform for deploying TinyML applications in UAV systems. The STM32H745 is part of the STM32H7 series from STMicroelectronics, which is renowned for its high performance and efficiency, making it an ideal choice for real-time anomaly detection tasks.

Key features of STM32H745

- High Performance: The STM32H745 operates at a high frequency of up to 480 MHz, which is critical for processing complex algorithms quickly. This high clock speed allows the microcontroller to perform intricate machine learning computations required for detecting anomalies in real-time [17].
- Dual-Core Architecture: This microcontroller features a dual-core architecture with both an ARM Cortex-M7 and a Cortex-M4 core. This allows for flexible task management and improved efficiency, as tasks can be distributed between the cores based on their computational demands and power consumption profiles [17].
- Advanced Memory Features: It includes up to 2 MB of flash memory and up to 1 MB of RAM. These memory capacities are crucial for storing and executing machine learning models directly on the device without the need for frequent data transfers that can introduce latency [17].
- Rich Connectivity Options: The STM32H745 supports a variety of connectivity options, including USB, UART, CAN, SPI, and I2C interfaces, facilitating easy communication with peripheral sensors and other UAV

components. This connectivity is vital for integrating the microcontroller within a broader UAV system architecture [17].

- Robust Peripheral Support: Equipped with advanced analog peripherals and multiple timers, the STM32H745 is capable of handling multiple input streams from sensors simultaneously. This is essential for real-time monitoring and processing of various flight parameters and environmental data [17].

2.2.1.2 Memory constraints

Memory is a significant constraint in TinyML applications. MCUs typically have limited RAM and flash memory, which dictates the maximum size of the machine learning model that can be deployed. Techniques such as model pruning and quantization are essential to reduce the model size without significantly compromising its performance. Additionally, the use of CMSIS-NN library offers functions specifically optimized for neural networks, which help in managing memory usage more efficiently on ARM Cortex-M CPUs [5].

2.2.1.3 Power management

Efficient power utilization is crucial, especially for unmanned aerial vehicles powered by batteries that require long periods of operation. Microcontrollers such as the ARM Cortex-M series come equipped with functionalities like dynamic voltage scaling and power gating, which can effectively lower power usage during inactive phases or when maximum processing capacity is unnecessary. When incorporating TinyML into such devices, it is important to thoughtfully assess these capabilities to enhance power efficiency without compromising performance.

2.2.1.4 Real-time processing capabilities

TinyML deployments in UAVs demand real-time data processing to respond swiftly to environmental changes or system anomalies. This requires MCUs with sufficient clock speeds and real-time operating system (RTOS) support that can handle multitasking—running the ML model while simultaneously monitoring sensors and communicating with other system components. Ensuring the MCU and associated hardware can handle these tasks simultaneously is crucial for the successful deployment of TinyML in real-time applications.

2.2.1.5 Peripheral support

Integration with other system components, like sensors and communication modules, is another crucial hardware factor. The microcontroller unit should possess sufficient input/output peripherals and be capable of operating with communication protocols such as SPI, I2C, and UART. These protocols are essential for gathering sensor data and transmitting telemetry data. This ability to integrate ensures that the TinyML system can function smoothly within the wider UAV framework, handling sensor inputs and carrying out control actions in real-time.

2.2.2 Software considerations: optimization and model efficiency in TinyML

For effective deployment of TinyML on devices like the STM32H745, software optimization is critical. This involves not only making the best use of the microcontroller's capabilities but also ensuring that the machine learning models are as efficient as possible. Below are some key aspects of software optimization that need to be carefully managed. Optimizing a TinyML model involves several strategies to balance performance with the microcontroller’s limitations in memory and processing power.

2.2.2.1 Model pruning

Model pruning is a crucial optimization technique in the field of Tiny Machine Learning (TinyML), especially when deploying models on devices with limited computational resources, such as the STM32H745 microcontroller. Pruning involves the reduction of a neural network's complexity by removing weights or neurons that have minimal impact on the model's performance. This process helps in creating a more efficient model that consumes less memory and computational power, which is essential for real-time applications on microcontrollers.

According to Han et al. [18], pruning can significantly reduce the number of parameters in a neural network without substantial loss in accuracy (Figure 2.3). The authors introduce a method for pruning neural networks that involves three main steps: training the network, pruning the unnecessary connections, and fine-tuning the network to recover any accuracy lost during pruning. This method is known as "Learning both Weights and Connections for Efficient Neural Network" and has shown impressive results in creating compact and efficient models [18].

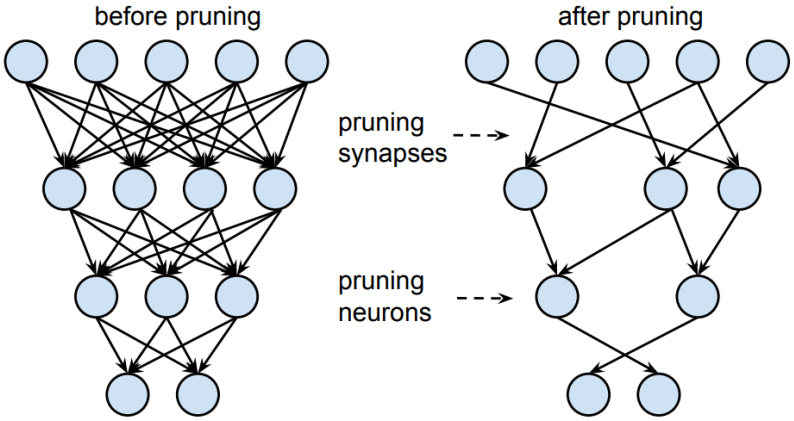

Figure 2.3 : Synapses and neurons before and after pruning [18].

Initially, the neural network is trained in the usual manner, using a large dataset to adjust the weights through backpropagation. The goal during this phase is to achieve high accuracy, ensuring that the network learns the relevant patterns in the data. This step is computationally intensive but necessary to establish a strong baseline model. Once the network is trained, the pruning process begins. Pruning identifies and removes weights that contribute minimally to the output. This is typically determined by setting a threshold; weights with magnitudes below this threshold are considered insignificant and are pruned as shown in Figure 2.4. The rationale behind this is that many weights in a large neural network have little effect on the final output and can be removed without significantly impacting the model's performance.

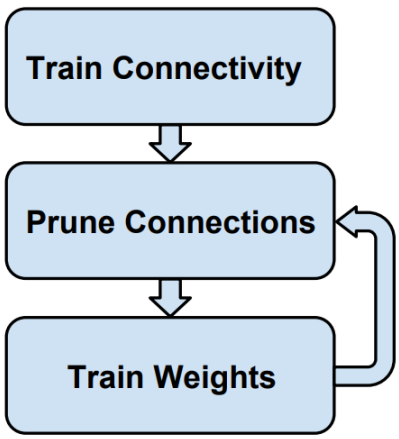

Figure 2.4 : Three-Step prune network training pipeline [18].

Han et al. [18], describe a specific pruning strategy where both individual weights and entire neurons can be pruned. They argue that pruning entire neurons can be particularly effective in convolutional neural networks (CNNs) where certain filters may become redundant after the initial training phase. By removing these redundant

filters, the overall size of the network is reduced, leading to faster inference times and lower energy consumption [18].

After pruning, the network often experiences a drop in accuracy. To address this, the pruned network undergoes a fine-tuning process. This involves retraining the network on the original dataset, allowing the remaining weights to adjust and compensate for the pruned connections. Fine-tuning is crucial as it helps in regaining the lost accuracy and stabilizing the network's performance.

In their experiments, Han et al. [18] demonstrate that fine-tuning can effectively restore the accuracy of a pruned network to levels comparable to the original, unpruned network. This is achieved with a significantly smaller number of parameters, resulting in a more efficient model that is well-suited for deployment on resource-constrained devices like microcontrollers [18]. One of the biggest limitations in deploying TinyML on the MCU is limited memory and processing power. With this approach, using the model pruning method can be used at the microcontroller level by preventing major distortions in the model parameters. In this way, model pruning becomes an important criterion for TinyML applications.

2.2.2.2 Quantization

Quantization involves reducing the precision of the numerical values used in the model, converting them from floating-point to fixed-point representations. This conversion significantly reduces the computational complexity and memory requirements of the model, making it more suitable for deployment on resource-constrained devices such as microcontrollers.

In the floating-point representation, numbers are stored with a high degree of precision, which necessitates substantial computational power and memory. In contrast, fixed-point representation uses fewer bits to represent numerical values, thus simplifying arithmetic operations and reducing both memory usage and computational load. This makes quantization particularly advantageous for embedded systems where resources are limited.

A key study in this area by Jacob et al. (2018) details methods for quantizing deep neural networks. The researchers explore techniques for converting 32-bit floating-point weights and activations to 8-bit integer representations. Their findings indicate that, with appropriate calibration and quantization-aware training, models can retain a

significant portion of their original accuracy while achieving substantial reductions in size and computational requirements [19].

The primary benefit of quantization is the dramatic decrease in model size. Smaller models are easier to store in the limited flash memory of microcontrollers and require less RAM during execution. This reduction not only speeds up the inference process but also allows for the deployment of more complex models that would otherwise be too large to fit within the constraints of the hardware.

However, quantization does introduce a trade-off: a slight decrease in model accuracy. This occurs because the reduced precision can cause a loss of information, particularly in deep learning models where fine-grained details can be critical. Despite this, the trade-off is often deemed acceptable because the gains in efficiency and speed are substantial. In many TinyML applications, the minor loss in accuracy is outweighed by the benefits of faster processing and lower power usage.

Advanced quantization techniques can mitigate some of the accuracy loss. For instance, techniques such as dynamic quantization, where weights and activations are quantized during runtime based on their distribution, and quantization-aware training, where the model is trained with quantization in mind, help in preserving the model's performance while reaping the benefits of reduced precision.

Implementing quantization on a microcontroller like the STM32H745 involves integrating the quantized model into the device’s software framework. The STM32H745, with its dual-core architecture and rich connectivity options, can effectively handle quantized models, enabling efficient real-time data processing and decision-making. CMSIS-NN kernels, specifically optimized for ARM Cortex-M processors, provide a suite of efficient implementations for neural network operations that are well-suited for handling quantized models. These kernels support various quantization schemes and significantly enhance the performance of TinyML applications on microcontrollers [5].

2.2.2.3 Model compression

Model compression is an essential part of Tiny Machine Learning (TinyML) that goes beyond just pruning and quantization. It includes different techniques to reduce the size of machine learning models while keeping their accuracy and performance intact. Effective model compression is vital for implementing machine learning models on

devices with limited resources like microcontrollers, where memory and processing capabilities are restricted.

One notable model compression method is knowledge distillation, where a smaller "student" model is trained to imitate the behavior of a larger, pre-trained "teacher" model. By mimicking the teacher model's output, the student model captures essential information in a more concise manner. This methodology, introduced by Hinton et al. (2015), has demonstrated a significant reduction in model size while maintaining high accuracy [20]. Knowledge distillation works by transferring knowledge from the teacher to the student through "soft labeling," where the student learns from the teacher's probability distributions rather than hard labels.

Another effective approach is weight clustering, which involves grouping neural network weights into clusters and sharing the same cluster centroid among multiple weights. This diminishes the number of unique weights that necessitate storage, resulting in a compact model representation. Weight clustering can be particularly advantageous for microcontrollers with limited storage capacity.

Furthermore, low-rank factorization is utilized for compressing models through the approximation of weight matrices with lower-rank matrices. This methodology reduces the quantity of parameters and computational complexity, enhancing the efficiency of the model for real-time applications on embedded devices Frequently utilized techniques such as singular value decomposition (SVD) are utilized for this objective [21].

Sparse representations are also integrated into model compression by enforcing sparsity in the network's weights, where many weights are set to zero. This leads to a more condensed model, demanding reduced memory and computational power for operation. Techniques like L1 regularization during training aid in achieving sparse representations.

While optimizing models for size and speed, it is essential to monitor the impact on accuracy. TinyML models must strike a balance between efficiency and effectiveness. Overly simplified models may fail to accurately detect anomalies, leading to critical errors in applications like UAV monitoring. Strategies like cross-validation during training can help evaluate the impact of optimizations on model performance, ensuring that the compressed model maintains its predictive capabilities.

Furthermore, the use of various compression techniques in combination could result in enhanced results For example, a model can be pruned to eliminate unnecessary weights, then quantized to reduce precision, and finally undergo knowledge distillation to further compress the model size. Each of these procedures contributes to augmenting the efficiency of the model while upholding its precision.

2.2.2.4 Utilization of hardware acceleration

Integration with other system components, such as sensors and communication modules, is another critical consideration. However, the control of these communication modules must be successfully created on a software-based basis in addition to the hardware infrastructure. The MCU must have adequate input/output peripherals and support for communication protocols like SPI, I2C, and UART, which are necessary for sensor data acquisition and telemetry data transmission. This integration capability ensures that the TinyML system can operate seamlessly within the broader UAV architecture, processing sensor inputs and executing control actions in real time. Integration with other system components, such as sensors and communication modules, is another critical hardware consideration. The MCU must have adequate input/output peripherals and support for communication protocols like SPI, I2C, and UART, which are necessary for sensor data acquisition and telemetry data transmission. This integration capability ensures that the TinyML system can operate seamlessly within the broader UAV architecture, processing sensor inputs and executing control actions in real time.

3. ANOMALY DETECTION USING MACHINE LEARNING

3.1 Identifying Anomalies

Identifying anomalies, often referred to as outlier detection, is essential in numerous domains such as defense technology, cybersecurity, finance, healthcare, manufacturing, and engineering [22], [23]. Identifying patterns in data that deviate from anticipated behavior is part of the process. Anomalies can indicate significant but rare events such as security breaches, equipment malfunctions, or unauthorized activities. In defense applications involving UAVs, anomalies could consist of unforeseen changes in flight paths, atypical sensor data, or abnormal system performance, indicating possible malfunctions, hostile interventions, or cyber threats.

Several factors can contribute to anomalies in UAV systems. These include hardware malfunctions, software bugs, environmental interferences, and intentional electronic warfare tactics. Detecting anomalies in real-time is imperative to uphold the security, dependability, and accomplishment of UAV operations. Conventional techniques for anomaly detection frequently demand substantial computational capabilities, rendering their application infeasible for utilization on resource-limited equipment like microcontroller units integrated within UAV frameworks. This is where TinyML becomes particularly valuable. TinyML denotes the implementation of machine learning models on devices that have low-power and resource constraints, facilitating instantaneous inference and decision-making at the periphery. TinyML capitalizes on progress in hardware effectiveness and software enhancement to execute complex machine learning algorithms on diminutive devices with restricted computational capabilities and storage capacity. This approach not only reduces the dependency on continuous connectivity to central servers but also enhances the responsiveness and autonomy of edge devices like UAVs.

TinyML is especially well-suited for anomaly detection in defense-oriented UAV systems owing to its capacity to locally process and scrutinize data, thus reducing latency and bandwidth consumption. Through the integration of machine learning models directly into the hardware of the UAV, TinyML facilitates instantaneous

surveillance and identification of anomalies, a crucial aspect for prompt interventions and safeguarding the vehicle's safety and operational integrity.

The capability of TinyML to transform the implementation of machine learning models on devices with limited resources has been extensively documented. According to Ray, "TinyML facilitates running machine learning at the embedded edge devices having very little processor and memory" [15]. This shift in paradigm enables the creation of intelligent systems that function effectively within the constrained resources present in edge devices. Furthermore, the use of specialized libraries like CMSIS-NN enhances the performance and reduces the memory footprint of neural network applications on ARM Cortex-M processors [5].

Overall, this section delves into the intricacies of anomaly detection, the potential causes of anomalies, and the innovative approach of TinyML in addressing these challenges. The integration of TinyML in UAV systems not only enhances their anomaly detection capabilities but also contributes to the broader goal of improving operational efficiency and security in various real-time defense applications.

3.1.1 Types of anomalies

Anomalies, alternatively referred to as outliers, represent data points or patterns that boldly stray away from the usual within a collection of data. These deviations can signal critical events. According to Chandola, Banerjee, and Kumar (2009), anomaly detection is vital for recognizing unusual patterns in data that diverge from expected behavior, aiding in the early identification of potential threats and issues [22].

Ghamry et al. (2024) further underscore the significance of anomaly detection across various domains, highlighting its crucial role in maintaining system reliability and safety [23].

Anomalies can be categorized into three primary groups [22], [23]:

- Point-based Anomaly
- Collective Anomaly
- Contextual Anomaly

3.1.1.1 Point-based anomaly

A specific data instance that contains anomalous information. This form of anomaly pertains to a singular data point that displays a substantial deviation from the remaining data points.

In Figure 3.1, normal data points are shown with blue circles. The data point at index 50 is highlighted in red, indicating it deviates significantly from the rest. This form of anomaly is identified as a point-based anomaly, in which a particular data point is distinguished as deviant. For instance, in UAV sensor data, a sudden spike in value might indicate a sensor malfunction or external interference. Detecting such anomalies is crucial for early identification of security breaches or equipment failures.

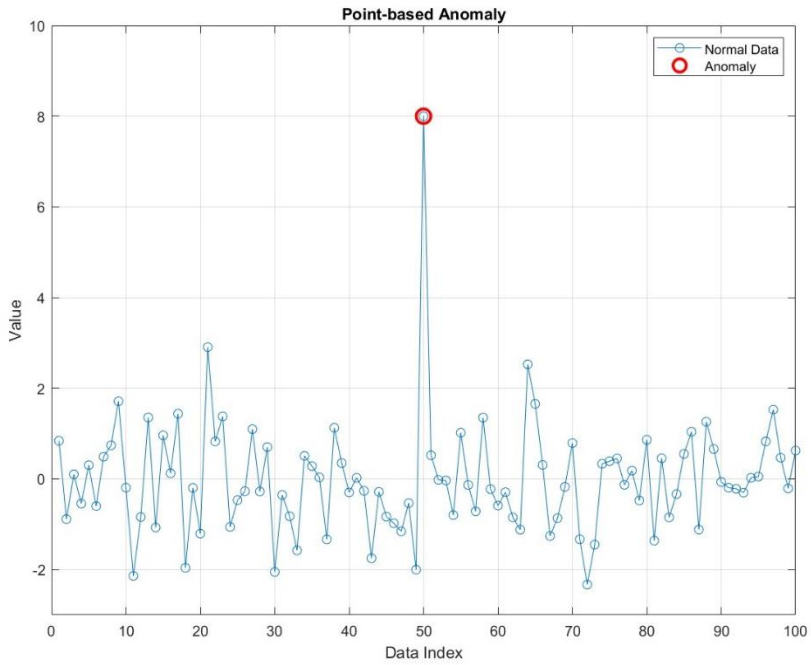

Figure 3.1 : Point-based anomaly.

3.1.1.2 Collective anomaly

A set of records comprising abnormal data in comparison to the complete dataset is observed. The individual data points within a collective anomaly might not qualify as outliers on their own, yet their combined presence is deemed anomalous. This is especially pertinent in identifying patterns or sequences that diverge from anticipated behavior.

In Figure 3.2, normal data points are shown with blue circles. The data points from index 40 to 50 are highlighted in red, indicating a collective anomaly. Although

individual data points within this range may not appear anomalous on their own, their occurrence together as a group deviates from the expected pattern. For example, in UAV operations, a group of sensor readings might indicate a pattern of failure or external interference that needs to be addressed.

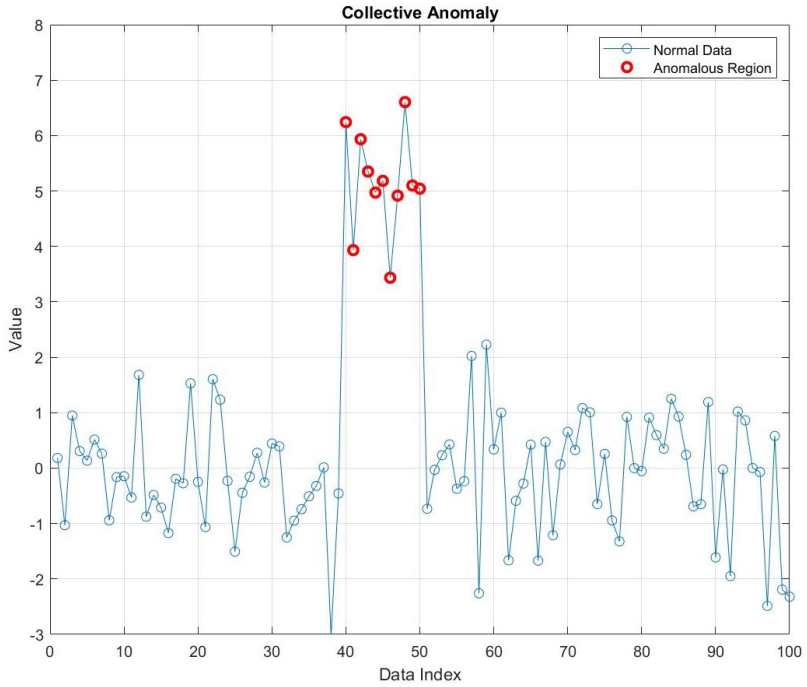

Figure 3.2 : Collective anomaly.

3.1.1.3 Contextual anomaly

The abnormality depends on the context of the data generation. In alternative terminology, a data point could be deemed typical in one scenario but atypical in another. Anomalies within a specific context are frequently observed in temporal or spatial data, where factors such as time or location are pivotal in delineating standard behavior. To illustrate, within the domain of defense-oriented UAV systems, anomalies may encompass unforeseen deviations in flight trajectories, atypical sensor data, or irregular system operations, signaling potential malfunctions, hostile interventions, or cyber assaults. These anomalies can arise due to hardware malfunctions, software bugs, environmental interferences, or intentional electronic warfare tactics.

In Figure 3.3, normal data points follow a sinusoidal pattern and are shown with blue circles. The data points from index 70 to 80 are highlighted in red, indicating a contextual anomaly. While these points may not appear anomalous in a different context, within the expected sinusoidal pattern, they clearly deviate from the norm.

Contextual anomalies are crucial for detecting issues that are context-dependent, such as seasonal changes in sensor readings or location-based anomalies in spatial data.

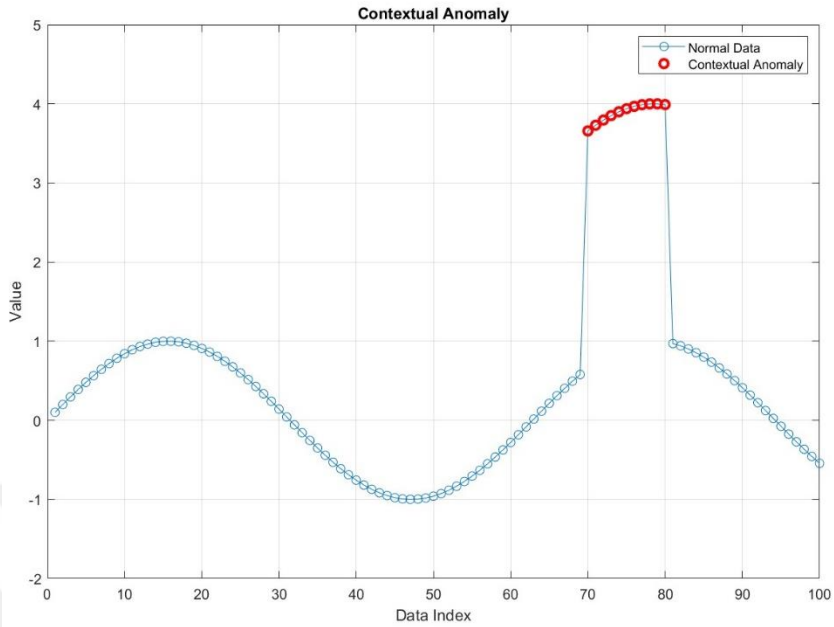

Figure 3.3 : Contextual anomaly.

3.2 Evaluation of Flight-Critical Parameters in UAV Systems

The use of drones is expanding in many essential fields, including defense monitoring and business parcel shipping. When UAVs function without human intervention or with limited human oversight, it becomes crucial to prioritize the safety and dependability of their activities. To uphold flight stability and avert mission failure, it is imperative to consistently monitor an array of flight-critical parameters, including Global Positioning System (GPS) signal integrity, engine performance metrics, battery status, and environmental variables such as altitude and wind conditions.

Flight-critical parameters are intrinsically associated with the UAV's capacity to sustain controlled navigation and fulfill its operational objectives. Any shift or irregularity in these indicators might compromise the UAV's operational stability, bringing about major threats like loss of command, systemic collapse, or tragic incidents. By utilizing sophisticated machine learning methodologies, these parameters can be scrutinized in real-time, facilitating the early identification of anomalies that may signify impending malfunctions or environmental threats. A proficient anomaly detection system is pivotal in recognizing such irregularities prior to their escalation into critical failures, thereby safeguarding both flight safety and

mission accomplishment. The assessment of these parameters is not merely essential for ensuring operational safety but also for the enhancement of UAV performance. Through the utilization of machine learning algorithms, UAV systems are proficient in autonomously adapting to environmental changes, detecting irregular behaviors, and implementing remedial actions when considered necessary. This adaptive methodology enables UAVs to function with greater efficacy in intricate environments, thereby augmenting their capacity to fulfill critical missions amidst challenging circumstances. To guarantee dependable and secure operation, it is imperative that several essential parameters be meticulously monitored in real-time. These parameters encompass GPS signal integrity and navigation precision, engine and propulsion system efficacy, battery condition, power management strategies, as well as environmental variables such as altitude and wind conditions. Each of these parameters critical to flight operations is instrumental in ascertaining the UAV's stability, operational performance, energy efficiency, and capability to fulfill its mission without encountering failure. In the subsequent sections, we shall investigate these essential parameters and examine their importance in preserving flight safety and enhancing UAV performance.

3.2.1 GPS signal and navigation

The Global Positioning System (GPS) is crucial in the realm of Unmanned Aerial Vehicles as it supplies dependable locational insights and promotes exact navigation abilities. The signals emitted by GPS are vital for ascertaining the UAV's geographical coordinates, speed, and altitude, thus enabling the system to adhere to predetermined flight trajectories and perform autonomous missions. Any deterioration in the strength of GPS signals or the precision of positioning can significantly compromise the UAV's navigational efficacy, heightening the likelihood of departure from the designated route, operational failure, or even total loss of control.

The quality of GPS signals may be affected by a multitude of external variables, including interference from adjacent structures, environmental factors such as dense cloud cover or rugged terrain, and multipath phenomena where signals reflect off various surfaces. Such disruptions may induce delays or inaccuracies in positional data, culminating in navigational discrepancies. To alleviate these hazards, UAV systems depend on the real-time assessment of GPS signal strength and data integrity.

The incorporation of machine learning algorithms is frequently employed to promptly identify these anomalies and implement corrective measures, such as transitioning to redundant navigation systems or recalibrating flight paths [24].

Furthermore, the accuracy of positioning is intrinsically associated with the success of missions, particularly in contexts such as parcel delivery, cartography, and surveillance operations. Minor discrepancies in positioning can precipitate substantial errors in these endeavors, thereby diminishing the operational effectiveness of the UAV. By perpetually scrutinizing the signal-to-noise ratio (SNR) and evaluating positional drift, the system is capable of detecting potential GPS anomalies prior to their escalation into mission-critical failures [25].

In recent developments, multi-GNSS (Global Navigation Satellite Systems) integration, which includes systems like GLONASS and Galileo alongside GPS, has been employed to improve positional accuracy and reliability. This multi-constellation approach reduces the likelihood of signal loss, enhancing the UAV's ability to operate in challenging environments [26]. Such advancements, combined with advanced anomaly detection techniques, ensure that UAVs maintain reliable and accurate navigation even under adverse conditions.

3.2.2 Engine and propulsion system performance

In UAV systems, the propulsion and engine components play a vital role in upholding flight stability while guaranteeing successful mission performance. These systems function under a variety of conditions, including fluctuations in altitude, temperature, and power requirements, all of which could precipitate performance degradation if not consistently overseen. The prompt identification of anomalies within these systems is essential in order to avert catastrophic failures during flight operations.

A pivotal parameter for oversight is engine temperature, as excessive heat may signify inadequate cooling, mechanical deterioration, or inefficient combustion processes. By employing real-time monitoring systems that are augmented with machine learning algorithms, UAVs are capable of detecting slight temperature fluctuations prior to their escalation into critical complications, thereby averting unanticipated failures and facilitating proactive maintenance [27]. Ongoing temperature surveillance can also assist in identifying engine degradation and systemic inefficiencies, thereby contributing to the formulation of more precise maintenance schedules.

Thrust generation and power output are also key indicators of propulsion system health. Variations in thrust can result in flight instability, particularly during takeoff or high-altitude operations. Machine learning models designed to detect anomalies in thrust data can alert operators to potential mechanical issues, such as motor degradation or imbalances in the propulsion system. These predictive maintenance techniques have proven effective in ensuring consistent flight performance [28]. Anomalies in thrust generation can lead to imbalanced propulsion, which affects the UAV's ability to maintain its flight trajectory under challenging conditions [29].

Fuel consumption represents a significant performance metric, particularly in the context of extended-duration missions. Irregularities in fuel consumption trends, such as abrupt increases, may indicate underlying problems such as fuel leakage or suboptimal combustion processes. With the persistent examination of real-time fuel data, Unmanned Aerial Vehicle systems are adept at maximizing fuel consumption and minimizing mission-critical failures. Methodologies such as fuel flow modeling have been established to forecast fuel consumption patterns and identify inefficiencies [28]. This optimization is imperative for enhancing the operational range and endurance of UAV missions [30]. Furthermore, the incorporation of health monitoring systems that employ data fusion methodologies can significantly improve the identification of performance anomalies. By synthesizing data from various sensors—including temperature, thrust, and fuel consumption—UAVs can more accurately forecast potential failures and proactively address them [31]. This comprehensive approach serves to enhance both the reliability and longevity of UAV propulsion systems [32]. Additionally, the analysis of vibration and noise is essential in the assessment of propulsion system performance. Unusual vibration patterns may signify mechanical problems such as motor imbalance or misalignment, while variations in noise levels can indicate deterioration in bearings or defects in propellers. Continuous monitoring of these parameters enables UAVs to avert minor issues from escalating into critical failures [33]. The implementation of such monitoring techniques ensures that the propulsion system maintains stability even under fluctuating operational conditions [34].

Through the utilization of complex anomaly detection methodologies, notably those that harness machine learning and neural networks, UAV technology is equipped to present instant diagnostics and optimize performance capabilities. This capability

enables UAVs to sustain operational efficacy, even within adverse environments, by concurrently analyzing a multitude of parameters and forecasting potential malfunctions prior to their manifestation [35].

3.2.3 Battery health and power distribution management

In UAV systems, maintaining the health of the battery and ensuring efficient power distribution are critical for flight safety and mission success. The battery functions as the primary energy reservoir for propulsion, avionics, and communication systems, thereby necessitating the continuous monitoring of critical parameters such as charge levels, battery condition, and power consumption trends. Any discrepancies within these parameters can result in performance deterioration or potentially catastrophic failures during operation.

UAV applications frequently rely on Lithium-ion (Li-ion) batteries, known for their impressive energy density and durability, which can, however, weaken with time. This impairment, defined by a decrease in capacity, escalated internal resistance, and a heightened likelihood of thermal runaway, underscores the essential requirement for immediate monitoring of the State of Charge (SoC) and State of Health (SoH). Maintaining equilibrium in battery charge levels is vital for averting uneven power distribution and abrupt power losses that could compromise flight stability. Routine assessments of these levels, in conjunction with advanced battery management systems (BMS), ensure that batteries function within designated safety parameters and yield precise estimations of SoC and SoH [36].

High power consumption events, particularly during phases of takeoff, rapid maneuvers, or under challenging meteorological conditions, exert considerable strain on the power system. In situations where effective anomaly detection and power management frameworks are missing, UAVs might face the danger of substantial power loss incidents, resulting in crashes or operational issues. These circumstances underscore the critical necessity of identifying and mitigating overcurrent events and voltage fluctuations that can compromise the power supply to essential subsystems [12], [36]. Additionally, as UAV systems evolve towards more intelligent and autonomous architectures, the need for dynamic anomaly detection structures becomes increasingly important. Traditional anomaly detection methods may struggle to keep up with the complex, real-time data processing demands of modern UAV systems.

Therefore, integrating machine learning models and adaptive algorithms capable of learning from the UAV's operational data will enable these systems to detect more subtle and evolving patterns of anomalies. These dynamic anomaly detection systems will not only help prevent catastrophic failures but also allow UAVs to optimize their performance by learning and adapting to different flight conditions, environmental factors, and power consumption patterns [37] .

3.2.4 Enviromental factors

The operational success and reliability of UAV systems are significantly influenced by environmental variables. Factors such as wind, fluctuations in temperature, humidity levels, precipitation, and variations in atmospheric pressure can pose challenges that affect essential components, including the propulsion system, battery integrity, and sensor precision. Effectively mitigating these environmental factors is crucial for ensuring consistent flight operations and circumventing potential malfunctions. Wind variations significantly affect UAVs while airborne, particularly in critical moments such as rising, falling, and when executing quick aerial moves. Wind gusts have the potential to disrupt the flight trajectory, necessitating increased effort from the propulsion system to uphold the UAV's designated path. This augmented energy expenditure has a direct correlation with battery longevity and operational duration, thereby necessitating real-time modifications to prevent power exhaustion.

Wind forces, particularly horizontal winds, interfere with the aerodynamics of UAVs by creating turbulence around the rotor blades. As shown in recent aerodynamic studies, wind speeds between 2-4 m/s can cause a noticeable increase in thrust, but at the cost of higher power consumption due to rotor interference. Computational Fluid Dynamics (CFD) simulations and wind tunnel experiments on an orthogonal octorotor UAV demonstrated that while low wind speeds can improve thrust efficiency, higher wind speeds lead to complex airflow interactions that diminish overall energy efficiency [38].

Furthermore, UAV swarms, operating in dynamic environments with wind and obstacle disturbances, must remain resilient to these factors to continue functioning as expected. Modeling the effects of wind disturbances in UAV swarm simulations has revealed that these environmental challenges can lead to interruptions in mission

progress and hinder overall performance. Wind disturbances, in particular, were identified as one of the primary disruptors, causing UAVs to expend more energy to maintain flight stability [39].

Humidity and precipitation can significantly affect the operational efficiency and safety of Unmanned Aerial Vehicles. Elevated humidity levels, particularly in tropical or coastal regions, heighten the likelihood of corrosion and short circuits within UAV electronic systems. Moisture can accumulate on critical components, including motors, sensors, and flight control systems, resulting in malfunctions and a reduction in reliability. Over an extended period, this accumulation may compromise the integrity of electrical connections and circuits, leading to power losses or sporadic failures. Implementing preventive measures, such as waterproof coatings and routine maintenance inspections, is essential to mitigate these risks.

Precipitation, such as rain or snow, poses immediate threats to UAV operations. When exposed to rain, UAVs risk water ingress into critical components, including propulsion systems and cameras, which can lead to failure during flight. Ice accumulation, especially on propellers, can disrupt aerodynamics, increase drag, and result in unstable flight behavior. UAVs are generally not designed to operate in heavy precipitation, and any sign of rain or snow should prompt immediate action to land the UAV. Pilots need to monitor weather conditions closely before and during flight to prevent these issues [40], [41]. Furthermore, the accumulation of humidity over time can have long-term effects on UAVs, including the gradual deterioration of materials and an increase in the likelihood of electrical shocks and circuit failures. Prolonged exposure to high-humidity environments without adequate dehumidification and ventilation can lead to significant repair costs and reduced operational lifespans [40], [41].

Atmospheric pressure and altitude play a crucial role in determining the operational effectiveness and energy utilization of UAVs. The ascent to higher altitudes correlates with a decline in air density, which adversely impacts lift and demands that the UAV's propulsion system dedicates further effort to maintain its altitude. This augmented requirement for thrust compels the UAV to utilize greater amounts of power, thereby potentially reducing flight durations, particularly in UAVs powered by batteries. At elevated altitudes, UAVs also encounter diminished cooling proficiency, as the rarified atmosphere is less capable of effectively dissipating thermal energy from the motors

and electronic components. These obstacles underscore the imperative for real-time assessment of the UAV's power utilization and thrust generation to guarantee consistent performance amidst fluctuating atmospheric conditions [42], [43].

Additionally, empirical investigations have indicated that Unmanned Aerial Vehicles functioning at elevations exceeding 3,000 meters encounter a significant reduction in aerodynamic efficacy. At such altitudes, the propulsion mechanisms of UAVs are compelled to mitigate the decline in lift by either augmenting rotor velocities or modifying flight trajectories, which in turn imposes additional strain on the battery and diminishes operational longevity. High-altitude UAVs, particularly those utilizing hybrid power systems such as photovoltaic cells in conjunction with batteries, are necessitated to implement sophisticated power management methodologies to optimize energy efficiency while considering altitude-induced power deficiencies [42], [44].

Unmanned Aerial Vehicles engineered for elevated flight altitudes frequently integrate specialized elements, including lightweight composites and high-performance propulsion systems, aimed at reducing energy expenditure. Through the dynamic modulation of flight parameters, including velocity and elevation, predicated on real-time insights derived from environmental sensors, UAVs are capable of optimizing their energy utilization, thereby enhancing their operational reach within high-altitude settings [44].

Solar radiation and Electromagnetic Interference (EMI) are pivotal factors that can markedly diminish the operational performance of unmanned aerial vehicles, especially when these systems are utilized at significant altitudes or near sources of electromagnetic emissions. The presence of solar radiation, particularly at increased altitudes, can precipitate long-term material degradation in UAVs, thereby compromising their structural integrity progressively over time. Extended contact with ultraviolet (UV) radiation can compromise the integrity of the UAV’s composite materials, ultimately affecting the operational longevity of critical parts like the airframe and rotor blades. Additionally, solar radiation has the capability to disrupt the functional performance of onboard electronics, which may result in the malfunction of essential systems such as GPS and communication modules. In order to mitigate these adverse effects, UAVs that operate under conditions of high solar intensity frequently

necessitate the application of protective coatings and designs that facilitate heat dissipation, in order to avert the overheating of sensitive components.

EMI, conversely, constitutes a more pressing hazard to the operational efficacy of UAVs. EMI can originate from various external entities, such as high-voltage power transmission lines or telecommunication towers, as well as from internal constituents, namely the UAV's own electronic apparatus. Elevated electromagnetic field exposure can possibly disturb the UAV's communication framework, which in turn affects the signal integrity with its related ground control setup. In extreme scenarios, substantial EMI can adversely affect flight control mechanisms, thereby engendering a potential loss of control or unpredictable behavior during flight operations. UAVs that function in proximity to Extra High Voltage (EHV) power facilities are notably susceptible to such electromagnetic interference. Empirical research has indicated that UAVs operating in the vicinity of EHV power lines experience significant levels of electromagnetic noise, which can detrimentally influence their hardware and communication channels [45], [46].

To handle these potential issues, unmanned aerial systems should be equipped with electromagnetic shields and secondary communication frameworks to secure uninterrupted performance in settings marked by intense electromagnetic presence. Current investigations concentrate on augmenting the Electromagnetic Compatibility (EMC) of UAVs, thereby enhancing their resilience to both solar radiation and EMI through sophisticated design methodologies and innovative material applications [45], [46].

3.3 Dataset Overview and Relevance

The ALFA (AirLab Failure and Anomaly) dataset is a comprehensive collection designed for research in Fault Detection and Isolation (FDI) and Anomaly Detection (AD) in UAVs. This dataset was introduced by Keipour, Mousaei, and Scherer in their 2021 publication [36] and is hosted on the AirLab website [47]. The data presented encompasses a wide range of flight scenarios, encompassing both standard and erroneous operations, thus serving as a crucial asset in the advancement of reliable anomaly detection mechanisms.

3.3.1 Sources and collection process

The ALFA dataset was collected from several UAVs during controlled flight tests, primarily focusing on a fixed-wing UAV platform, the Carbon-Z T-28. This UAV, equipped with a Holybro PX4 2.4.6 autopilot and an Nvidia Jetson TX2 onboard computer as shown in Figure 3.4, was used to gather high-frequency sensor data. The tests were conducted at an airport near Pittsburgh, Pennsylvania, ensuring a controlled environment for reliable data collection [36] [24], [47].

The primary platform for data collection was the Carbon-Z T-28 as shown in Figure 3.4, a fixed-wing UAV with a wingspan of 2 meters and a single electric engine. It was equipped with additional modules such as a Pitot Tube, GPS, and various sensors to monitor flight dynamics.

Figure 3.4 : The Carbon-Z T-28 fixed-wing UAV platform [24].

The dataset includes a wide range of flight parameters captured through various sensors as shown in Table 3.1.

Table 3.1 : ALFA dataset parameters and descriptions.

Parameter	Description
Sampling Rate	Data is recorded at high frequency (up to 100 Hz), ensuring detailed capture of flight dynamics.
Sensor Types	Includes accelerometers, gyroscopes, magnetometers, GPS, barometers, and temperature sensors.
Flight Data	Covers altitude, speed, orientation, battery status, and control inputs.
Fault Types	Simulated faults include motors failures, sensor malfunctions, and GPS signal loss.
Data Volume	The dataset several gigabytes of data, structured into time-series records.

3.3.2 Fault scenarios

The ALFA dataset includes various fault scenarios, each precisely documented with ground truth labels. These scenarios encompass sudden control surface faults, engine failures, and other critical failures. Table 3.2 summarizes the types of errors and their occurrence in the dataset [47].

Table 3.2 : ALFA dataset types and occurences of faults.

Fault Type	Number of Test Cases	Flight Time Before Fault (s)	Flight Time with Fault (s)
Engine Full Power Loss	23	2282	362
Rudder Stuck to Left	1	60	9
Rudder Stuck to Right	2	107	32
Elevator Stuck at Zero	2	181	23
Left Aileron Stuck at Zero	3	228	183
Right Aileron Stuck at Zero	4	442	231
Both Ailerons Stuck at Zero	1	66	36
Rudder & Aileron at Zero	1	116	27
No Fault	10	558	-
Total	**47**	**3935**	**777**

This dataset was chosen for its comprehensive and high-frequency recordings, which are essential for detecting subtle anomalies that could indicate potential safety risks. The dataset's detailed recordings enable the development of robust machine learning models capable of real-time anomaly detection. By leveraging the ALFA dataset, researchers can simulate and analyze various fault conditions, enhancing the resilience and reliability of UAV operations.

3.4 Machine Learning Approach to Anomaly Detection: Autoencoders

Autoencoders have emerged as a significant methodology in the domain of anomaly detection, attributable to their proficiency in encapsulating efficient representations of data while concurrently minimizing reconstruction error. In unsupervised learning, autoencoders reduce high-dimensional input data into a lower-dimensional latent space and then attempt to reconstruct the original data from this compressed form. Subsequently, the model endeavors to reconstruct the original data from this compressed representation. The key aim is to lessen the reconstruction error, reflecting the discrepancy between the input and the recreated output. This characteristic renders autoencoders particularly advantageous for anomaly detection, as anomalous data

typically yields greater reconstruction errors in comparison to normative data. In the area of UAVs, this strategy is notably beneficial for recognizing atypical sensor values, like shifts in altitude or speed, which might reveal likely system failures.

Structurally, an autoencoder consists of two main components: the encoder, which compresses the input data into a condensed representation, and the decoder, which reconstructs the original input from this compressed form as shown in Figure 3.5. During the training phase, the model learns to minimize reconstruction error, typically using loss functions such as Mean Squared Error (MSE).

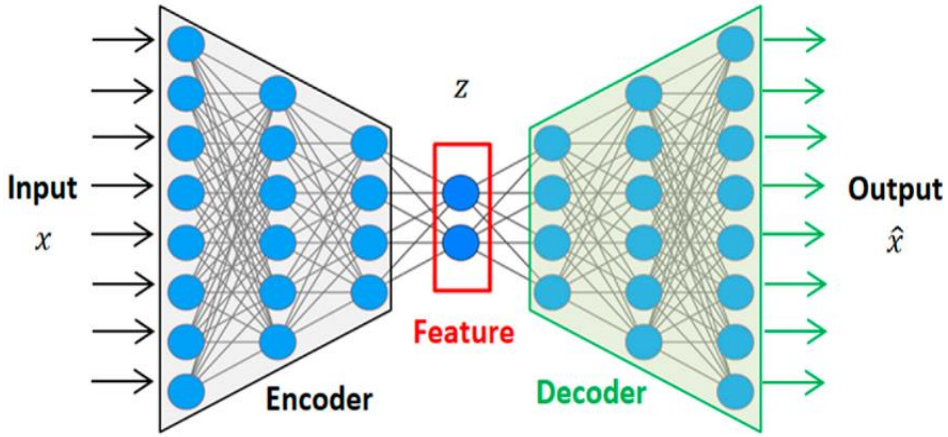

Figure 3.5 : Autoencoder general overview [48].

Autoencoders have demonstrated considerable success in anomaly detection, especially when trained on normal (non-anomalous) data. Throughout the training process, the model becomes proficient at reconstructing standard data. Consequently, when the autoencoder encounters anomalous data, the reconstruction error tends to increase, as the model has not been trained to accurately reproduce abnormal patterns. This property establishes autoencoders as a reliable tool for identifying deviations from expected patterns in datasets [49].

The training process typically involves collecting a dataset composed of normal operational data, preprocessing the data to ensure consistency, and then training the autoencoder to minimize reconstruction error. Once trained, the model is evaluated on a separate validation set that includes both normal and anomalous data. The reconstruction error is computed for each data point, and a threshold is set to classify the points as normal or anomalous.. In the context of UAVs, autoencoders are highly effective in detecting anomalies within flight data. For instance, the ALFA dataset, which contains diverse sensor readings from UAV flights, can be used to train an

autoencoder. By analyzing the reconstruction error, the autoencoder can detect anomalies that may indicate potential malfunctions or failures in the UAV's operation. The ALFA dataset provides high-frequency sensor data, including parameters such as altitude, speed, orientation, and battery status, which are critical for accurate anomaly detection [36] , [47]. Autoencoders present several advantages for anomaly detection. First, they effectively reduce the dimensionality of the input data, allowing the model to compress large datasets while retaining essential features. Second, autoencoders do not require labeled data for training, making them ideal for scenarios where labeled data is scarce or unavailable. Third, autoencoders can be scaled to handle large datasets and complex data structures, providing flexibility and robustness across a range of applications [49]. By leveraging the strengths of autoencoders, researchers can develop highly efficient anomaly detection systems that can perform real-time monitoring and fault detection, thus ensuring the reliability and safety of UAV operations by rapidly identifying and addressing potential issues.

3.4.1 Advantages of autoencoders in anomaly detection

Autoencoders possess several significant advantages in the domain of anomaly detection, rendering them an effective instrument for the identification of irregularities. A primary advantage is their capacity to operate without the necessity of labeled data. During the training phase, autoencoders are capable of assimilating the normal patterns of data, thereby facilitating the detection of deviations that may indicate potential anomalies. This capability is especially advantageous in scenarios where the acquisition of labeled data is rendered impractical, such as in real-time monitoring or industrial contexts. By learning directly from the data, autoencoders exhibit adaptability to a variety of anomaly detection challenges with minimal oversight [49]. Another critical advantage lies in their proficiency in diminishing the dimensionality of extensive datasets. Autoencoders facilitate the compression of high-dimensional datasets into a more concise representation, adeptly encapsulating the most prominent features while eliminating superfluous noise. This attribute renders them particularly suited for managing complex data in scenarios where efficiency and speed are paramount, such as in sensor networks or unmanned aerial vehicle systems. The reduction in data complexity not only enhances processing speed but also facilitates real-time anomaly detection in environments characterized by high throughput.

Furthermore, autoencoders possess a distinguished capability to pinpoint slight anomalies that may evade ordinary detection systems. By juxtaposing the reconstructed output against the original dataset, any notable discrepancies are identified as anomalies. This heightened sensitivity to minor variations is paramount in recognizing early indicators of potential system failures or dysfunctions, thereby facilitating prompt interventions prior to the escalation of more significant issues. Owing to their adaptability and efficacy, autoencoders have emerged as a preferred option for numerous sectors that depend on continuous surveillance and anomaly identification. Their proficiency in functioning with minimal oversight, efficiently processing extensive datasets, and recognizing even the most minute irregularities renders them a robust and scalable solution for the detection of anomalies across diverse applications.

3.4.2 Limitations and challenges of autoencoders

Autoencoders offer several advantages for anomaly detection; however, they also have certain limitations that can impact their performance, particularly in embedded systems and TinyML applications. A significant challenge encountered is overfitting, which transpires when the model undergoes excessive training on the training dataset. In these cases, the autoencoder manifests a significant expertise in reproducing the specific patterns it has internalized, consequently obstructing its skill to generalize to unfamiliar, unseen data. This phenomenon can result in the oversight of genuine anomalies (false negatives), as the model encounters difficulties in identifying deviations from its training dataset. To prevent overfitting, techniques such as regularization, dropout, and early stopping are commonly used to improve the model's ability to generalize.

In the realm of embedded systems, the phenomenon of overfitting presents significant challenges owing to the constrained computational resources at hand. Devices like microcontrollers impose stringent limitations regarding memory capacity and processing capabilities, thereby constraining the dimensionality and intricacy of the models that may be implemented. To mitigate this issue, the application of model optimization methodologies, including pruning and quantization, is essential to guarantee that autoencoders operate effectively on embedded hardware while maintaining performance integrity.

Another constraint is the deficiency in interpretability. Autoencoders identify anomalies through an assessment of the reconstruction error; however, they fail to furnish explicit justifications as to why a specific data point is deemed anomalous. This issue can be particularly concerning in critical sectors such as healthcare or autonomous systems, where comprehending the rationale behind a model's inference is of paramount importance. In embedded systems, this dilemma is further intensified by limited computational resources, making the implementation of methods like Explainable AI (XAI) more challenging, as these approaches aim to enhance the clarity and interpretability of the decision-making process [50].

Autoencoders may exhibit limitations in the detection of nuanced anomalies, particularly when utilized in resource-constrained embedded systems. In instances wherein the disparities between normative and anomalous datasets are minimal, the model may lack the requisite sensitivity to effectively differentiate between the two. This deficiency can lead to both false positives and false negatives, a situation that is particularly alarming in domains such as cybersecurity or IoT sensor networks, where even slight deviations could indicate considerable threats. Enhancing the model’s sensitivity to these subtle distinctions frequently necessitates the integration of autoencoders with complementary methodologies, such as recurrent neural networks (RNNs) or attention mechanisms; however, the computational implications of these approaches must be judiciously addressed when functioning on embedded hardware [50].

3.4.3 Autoencoders in different application domains

Autoencoders have demonstrated efficacy across a diverse array of domains. In healthcare settings, specialists are engaged in uncovering anomalies in medical imaging, which encompasses the detection of tumors in Magnetic Resonance Imaging (MRI) scans or irregular features in X-ray images. The intrinsic unsupervised characteristic of autoencoders facilitates their capacity to assimilate knowledge from extensive datasets of normative images and recognize deviations that may indicate the presence of disease [51]. In industrial contexts, autoencoders are deployed to identify equipment malfunctions or declines in performance through the analysis of sensor data. By conducting real-time surveillance of machinery, autoencoders are capable of recognizing initial indicators of failure, thereby enabling predictive maintenance and

minimizing operational downtime. This utility is particularly advantageous in sectors such as manufacturing, where equipment breakdowns can result in substantial financial repercussions [52]. Within the realm of UAV technology, autoencoders serve a significant purpose in guaranteeing safe and reliable flight activities. The ALFA dataset, which encompasses high-frequency sensor data from UAVs, empowers autoencoders to detect anomalies in critical parameters such as altitude, velocity, and orientation. This capability can mitigate the risk of catastrophic failures by identifying potential issues prior to their escalation into serious malfunctions [36], [47].

3.4.4 Enhancing autoencoder performance in anomaly detection

To address some of the limitations of autoencoders, several enhancements can be made. Regularization techniques such as dropout and weight decay are commonly employed to prevent overfitting. These techniques force the model to generalize better by randomly deactivating certain neurons during training or penalizing overly large weights in the model, leading to improved performance on unseen data [18].

Variational Autoencoders (VAE), which encode input data as probabilistic distributions rather than fixed values, offer additional flexibility and are particularly effective for handling complex and noisy data. VAEs have been shown to perform well in real-world applications, such as anomaly detection in high-dimensional time-series data [49]. To further improve their performance on sequential datasets, VAEs and other autoencoders can be integrated with models like Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks. This combined approach allows the model to capture temporal dependencies more effectively, providing robust results in time-series anomaly detection [53].

3.5 Software Frameworks and Techniques for TinyML Applications

The successful development and deployment of TinyML applications demand a comprehensive understanding of specialized software frameworks and implementation techniques. This section delves into the critical steps and considerations for the software development process in TinyML projects. Essential topics include frameworks like TensorFlow Lite for Microcontrollers, the process of project compilation and configuration through build systems, as well as the use of project generation tools. Furthermore, this section will examine the optimization strategies

and deployment challenges for running TinyML models on embedded systems, addressing the specific hardware and software requirements necessary to ensure seamless execution.

The fundamental aim of TinyML is to proficiently implement machine learning algorithms on embedded systems that are constrained in resources, which demands advanced software and optimization methodologies. The interpretation of models, in particular, is crucial for the effective allocation of memory, computational power, and other system resources. Consequently, this segment will expound upon the software methodologies that are imperative for the successful development, management, and deployment of TinyML models within embedded environments.

The structural framework of TinyML-oriented applications, especially for functions such as real-time anomaly detection within UAV systems, is centered on a highly efficient workflow that synergistically incorporates model training, optimization, conversion, and deployment onto devices with limited resources, as depicted in Figure 3.6. To enable the flawless implementation of machine learning algorithms within embedded systems, including ARM Cortex-M microcontrollers, a number of key procedures must be diligently observed.

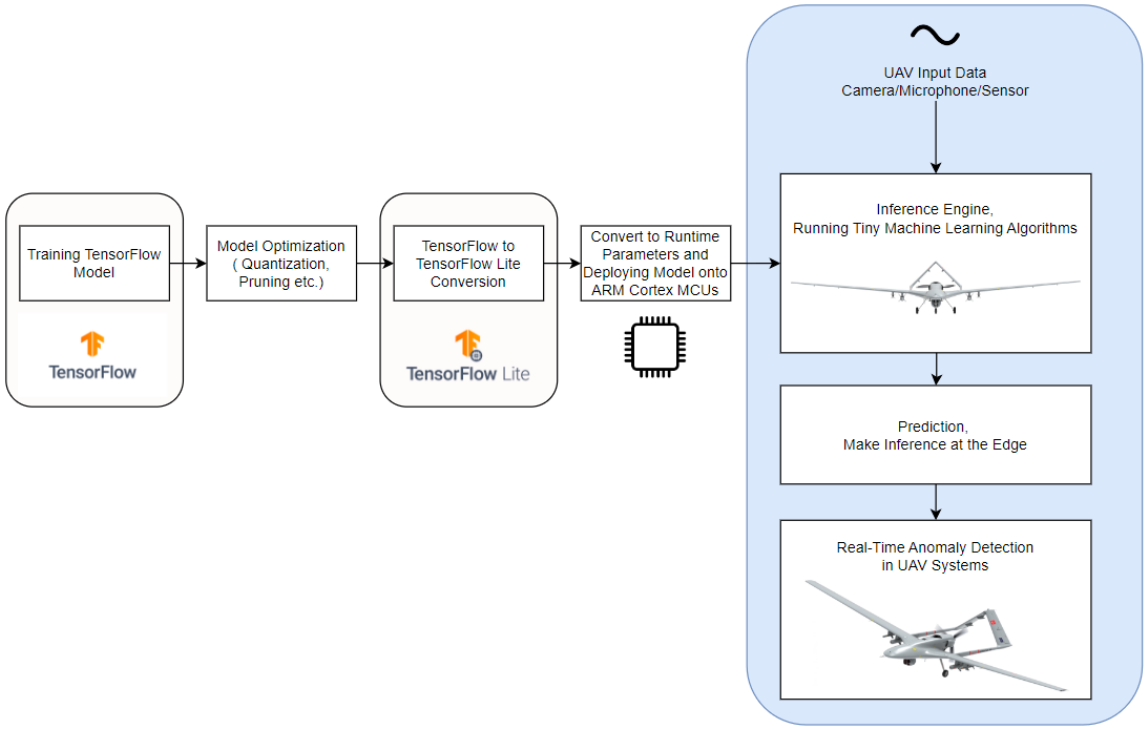

Figure 3.6 : General architecture of TinyML anomaly detection.

The procedural framework commences with the training of the machine learning model utilizing high-performance computing environments, wherein UAV sensor data (e.g., visual and auditory inputs) is analyzed, and the model assimilates normative data patterns and behaviors. The model undergoes training via established TensorFlow architectures on either a desktop or cloud-based platform, thereby guaranteeing its capability to identify intricate patterns within the dataset.

Once the model has been trained, it is subjected to various optimization techniques including quantization and pruning. These optimizations are imperative for diminishing the model's dimensionality and computational demands, which holds particular significance when deploying models on embedded systems characterized by limited memory and processing capabilities. Quantization entails the reduction of the numerical precision of the model's weights, whereas pruning eliminates less critical connections within the neural network, thereby aiding in the reduction of complexity without considerable degradation of accuracy. Subsequently, the optimized model is transformed into TensorFlow Lite format, rendering it appropriate for deployment on microcontrollers. TensorFlow Lite is exclusively designed for inference, discarding the training phase to enhance the model's efficacy on low-power devices. This conversion procedure guarantees that the model remains lightweight and is capable of executing efficiently in real-time applications. The deployment phase encompasses the transfer of the converted model to an embedded system, such as an ARM Cortex-M microcontroller. This procedure guarantees that the model can function with the requisite runtime parameters and generate predictions directly on the device, obviating the need for communication with cloud servers. This aspect is especially crucial in UAV systems, where low-latency, real-time decision-making is vital for operations such as anomaly detection.

The inference engine runs the machine learning model on the microcontroller, enabling the system to detect anomalies in real-time. By continuously monitoring the UAV's sensor data, the system can detect any irregularities or abnormal behaviors, triggering alerts or corrective actions when necessary. This real-time processing capability is vital for ensuring the safety and effectiveness of UAV operations, especially in mission-critical applications. As depicted in Figure 3.6, this overarching framework establishes the groundwork for comprehending the distinct elements that constitute the TinyML workflow, which will be elaborated upon in the subsequent sections: TensorFlow Lite

for Microcontrollers, Prerequisites for TinyML Model Deployment, and the Function of Model Interpretation within Embedded Systems.

The transition from traditional embedded system architectures to TinyML-based architectures marks a significant evolution in how machine learning models are deployed and utilized on resource-constrained devices. In traditional embedded systems, computation is often rigid, with predefined algorithms hard-coded into the system, leaving little room for flexibility or adaptation. These systems typically perform specific, repetitive tasks without the ability to learn or adapt from new data inputs in real-time.

In contrast, TinyML architectures introduce a more dynamic and adaptive workflow, allowing devices to not only make predictions but also adapt to evolving conditions based on real-time data. By integrating machine learning models directly onto embedded systems, particularly microcontrollers like ARM Cortex-M, TinyML enables these devices to process data locally, reducing latency and the need for constant communication with cloud servers. This capability significantly enhances the autonomy and efficiency of embedded systems, especially in real-time applications such as anomaly detection in UAV systems.

TinyML's adaptability comes from its ability to handle varying data patterns and learn from new inputs on the edge. The use of optimization techniques such as quantization and model pruning further enhances this adaptability by ensuring that models can run efficiently even on devices with limited memory and computational power. Unlike traditional embedded systems, which are often constrained by static rules and limited computational capabilities, TinyML systems are capable of dynamically adjusting to new data inputs, improving performance over time as the model learns and evolves. As a result, TinyML-based systems provide a more flexible and intelligent architecture for modern applications. This transition to a dynamic, machine learning-driven approach enables a range of innovative applications in fields such as real-time anomaly detection, predictive maintenance, and autonomous decision-making in UAV systems, ultimately transforming the landscape of embedded systems.

3.5.1 TensorFlow Lite for microcontrollers

TensorFlow is established as one of the leading and versatile machine learning frameworks. Initially released by Google in 2015, it rapidly became the foundation for numerous applications ranging from natural language processing to computer vision, benefiting from its expansive contributor base and open-source nature. The primary goal of TensorFlow was to offer a comprehensive platform for training and deploying machine learning models on high-performance platforms such as desktops and cloud servers. Nonetheless, its initial design encountered limitations when applied to resource-constrained platforms like mobile devices and embedded systems due to the framework's large size and high resource demands [54], [55].

To overcome these challenges, Google introduced TensorFlow Lite in 2017, a streamlined version of TensorFlow optimized for mobile and embedded devices. TensorFlow Lite focuses exclusively on inference, omitting the training phase to reduce size and complexity. It incorporates optimizations such as 8-bit quantization, which enhances model efficiency without significantly compromising accuracy. This design makes TensorFlow Lite particularly suitable for mobile environments, where memory and power constraints are critical [53]. It has gained significant popularity among mobile developers due to its flexibility and efficiency in running pre-trained models on Android and iOS devices [56].

Despite these technological advancements, the binary size of TensorFlow Lite remains a considerable obstacle for microcontrollers and other environments defined by extreme limitations. In 2018, Google introduced TensorFlow Lite for Microcontrollers as a strategic response to this issue. This iteration is meticulously engineered to satisfy the rigorous demands of embedded systems, where both memory and processing capabilities are severely restricted. With a binary footprint of under 20 KB, TensorFlow Lite for Microcontrollers facilitates the execution of real-time machine learning on ARM Cortex-M microcontrollers and analogous devices. This edition incorporates fundamental optimizations such as model quantization, which empowers developers to implement machine learning models effectively in practical applications, including speech recognition and anomaly detection on edge devices [57].

3.5.2 Requirements for TinyML model deployment

Deploying TinyML models on embedded systems presents a set of unique challenges, driven by the resource constraints typical of such environments. Unlike traditional platforms, embedded systems often operate without a full-fledged operating system, making it critical to minimize dependencies and optimize performance. Several key requirements must be met to ensure that machine learning models can run efficiently in these constrained environments.

One of the most important considerations is the absence of operating system dependencies. Since many embedded platforms do not run on conventional OSes, TinyML libraries must be designed to avoid system-level calls. This is particularly important as the machine learning model, at its core, functions as a mathematical "black box" that processes input data and returns predictions without needing to interact with the underlying hardware or operating system [54], [55], [58].

Another pivotal criterion involves the reduction of dependencies on conventional C or C++ libraries. Embedded systems frequently operate within stringent memory constraints, occasionally possessing only a few kilobytes, rendering the incorporation of extensive libraries impractical. For instance, functions such as sprintf() from the standard C library can augment a program's binary size by tens of kilobytes. TinyML frameworks circumvent such functions to maintain a compact overall binary size, opting instead for header-only dependencies or compile-time constants [53]. An exception to this stipulation is the standard C math library, which is imperative for executing fundamental mathematical operations such as trigonometry.

Additionally, most embedded platforms are deficient in hardware support for floating-point arithmetic, thereby exacerbating deployment challenges. To mitigate this issue, TinyML models predominantly utilize integer-based computations, particularly employing 8-bit integers for model parameters and arithmetic operations. This approach not only diminishes memory consumption but also enhances computational efficiency, as integer operations are markedly faster than their floating-point counterparts on embedded systems [56], [58].

Another significant challenge pertains to the management of memory allocation. Embedded systems are typically required to operate continuously for prolonged durations—frequently spanning months or even years—which renders dynamic

memory allocation a precarious endeavor. The utilization of malloc() or free() functions can result in memory fragmentation over time, which may ultimately precipitate system failures. To mitigate this risk, TinyML frameworks such as TensorFlow Lite for Microcontrollers implement static memory allocation, wherein the memory requirements are predetermined at the initialization stage and no subsequent dynamic allocation occurs during the inference process. This approach guarantees a stable and reliable performance over extended timeframes. Despite these constraints, some flexibility is maintained to ensure compatibility with more powerful platforms like mobile devices. For example, while C++11 is required for writing the TinyML framework, this choice was made to support modern toolchains without compromising compatibility across different versions of TensorFlow Lite [57]. Additionally, although the framework is optimized for 32-bit processors, there have been successful implementations on 16-bit platforms using advanced toolchains, though these are not the primary focus.

Meeting these requirements ensures that TinyML models can be deployed effectively on a wide range of embedded systems, from low-power microcontrollers to more advanced processors, while maintaining efficient performance and minimizing resource consumption.

3.5.3 The role of model interpretation in embedded systems

In embedded machine learning applications, particularly in TinyML, one key design choice is whether to interpret the model at runtime or generate code ahead of time. Both approaches have their advantages, but interpreting the model offers several critical benefits that align with the constraints of embedded systems. Code generation involves converting a machine learning model into C or C++ code, embedding the model's parameters directly into data arrays and representing the architecture through function calls. While this method can result in simplified build systems and modifiability, it also presents challenges when working with multiple models or when upgrading the framework. Code generation is less flexible, as changing or updating models requires recompiling the entire program, which can be time-consuming and impractical in dynamic environments [54].

On the other hand, interpreting a model at runtime offers flexibility, upgradability, and efficiency. In this approach, the model is treated as a data structure that defines the

operations to be executed, while the underlying code remains static. This means that new models can be easily swapped in without recompiling the program, which is especially valuable in embedded systems where storage and computational resources are limited [53], [55]. Moreover, interpreted models enable developers to run multiple models without duplicating source code, making it easier to manage large-scale deployments in resource-constrained environments [58].

Interpreting models also simplifies the process of upgrading the machine learning framework. Since the framework remains unchanged and only the model data is modified, new optimizations or features can be integrated without requiring significant changes to the existing system. This makes model interpretation particularly attractive for long-term embedded applications, where maintaining performance while reducing manual intervention is crucial [56], [58].

By using project generation techniques, TinyML frameworks can leverage the advantages of model interpretation, such as ease of deployment and flexibility, without incurring the downsides of code generation. This approach secures that TinyML applications are adaptive and can be seamlessly upgraded, meanwhile ensuring they perform well and efficiently in environments with constrained resources.

4. IMPLEMENTATION AND RESULTS

This chapter presents a thorough examination of the comprehensive implementation process from beginning to end and the findings derived from the research. This part is arranged into numerous individual sections, each emphasizing a specific phase of the examination, incorporating the setup and assessment of the ALFA data, the creation and training of machine learning systems, and the integration of the developed systems into a software architecture tailored for real-time anomaly identification. Initially, the chapter focuses on the detailed preprocessing techniques employed on the ALFA dataset to ensure data integrity and relevance. This includes handling missing values, normalization, and transformation of the raw flight data. A custom-built Graphical User Interface (GUI) was developed to visualize and monitor flight data, providing a critical understanding of UAV behaviors under various operating conditions. Furthermore, in-depth analysis is provided for multiple flight parameters, such as global position, roll-pitch-yaw dynamics, and sensor diagnostics, which serve as crucial indicators for identifying anomalous behaviors in UAV systems.

The ensuing sections investigate the evolution and conditioning of the machine learning frameworks. A range of architectures was rigorously examined and enhanced to maximize performance indicators such as precision, recall, and F1-score. Hyperparameter optimization, encompassing modifications to the learning rate, number of epochs, and batch size, was executed to augment the model's proficiency in effectively identifying anomalies. The essence of the implementation centers around the integration of the trained models onto a resource-constrained ARM Cortex-M microcontroller. This segment clarifies the conversion process of the TensorFlow model into a finely tuned TensorFlow Lite (TFLite) format, with particular emphasis on methodologies including quantization and pruning aimed at diminishing memory usage and computational demands. The conversion procedure is articulated, underscoring how each optimization phase influences the model’s appropriateness for embedded systems. Additionally, a thorough assessment of the efficacy of the implemented model is presented. A variety of metrics, encompassing inference

latency, CPU utilization, and memory consumption, are examined to assess the model's feasibility for real-time applications on the ARM Cortex-M7 processor A comparative assessment is performed to demonstrate the efficacy of the optimized TFLite models relative to their original TensorFlow versions. The challenges faced during the implementation process, such as failures in memory allocation and scheduling conflicts inherent in the dual-core architecture, are extensively examined alongside the strategies employed for their resolution. Ultimately, this chapter presents a comprehensive evaluation of the findings, illustrating the practicality and efficacy of the suggested TinyML-oriented anomaly detection framework tailored for UAV systems. Additionally, it delineates recommendations for prospective research avenues, including the investigation of more sophisticated pruning methodologies and integrated scheduling frameworks, to augment the implementation of machine learning algorithms on ultra-low-power embedded systems.

4.1 ALFA Dataset Processing and Analysis

4.1.1 Data refinement and preprocessing of UAV sensor data

A detailed plan for data refinement is important for augmenting the capability of models focused on anomaly detection, particularly in complex frameworks like Unmanned Aerial Vehicles. The ALFA dataset, which incorporates an extensive array of sensor data derived from various UAV operational scenarios, was employed for this investigation. This dataset was chosen owing to its thorough depiction of UAV operational states and the integration of both normative and anomalous conditions, thereby establishing it as an exemplary foundation for the training and assessment of anomaly detection models [27], [36], [47]. The ALFA dataset comprises a broad selection of aeronautical data obtained from high-frequency sensors, inclusive of accelerometers, gyroscopes, magnetometers, GPS apparatus, barometers, and temperature measuring devices. These data streams furnish a complex understanding of the unmanned aerial vehicle's performance across varying conditions, thereby facilitating the detection of irregularities that may indicate potential malfunctions.

A MATLAB-based analytical framework has been established to perform an in-depth investigation of both the "Raw" and "Processed" data within the ALFA dataset. In the established structure, the information goes through various initial procedures to confirm its preparedness for later examination.

- Standardization: To ensure consistency and enhance the model's efficacy, all sensor data underwent standardization. Standardization adjusts the data to a common range, typically [0, 1] or [-1, 1], thus averting any single feature from unduly influencing the model. This stage is pivotal for advancing the convergence rate during training and guaranteeing effective learning from the data.
- Dealing with Missing Values: Missing data in the dataset, stemming from sensor malfunctions or data transfer issues, were tackled using interpolation methods. Interpolation involves approximating missing values based on the existing data, guaranteeing the dataset's completeness and impartiality. This procedure is essential for upholding data integrity and ensuring the training process's reliability.
- Data Segmentation: The continuous sensor data were segmented into fixed-size sequences to ease the autoencoder model's training. Each segment denotes a specific time window, enabling the model to capture temporal relationships and patterns within the data. This segmentation is crucial for models crafted to analyze time-series data, as it empowers them to learn from the temporal organization of the inputs.
- Categorization: Solely normal operational data were utilized for training the autoencoder in an unsupervised manner. Abnormal data points were excluded from the training group and set aside for validation and testing. This segregation guarantees that the model can accurately reconstruct normal data, rendering it more attuned to anomalies during assessment.
- Feature Selection: Vital features pertinent to UAV operation, like altitude, speed, orientation, and battery status, were cherry-picked for incorporation in the training data. These features are pivotal for detecting anomalies that could influence the UAV's performance and safety. By concentrating on these key parameters, the model is better equipped to pinpoint significant deviations from regular behavior.
- Data Partitioning: The dataset was systematically divided into three distinct subsets: training, validation, and testing sets. The training portion was involved in the development of the autoencoder model with normative insights, while

the validation portion was employed to modify the model's settings and minimize overfitting risks, and the testing portion was engaged to assess the model's success on new information, which comprises both normative and unusual examples. This stratified division facilitates a comprehensive evaluation of the model's capacity for generalization and its proficiency in detecting anomalies.

The ALFA dataset is specifically designed for fault and anomaly detection in Unmanned Aerial Vehicles, containing comprehensive log records from multiple test scenarios. Figure 4.1 illustrates the flight trajectory of the UAV during one of these test sessions, showcasing the area where the data was recorded. This trajectory highlights the consistent movements and operational patterns of the UAV under controlled conditions.

Figure 4.1 : UAV flight trajectory in the ALFA dataset [36].

The dataset consists of four main types of data logs, as detailed by Keipour et al. (2021):

1. Autonomous Flight Sequences with Failures: These logs capture autonomous flight data focused solely on engine failure events. Available in CSV, MATLAB .mat, and ROS .bag formats, this subset provides a clear depiction of flight anomalies.

2. Raw Flight Sequences: This category includes unprocessed flight data in various modes, containing multiple failure scenarios. It serves as the basis for deriving specific autonomous flight logs.
3. Telemetry Logs from TX2: Recorded by the onboard TX2 computer, these logs provide a detailed account of the UAV's operational parameters without including fault-specific ground truth data.
4. Dataflash Logs from Pixhawk: These logs, directly extracted from the Pixhawk autopilot system, offer unprocessed details about the UAV's flight conditions, serving as an essential input for anomaly detection algorithms.

Figure 4.2 provides an example visualization of key flight parameters roll, pitch, and yaw processed from the ALFA dataset. These parameters illustrate the UAV's orientation changes over time and highlight specific intervals corresponding to engine failures. While Figures 4.2 focuses on roll, pitch, and yaw as representative metrics, the ALFA dataset also includes a wide range of other critical flight parameters such as altitude, GPS position, acceleration, battery health, and engine thrust, all of which can be processed and analyzed using the developed MATLAB framework.

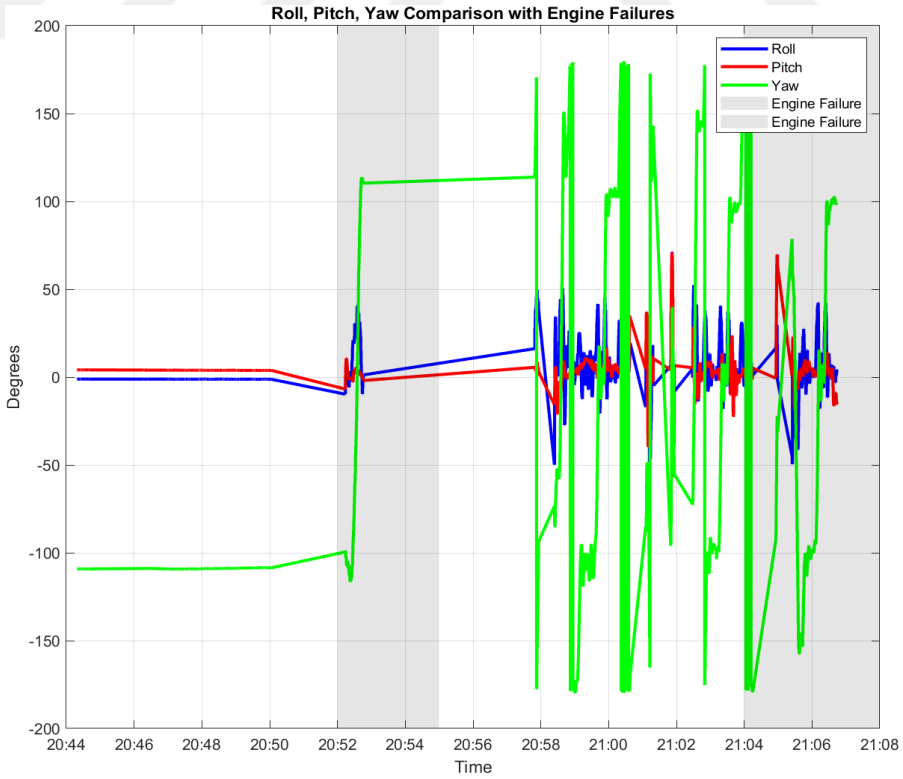

Figure 4.2 : Roll, pitch and yaw comparison with engine failures.

Figure 4.3 depicts the roll, pitch, and yaw axes on a Turkish-made tactical unmanned aerial vehicle, Bayraktar TB2. This figure helps to visually contextualize the orientation metrics discussed, providing a clearer understanding of how these axes relate to the UAV's movements during flight.

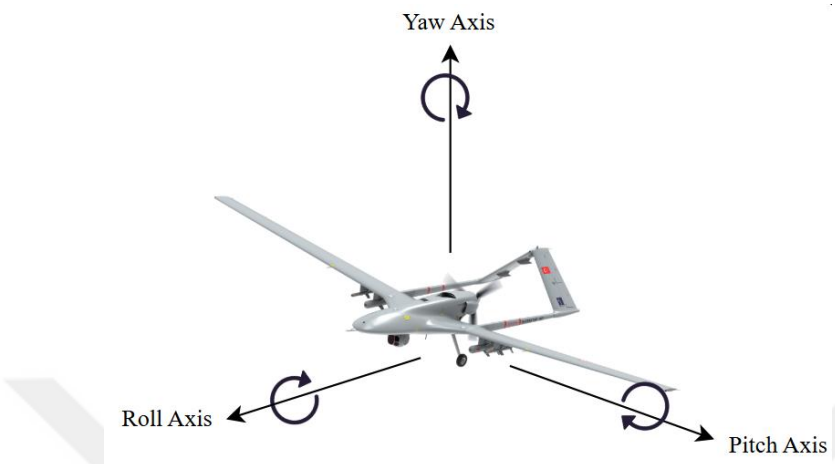

Figure 4.3 : Roll, pitch and yaw axes representation on Bayraktar TB2 UAV.

In the MATLAB-based analytical framework, the first two data types from the ALFA dataset, autonomous flight sequences with failures and raw flight Sequences were extensively utilized to create a dedicated flight analysis script. This script was specifically designed to process and analyze flight dynamics by transforming raw data into structured information, enabling the identification of irregularities during engine failure events. These data types provide the foundational elements for observing both isolated failure events and broader operational trends, which are crucial for effective anomaly detection. The primary objective of this preprocessing and analysis stage is to transform raw data from the ALFA dataset into a structured and analyzable form. By incorporating these two specific data types, the framework enables detailed visualization and correlation of UAV parameters, providing valuable insights into the system's operational and failure states. For example, correlations between parameters such as engine thrust and acceleration, or battery health and flight time, can be observed, highlighting potential dependencies and patterns. This step is critical before progressing to anomaly detection model development and training. Without accurate preprocessing and analysis of these data types, the raw data would remain unstructured and unsuitable for machine learning tasks. By performing this analysis, it becomes possible to verify the quality and integrity of the dataset, ensuring that the subsequent anomaly detection model is built on reliable and meaningful data. The ability to

correlate diverse flight parameters enables not only anomaly detection but also the validation of the UAV's performance in various scenarios. This foundational analysis ensures that the dataset supports the development of robust and accurate models for detecting anomalies, enhancing the safety and reliability of UAV operations.

4.1.2 Graphical User Interface (GUI) for data analysis

A simplified Graphical User Interface (GUI) has been designed to facilitate the analysis and visualization of UAV flight data for real-time anomaly detection. This interface is part of the broader analyzer project, which was developed after a detailed investigation of the research topic to observe flight parameters efficiently. The GUI allows users to load, analyze, and visualize data from .bag files, a format widely used in UAV systems based on the Robot Operating System (ROS).

The primary function of the GUI is to provide a user-friendly platform where users can interact with flight data in a structured manner. Upon launching the GUI, users are prompted to select a .bag file, which contains the raw flight data. Once the file is loaded, the interface presents various options for visualizing key flight parameters. These include Roll, Pitch, Yaw, Flight Acceleration, Magnetic Field Data, Battery Health, Global Position, and Flight Diagnostics. The GUI allows users to either analyze specific parameters or conduct a general analysis, which automatically generates plots for all key metrics. Each button on the interface corresponds to a distinct flight parameter, allowing the user to selectively visualize data based on their needs. For instance, selecting "Roll, Pitch, Yaw" generates individual graphs for these orientation parameters, providing detailed insights into the UAV's flight dynamics. Other parameters, such as Flight Acceleration and Battery Health, offer critical information about the performance and condition of the UAV during its mission. The general analysis function compiles all these plots, giving a holistic view of the UAV's performance throughout the flight.

Included in the GUI is an important option for transferring the evaluated information to a CSV (Comma-Separated Values) document. This export process is crucial for transitioning the data from analysis to machine learning tasks. The exported CSV file contains clean and processed data, ready to be used in machine learning models for anomaly detection. During the data processing phase, various methodologies are employed to address the issue of missing values, including mean imputation, moving

averages, or the assignment of NaN (Not a Number) or zero values contingent upon the specific context. This ensures that the exported dataset is robust and suitable for training models in Python. This GUI, with its simplified design, not only aids in visualizing and analyzing flight data but also serves as a bridge between data exploration and machine learning. The integration of CSV export capabilities streamlines the process of anomaly detection by preparing the data for model training, contributing to the overall goal of real-time anomaly detection in UAV systems.

4.1.3 Flight data visualization and analysis

In this research, flight data from various UAV missions included in the ALFA dataset were analyzed to evaluate the UAV's behavior under both normal and abnormal conditions. Among these flights, the data recorded on July 18, 2018, was selected as a representative example for detailed graphical evaluations. This particular flight experienced multiple engine failures, making it an essential case for observing how various parameters respond under critical conditions. The analysis focused on several key parameters, including roll, pitch, yaw, altitude, global position, magnetic field, battery health, flight acceleration, and system diagnostics. The primary purpose of this analysis was to visualize parameter variations and identify anomalous behavior, particularly in correlation with engine failure events. While all flights in the ALFA dataset were processed and reviewed, the graphical evaluations in this section are based on the flight data from July 18, 2018, due to its significant engine failure scenarios. The first parameter analyzed was global position and altitude, which provided insights into the UAV's latitude, longitude, and altitude variations throughout the flight. Engine failures occurred at various time intervals, and their impact on the UAV’s positioning was clearly visible. Significant deviations in altitude during engine failure periods suggest potential issues in the control mechanisms or power distribution system, as the UAV struggled to maintain stable flight. In the subsequent sections, other critical parameters such as roll, pitch, yaw, magnetic field, battery health, and flight acceleration will be analyzed in detail. These evaluations aim to understand how the UAV's internal systems responded to failure conditions, providing valuable insights for anomaly detection model development. Each graph will highlight the correlation between engine failures and parameter fluctuations, showcasing the UAV's overall behavior during critical events.

4.1.3.1 Analysis of global position and altitude during engine failure

The analysis of global position and altitude data is critical in understanding the UAV's overall flight behavior, particularly when engine failures occur. Monitoring these parameters helps in identifying deviations from normal flight patterns, which is essential for anomaly detection. UAV systems are designed to maintain a stable flight even under adverse conditions, such as engine failure. However, deviations in the latitude, longitude, and altitude could signal that the system is compensating for failures, or in worse cases, losing control. Therefore, any substantial fluctuation in these metrics can be considered an anomaly and requires careful examination. As depicted in Figure 4.4, both the latitude and longitude values fluctuate during the red-shaded intervals, which correspond to the times of engine failure. These fluctuations suggest that the UAV's stability was affected during these periods. Such deviations in global position may indicate that the UAV was attempting to counteract the effects of the engine failure, such as maintaining its course or adjusting to external forces like wind. However, without a reference for expected flight behavior under normal conditions, it becomes challenging to determine whether these changes are critical anomalies or the UAV's normal compensatory mechanisms.

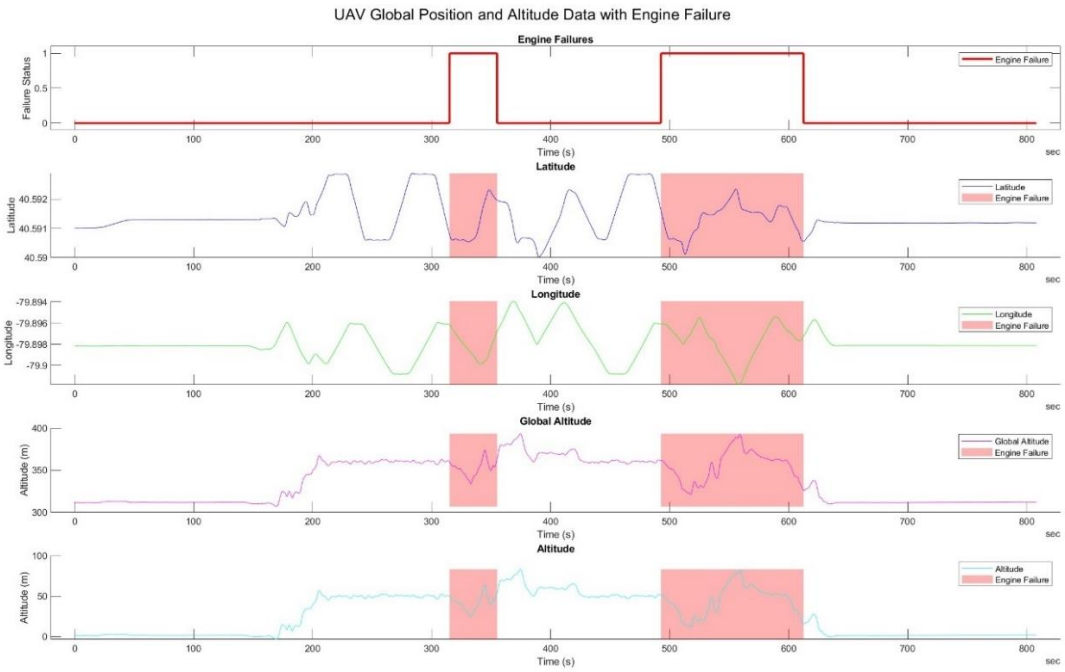

Figure 4.4 : UAV global position and altitude data with engine failure.

Altitude is another critical factor in anomaly detection during UAV flights. Significant changes in altitude, especially during engine failure periods, can indicate a loss of

control or inability to maintain the desired flight path. The altitude graph in Figure 4.4 provides a detailed representation of altitude variations. During the engine failure periods, sharp altitude drops are observed, further confirming the impact of engine failure on the UAV's ability to maintain a stable height. These drops highlight potential critical points in the flight that require further analysis to determine the extent of the failure's impact.

The altitude graph provides a more detailed representation of altitude variations. The sharp drops in altitude during the failure periods (highlighted in red) indicate a significant loss of height, further confirming the adverse effect of engine failure on the UAV's flight stability. This loss of altitude during engine failure could be indicative of the UAV's inability to generate enough thrust to maintain its altitude or could signal a failure in the flight control system.

In conclusion, global position and altitude data are crucial for detecting anomalies during UAV flights. The fluctuations in these parameters, especially during failure periods, offer insights into the UAV’s performance under stress. A detailed comparison with flights under normal conditions is required to fully assess whether these deviations are anomalous behaviors or expected system responses.

4.1.3.2 Analysis of roll, pitch and yaw during engine failure

An important aspect of flight stability and control in UAV systems revolves around the roll, pitch, and yaw angles, collectively referred to as the "attitude" of the UAV. These parameters are critical in maintaining proper orientation, particularly during dynamic conditions such as engine failures. In this research, the roll, pitch, and yaw angles are continuously monitored throughout the flight to assess how the UAV responds to engine anomalies.

As depicted in Figure 4.5, the red-shaded intervals correspond to periods of engine failure, during which notable changes in these angular parameters can be observed. The roll angle exhibits significant fluctuations, particularly during the failure intervals between 300 and 600 seconds. These fluctuations are indicative of the UAV's attempts to stabilize itself in response to the engine failures, though the extent of the variations suggests moments of instability. Large changes in the roll angle can lead to compromised control, especially when they occur rapidly or unexpectedly, which can pose a risk to flight safety.

The pitch angle, on the other hand, demonstrates fewer significant changes throughout the flight, with only occasional spikes occurring during the engine failure events. This might suggest that the UAV's control system was more successful in managing pitch during failures, or that the loss of an engine had less of an impact on the pitch due to its particular flight dynamics at the time.

The yaw angle exhibits notable fluctuations, particularly during intervals characterized by engine failure. Yaw control is inherently sensitive to power asymmetry induced by engine malfunctions, and the detected variations in yaw may be attributed to the unmanned aerial vehicle striving to uphold its directional stability. Considerable yaw deviations can result in diminished navigational precision, which is especially crucial during extended or mission-critical operations.

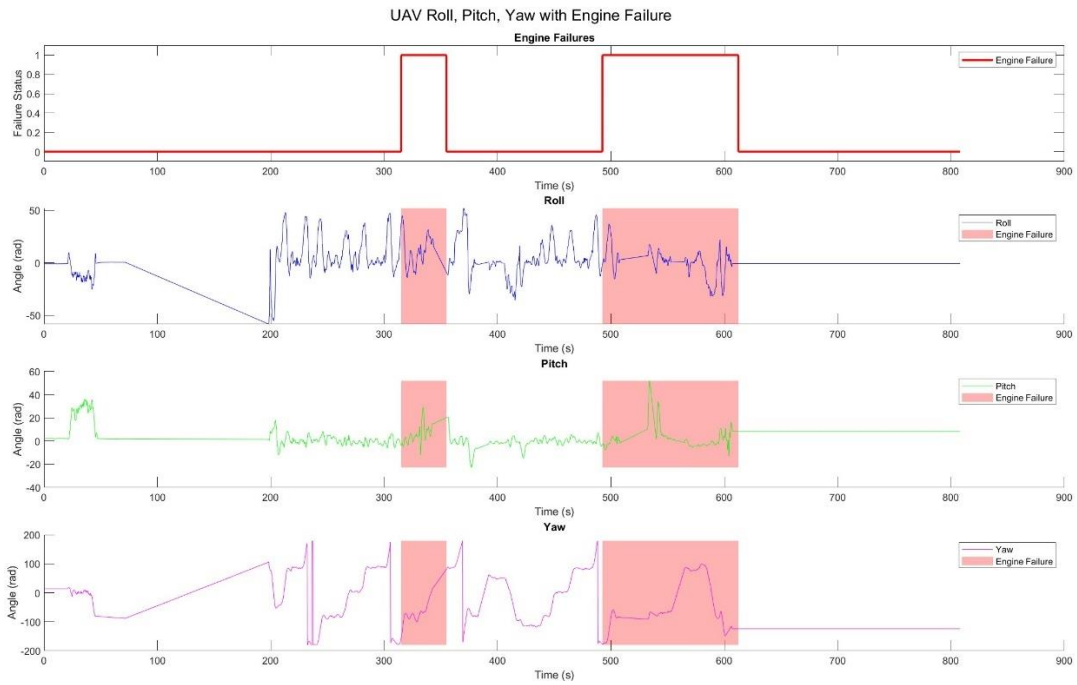

Figure 4.5 : UAV roll, pitch and yaw data during engine failure intervals.

Notwithstanding the discernible alterations in roll, pitch, and yaw amidst engine failure occurrences, the identification of these irregularities solely through visual examination of the graphical data poses significant challenges. The variations in these angular measurements could be interpreted as an integral aspect of the UAV's standard response to engine failures, complicating the differentiation between normal compensatory behaviors and genuine anomalies in the absence of a comprehensive baseline for anticipated performance. Additionally, human observation may fail to detect nuanced yet consequential changes in the data, particularly during episodes of

high-frequency oscillation. Consequently, depending on human intervention for the real-time identification of anomalies would be inefficient and susceptible to inaccuracies. Thus, automated anomaly detection frameworks—specifically, the machine learning algorithms utilized in this study—are indispensable for an exhaustive and precise evaluation of these angular parameters. Through the examination of variations in roll, pitch, and yaw in conjunction with other flight metrics, the model is capable of recognizing nuanced patterns that signify potential system malfunctions or operational challenges. This holistic methodology facilitates the detection of not only engine-related irregularities but also other significant system failures that may jeopardize the performance of the UAV.

In conclusion, roll, pitch, and yaw represent critical parameters for the maintenance of flight stability, and their examination yields vital insights into the UAV's reactions to dynamic conditions such as engine failures. Nevertheless, in the absence of a systematic framework for anomaly detection, accurately pinpointing the precise moments of failure based solely on these graphical representations proves to be challenging. The incorporation of these parameters into a comprehensive anomaly detection model is crucial for guaranteeing real-time recognition of issues and for upholding the safe operation of UAVs.

4.1.3.3 Flight diagnostics and system monitoring during UAV flight

Conducting flight assessments is fundamental for securing the dependability and safety of UAV operations. These diagnostic systems perpetually assess an array of system health indicators to identify anomalies or failures in real-time. Figure 4.6 illustrates the diagnostic tiers for several integral components of the UAV, encompassing Flight Control Unit (FCU) connections, GPS diagnostics, system heartbeat, and comprehensive system diagnostics.

- FCU Connections: This parameter ensures stable communication between the flight control unit and the UAV’s subsystems. As shown in the first plot, the connection remains stable throughout the flight with no warnings or errors. A consistent connection is essential for maintaining control over the UAV.
- GPS Diagnostics: The second plot shows the health of the GPS system, which remains stable and operational without any disruptions. GPS is crucial for

navigational accuracy, especially in autonomous flight modes. Any deviation in this parameter could lead to loss of positional awareness and mission failure.

- Heartbeat Diagnostics: The third plot tracks the system heartbeat, which indicates whether the UAV's core systems are operational. Any irregularities in this parameter could indicate issues such as power loss or component failure, though no such issues were observed in this flight.
- System Diagnostics: The fourth plot indicates an error in the system diagnostic level, which corresponds to a period of engine failure. This diagnostic alert reflects a critical issue that would require immediate attention. In the context of anomaly detection, this is a key metric that signals significant issues in flight stability.
- Battery Diagnostics: Finally, the battery diagnostic level remains stable during the flight, as shown in the last plot. Stable battery performance is crucial, as power fluctuations could disrupt the UAV’s ability to maintain flight or perform critical tasks.

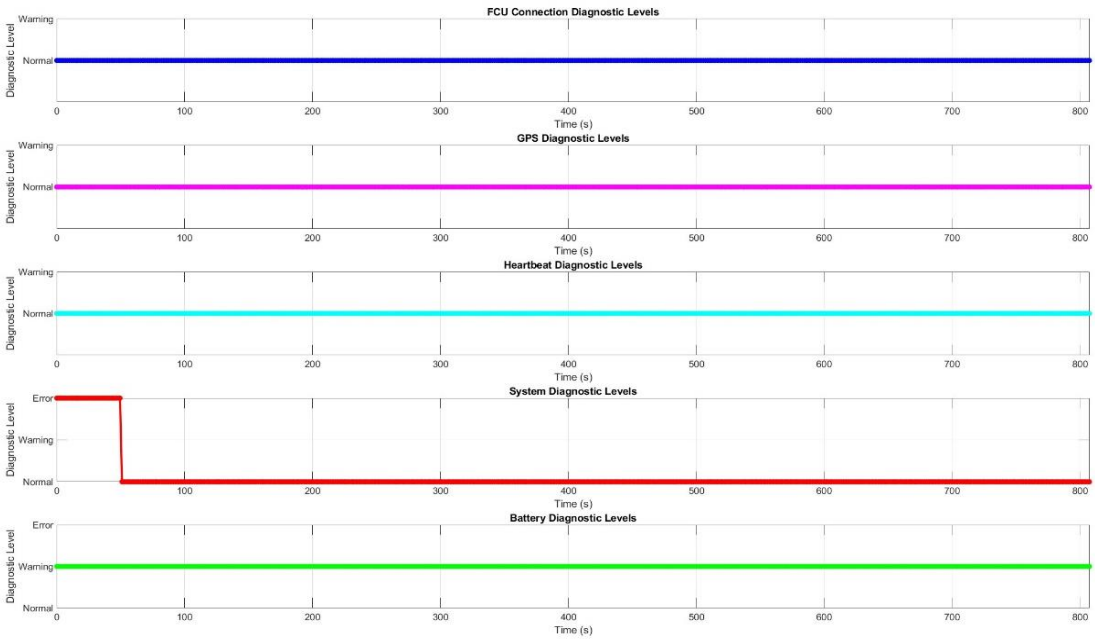

Figure 4.6 : UAV system and diagnostic health monitoring during flight.

The consistent levels in the first three diagnostics - FCU, GPS, and heartbeat - indicate that the UAV’s core systems were operating normally. However, the sharp change in system diagnostics around the engine failure intervals is a critical indicator of the UAV's vulnerability during these periods. Monitoring these parameters is essential for

maintaining real-time awareness of the UAV's operational state. Without automated diagnostics or real-time anomaly detection systems, such faults could go unnoticed until they escalate into more serious failures.

In conclusion, integrating flight diagnostics with machine learning-based anomaly detection models allows for a more robust assessment of UAV performance. These diagnostics, when analyzed together, provide a comprehensive view of system health, enabling timely interventions and ensuring safe flight operations.

4.1.3.4 Analysis of flight acceleration data during engine failure

Flight acceleration metrics yield essential understanding regarding the motion dynamics of Unmanned Aerial Vehicles, particularly during pivotal occurrences such as engine malfunctions. Figure 4.7 illustrates how the data is organized into three unique elements: acceleration across the X, Y, and Z dimensions, together with the UAV's ground speed, which is further displayed to deliver a holistic view of its flight performance.

- Acceleration in X Direction: This component primarily reflects changes in forward and backward motion. In the red-shaded intervals representing engine failure events, we observe significant variations in the X-direction acceleration. These fluctuations suggest that the UAV experienced rapid deceleration or acceleration as it attempted to compensate for the engine failure. The acceleration appears to become erratic during engine failures, indicating instability in maintaining a constant forward motion.
- Acceleration in Y Direction: The Y-axis represents lateral movements, and it is relatively stable throughout the flight. However, during the engine failure periods, minor deviations are observed. Although the changes are less pronounced compared to the X direction, they indicate that the UAV experienced some lateral instability during the failure events, possibly due to asymmetric thrust or control efforts.
- Acceleration in Z Direction: The Z-axis quantifies vertical acceleration, which is imperative for comprehending the altitude control of the UAV. During the intervals characterized by engine failure, we document significant surges in the Z-axis acceleration, particularly within the temporal range of 300 to 500 seconds. These surges indicate that the UAV underwent abrupt fluctuations in

vertical velocity, potentially attributable to a loss of lift or the UAV's endeavors to sustain altitude amidst the impairment of engine power. This phenomenon is of paramount importance, as vertical instability may precipitate altitude discrepancies that jeopardize the mission or, in extreme cases, culminate in catastrophic failures.

- Ground Speed: Ground speed serves as a comprehensive indicator of the velocity at which the UAV traverses the terrestrial surface. The graphical representation illustrates variances in ground speed coinciding with occurrences of engine malfunctions, with discernible deceleration events correlating with the intervals of failure. This further substantiates the hypothesis that engine malfunctions substantially influenced the UAV's capacity to sustain a consistent flight trajectory, culminating in reductions in speed and necessitating compensatory maneuvers.

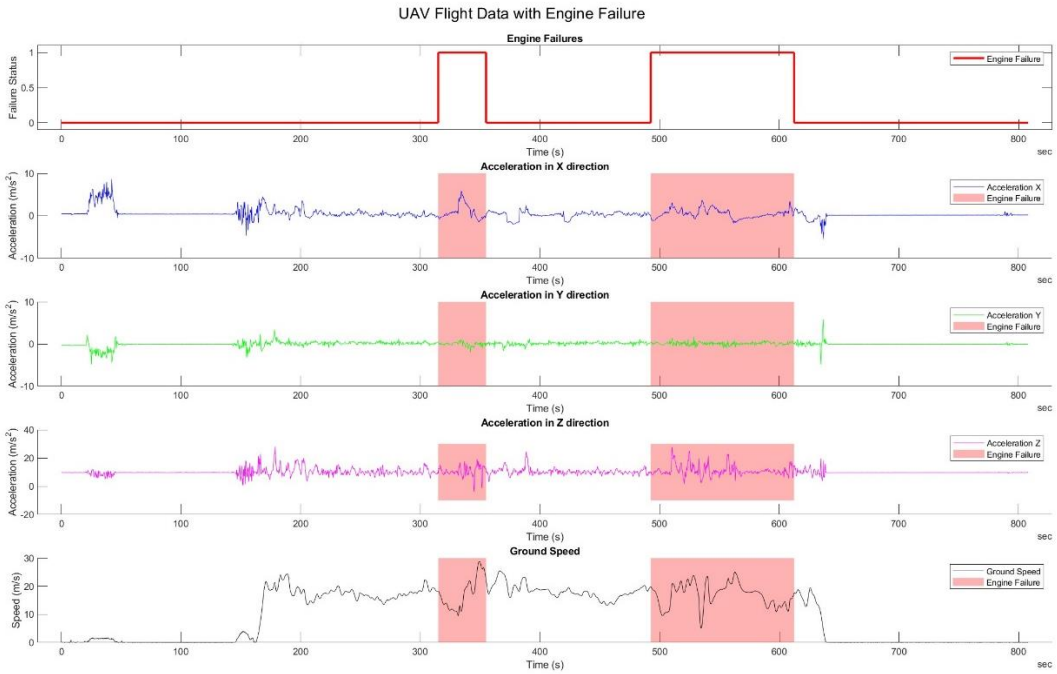

Figure 4.7 : UAV flight acceleration and ground speed during engine failure.

The acceleration data, when analyzed together with ground speed, offers a comprehensive view of the UAV's dynamic response to engine failures. The spikes and dips across all axes suggest that maintaining stability during these failures was challenging. The red-shaded intervals clearly highlight how engine power loss disrupted the UAV's otherwise stable flight. However, distinguishing normal

compensatory responses from anomalies in these graphs is challenging without further analysis, particularly given the complex nature of UAV dynamics.

Thus, automated anomaly detection systems are essential for interpreting these acceleration patterns. By leveraging machine learning models, the subtle changes in acceleration and ground speed can be analyzed to detect deviations that signal not just engine failures but other flight-critical issues. This ensures real-time identification and rectification of potential hazards during flight operations.

In conclusion, flight acceleration data plays a crucial role in assessing UAV performance during engine failures. While human observation can highlight some anomalies, a more thorough analysis using automated systems is necessary for real-time anomaly detection. The comprehensive analysis of acceleration in all directions, along with ground speed, enables more accurate assessments of UAV flight stability and performance.

4.1.3.5 Analysis of magnetic field data during engine failure

Magnetic field data is a critical parameter that influences the stability and orientation of UAV systems. During flight, particularly in the event of engine failures, magnetic field disturbances can have a significant impact on navigation systems and compass-based directional control. The UAV's magnetic field data is analyzed across three axes: X, Y, and Z, as shown in Figure 4.8, which highlights the periods of engine failure in red-shaded intervals. The fluctuations in these magnetic components provide insights into how the UAV responds to engine failures and external disturbances.

- Magnetic Field in X Direction: This component shows considerable fluctuations, especially during the red-shaded intervals representing engine failure events. Between 300 and 600 seconds, significant spikes and dips are observed, indicating heavy disturbances in the magnetic field during these periods. Such fluctuations could potentially lead to inaccuracies in the UAV's compass readings, which in turn may result in navigation errors. These anomalies in the X direction reflect the UAV's attempts to adjust its orientation in response to the asymmetric thrust caused by the engine malfunction.
- Magnetic Field in Y Direction: Similar to the X direction, the Y component also displays irregular behavior during engine failure intervals. Although the magnetic disturbances in this axis are less pronounced, they still indicate

significant variations, suggesting the UAV was affected by internal system irregularities or environmental factors. These changes in the Y-axis magnetic field may impact the UAV's lateral control, contributing to overall flight instability during engine failure events.

- Magnetic Field in Z Direction: The Z component exhibits the most erratic behavior during the engine failure events, particularly between 300 and 600 seconds. These sharp changes in the Z-axis magnetic field could be indicative of significant vertical disruptions, possibly due to the UAV's engine power loss affecting altitude control. The instability in the Z direction highlights the challenges faced by the UAV in maintaining vertical stability, and this data is crucial in understanding how the UAV compensates for such failures.

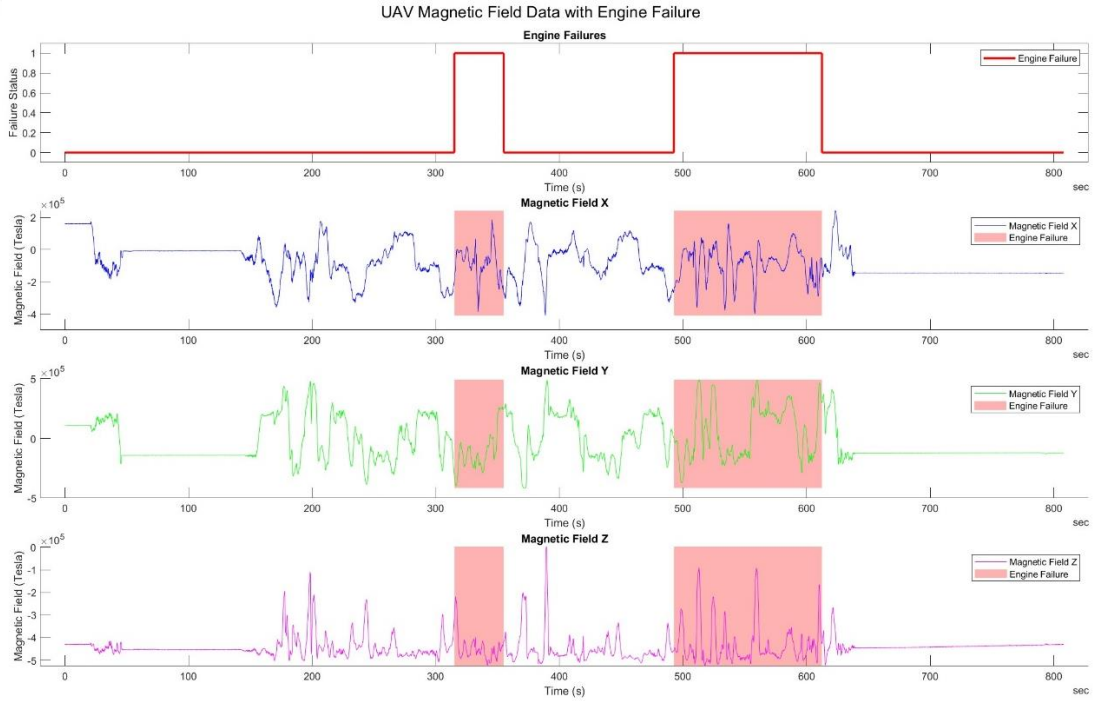

Figure 4.8 : UAV magnetic field intensity during engine failure intervals.

In general, fluctuations in the magnetic field can disrupt the UAV's navigation systems, leading to potential flight path deviations or directional inaccuracies. Since the magnetic field is closely tied to the compass and navigation mechanisms, disturbances in these readings may compromise the UAV's ability to maintain a steady flight trajectory. This can be particularly problematic during mission-critical operations where precise navigation is essential.

Despite the observable disturbances in the magnetic field data, detecting these anomalies visually remains challenging. The fluctuating nature of the magnetic field, especially during failure intervals, could be part of the UAV's normal response to engine malfunctions. Human observation may miss subtle yet significant changes, making real-time anomaly detection difficult without the assistance of automated systems. Incorporating magnetic field data into the anomaly detection model enhances the system's ability to identify engine-related failures and other critical issues that affect UAV performance. By analyzing the deviations in magnetic field readings alongside other flight parameters, the model can more effectively detect anomalies that would otherwise be overlooked. Magnetic field data is essential in assessing the UAV's response to engine failure, particularly in how it maintains stability and navigational accuracy. The disturbances in the X, Y, and Z components of the magnetic field underscore the challenges the UAV faces in such situations. Integrating this data into a broader anomaly detection framework ensures more precise and reliable identification of system failures during real-time flight operations.

4.1.3.6 Analysis of battery health data during engine failure

The analysis of battery health data, as represented in Figure 4.9, shows that the battery current remains relatively constant throughout the flight, including during the engine failure intervals. The graph demonstrates a flat line around 10 mA, which seems inconsistent with the expected behavior of a UAV battery during flight operations.

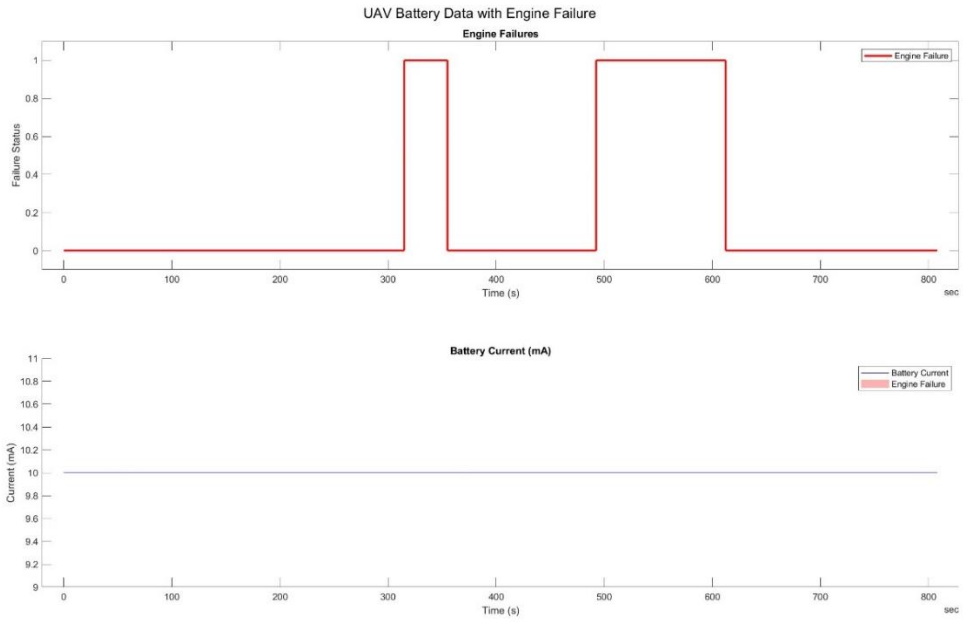

Figure 4.9 : UAV battery current during engine failure intervals.

Given that UAVs typically operate at much higher current levels to power their motors and onboard systems, this suggests that the ALFA dataset does not provide sufficient or accurate data for the battery's power consumption during flight. The lack of significant variation in the battery current during engine failures may indicate that the dataset does not include detailed or high-resolution power data. Monitoring both the current and voltage of the battery during engine failures and other critical events would provide essential insights into the UAV's power management system. A UAV's battery health is crucial for understanding the overall system's performance and stability, especially during anomalous events such as engine failures. Although the existing information derived from the ALFA dataset lacks the granularity necessary for a thorough examination of battery health, it is crucial to acknowledge that, within a conventional UAV anomaly detection framework, the monitoring of battery current and voltage is of paramount importance. Variations in these parameters may indicate potential complications such as energy deficits, excessive loading, or equipment malfunctions, all of which can adversely affect flight safety and operational efficacy. Hence, the integration of precise battery health metrics into the anomaly detection algorithm would significantly augment the UAV's capability to identify and address power-related anomalies in real-time.

In conclusion, although the battery current data from the ALFA dataset during engine malfunctions is inadequate for an in-depth analysis, subsequent research should emphasize the acquisition of more extensive power data, encompassing both current and voltage measurements. These parameters are essential for ensuring the reliability of UAV systems and for the precise identification of potential power system anomalies during critical situations such as engine failures.

4.2 Machine Learning Model Development, Training, and Evaluation

4.2.1 Data preprocessing

The ALFA dataset is a time-series dataset that captures sequential UAV parameters recorded at high frequency during various flight scenarios. Each data point is associated with a timestamp, representing a specific moment in the flight. This temporal structure makes time-series data unique, but it also presents challenges that demand careful preprocessing. In anomaly detection tasks, preserving the temporal continuity and integrity of the dataset is essential, as the identification of irregular

patterns heavily relies on the accurate representation of the time-dependent relationships between data points. For the development of a robust anomaly detection model, preprocessing the ALFA dataset serves as a foundational step in the machine learning pipeline. This phase ensures that the input data is of high quality, consistent, and relevant, directly affecting the model's ability to detect anomalies with high accuracy.

In the context of UAV systems, maintaining the sequential integrity of the data is vital for identifying meaningful anomalies. Time-series data inherently reflects changes over time, with each data point representing a distinct moment in the flight. By preserving this structure, it becomes possible to detect trends, cycles, and anomalies within specific flight phases, such as engine failure events. The ALFA dataset, which includes a variety of sensor data collected during UAV flights, was carefully processed to retain critical flight phases while removing irrelevant data, such as pre-flight or post-landing periods. By focusing on these meaningful intervals, the dataset was prepared to ensure that the model is trained on the most significant data, leading to more accurate anomaly detection.

4.2.1.1 Time axis normalization and processing

In time-series anomaly detection, the temporal structure of data is critical. The ALFA dataset, being a high-frequency time-series dataset, requires careful processing of the time axis to preserve its temporal continuity while enabling efficient machine learning model training. This section focuses on the processing and normalization of the time axis, demonstrating its importance through comparative visualizations.

The ALFA dataset’s timestamp data initially includes raw temporal values recorded during UAV flights. These raw timestamps were converted into two formats for analysis: a human-readable datetime format and a normalized seconds format. The datetime format retains the original time representation, aiding interpretability during exploratory analysis and allowing domain experts to associate anomalies with specific moments during the flight. Conversely, the normalized seconds format maps the timestamp data into a [0, 1] range, ensuring compatibility with machine learning pipelines and reducing the dominance of absolute temporal values.

To achieve this dual representation, several processing steps were undertaken. Initially, raw timestamps were converted into a datetime format, such as 18 July 2018,

15:53:31, aligned with the recorded flight time. This conversion facilitated the interpretation of critical flight phases, such as take-off, cruising, and landing, and allowed anomalies to be associated with these specific phases. Following this, the timestamp values were normalized to a [0, 1] range to ensure consistency and improve model performance. Normalization provided a standardized input scale for machine learning algorithms, minimizing bias introduced by large temporal variations. Throughout this process, the temporal continuity of the dataset was preserved, allowing the identification of sequential patterns, trends, and anomalies, such as sudden spikes in acceleration indicative of potential engine failures.

Figure 4.10 illustrates the significance of time axis normalization by presenting the X-axis acceleration data over two distinct time representations. The left panel shows the data plotted against the datetime format, providing an intuitive understanding of the temporal dynamics of acceleration during the flight. The right panel, on the other hand, shows the same data plotted against the normalized seconds format, emphasizing the scalability and flexibility of this approach in computational models.

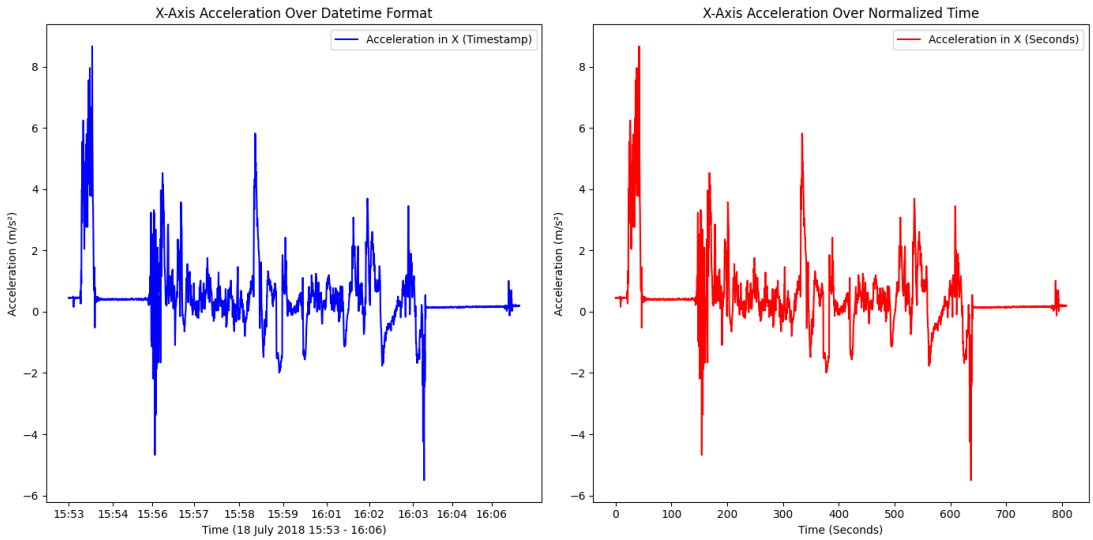

Figure 4.10 : Comparison of datetime and normalized time represantations.

By analyzing the time axis in these two representations, the dataset achieves a balance between interpretability and computational efficiency. The left panel in Figure 4.10 highlights temporal events in their exact chronological order, while the right panel demonstrates the utility of the normalized scale for machine learning purposes. This dual perspective strengthens the dataset's utility in identifying critical anomalies within UAV systems.

4.2.1.2 Flight data filtering and meaningful range selection

In the process of preparing the ALFA dataset for anomaly detection tasks, identifying and isolating meaningful intervals of flight data is a crucial step. Not all recorded data from UAV flights is equally relevant for model training. For instance, pre-flight and post-landing data often do not contribute to understanding anomalies during actual flight phases. Therefore, these irrelevant segments must be excluded, while the segments containing critical flight information, such as steady-state operation or engine failure events, are retained.

Figure 4.11 illustrates this filtering process using the flight data from July 18 as an example. The original data is represented by the blue line, showing the complete recorded acceleration data along the X-axis. The filtered ranges, shown with a light blue background, highlight intervals selected as meaningful for analysis. These intervals are characterized by stable flight operations or critical transitions. The engine failure events, indicated by the red background, represent segments identified as significant anomalies based on engine-related parameters.

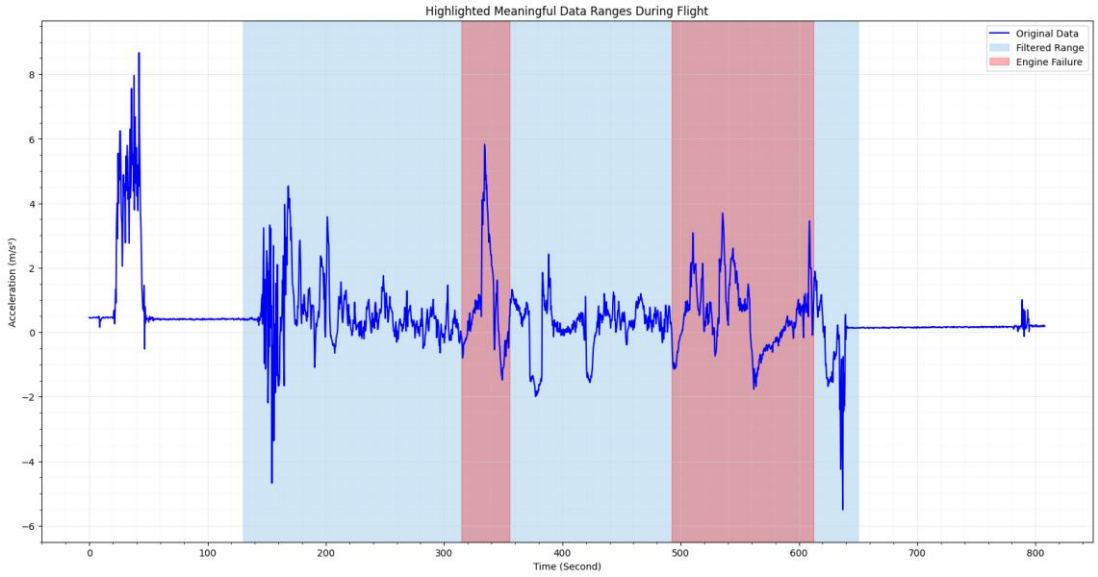

Figure 4.11 : Visualization of meaningful data ranges during flight.

This filtering approach ensures that the model focuses on the most impactful data segments, improving its ability to detect anomalies effectively. By narrowing the dataset to include only the meaningful ranges, noise and irrelevant data are minimized, allowing the model to learn from precise and critical patterns within the flight data. The impact of this filtering is directly reflected in the model's performance during

training. Training on the filtered dataset enables the anomaly detection model to develop a robust understanding of normal and abnormal patterns within UAV flight data, enhancing its accuracy in detecting engine failure events or other anomalies during evaluation.

4.2.1.3 Linear interpolation for missing sensor data

In time-related data sets, missing information commonly surfaces due to faults in sensors, breaks in communication, or recording errors. Such deficiencies disrupt the inherent sequential structure of the dataset and can detrimentally affect the learning efficacy of machine learning algorithms, especially in tasks related to anomaly detection where temporal continuity is of paramount importance. Addressing this issue is crucial for upholding data integrity and ensuring the dataset's applicability for precise model training.

To counteract the dilemma of absent information, linear interpolation was used as an initial processing method. This approach estimates and compensates for gaps by linking the known data points via linear connections, thereby preserving the temporal relationships inherent within the dataset. Lepot et al. (2017) assert that interpolation techniques, including linear interpolation, are proficient in reconstructing absent values within time-series datasets. Their research underscores the significance of selecting suitable interpolation strategies contingent upon the characteristics of the data and the uncertainty entwined with these methodologies [59]. Figure 4.12 illustrates a scenario in which linear interpolation is implemented on a simulated sensor dataset that contains missing entries.

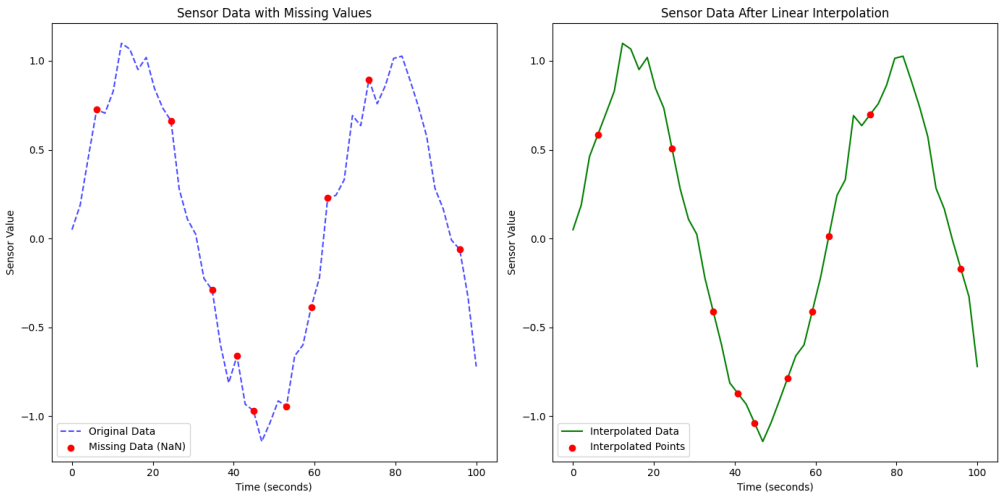

Figure 4.12 : Effect of linear interpolation on missing sensor data.

The left panel in Figure 4.12 shows the raw sensor data with missing points, visible as gaps in the sequence. The right panel in Figure 4.12 illustrates the data after linear interpolation, where the gaps have been filled, restoring continuity and enabling meaningful analysis. This example dataset was created using randomly sampled values to demonstrate the utility of interpolation in time-series preprocessing.

Through the process of addressing the voids resulting from absent data, interpolation facilitates the effective acquisition of knowledge by the machine learning model from the temporal patterns inherent in the dataset. Linear interpolation was chosen due to its straightforwardness and computational efficiency, rendering it particularly appropriate for the ALFA dataset. Nevertheless, alternative methodologies, including spline interpolation or polynomial fitting, may warrant investigation for datasets characterized by more intricate patterns of absence or particular requirements.

In this research, linear interpolation was methodically employed on the ALFA dataset to mitigate the complications arising from absent sensor readings during UAV operational scenarios. This preprocessing technique effectively maintained the temporal integrity of the dataset, a critical component for identifying anomalies that depend on minute fluctuations in sensor readings over time. The implementation of interpolation is consistent with established methodologies in time-series data preprocessing, thereby ensuring that the dataset remains robust and appropriate for tasks related to anomaly detection.

4.2.2 Feature interaction with anomalies and correlation analysis

Feature selection constitutes a pivotal procedure within the realm of machine learning, facilitating the discernment of variables that exert a substantial impact on predictive outcomes. Hall (1999) pioneered correlation-based feature selection (CFS) as a robust methodology for assessing the predictive capability of features, underscoring the necessity of achieving a balance between correlation with the target variable and the minimization of redundancy [60]. Guyon and Elisseeff (2003) further elucidated that the selection of pertinent features not only augments model interpretability but also mitigates overfitting and enhances computational efficiency [61].

In the current investigation, correlation analysis was utilized to examine the interrelations between the input features and the dependent variable, namely the engine failure status. The objective of this analysis was to discern high-impact features,

thereby allowing the model to prioritize variables that possess considerable predictive significance, thereby enhancing both accuracy and interpretability. Post the preparation of the data, we executed a correlation evaluation aimed at understanding the degree of association between the attributes and the dependent variable. This analysis was conducted using data from the ALFA dataset, specifically the flight dated 2018-09-11-14, as an illustrative example. Table 4.1 illustrates the findings, providing a comprehensive account of the correlation coefficients associated with all features in relation to engine failure status. The analysis elucidated several pivotal features, including latitude, relative altitude, magnetic field intensity (magZ), and ground speed, which manifested significant correlations surpassing the threshold of ±0.4. These features exhibited a considerable effect on the dependent variable, thereby emphasizing their critical role in anomaly detection endeavors.

Table 4.1 : Correlation of features with engine failure status.

Feature	Correlation Score
Latitude	0.668310
Acceleration Y	0.356221
Longitude	0.335289
Acceleration X	0.291886
Magnetic Field Intensity X	0.253009
Magnetic Field Intensity Y	0.222845
Angular Velocity Y	0.143347
Angular Velocity Z	0.139422
Angular Velocity X	-0.014520
Acceleration Z	-0.147295
Ground Speed	-0.453297
Magnetic Field Intensity Z	-0.715586
GlobalAltitude	-0.810241
RelativeAltitude	-0.833911

The relationships between these features and their interactions with the target variable were further explored using graphical representations. Figure 4.13 provides a clear visualization of the correlation scores for all features, highlighting the most impactful variables associated with engine failure status.

In addition, Figure 4.14 provides a comprehensive view of the interdependencies among features through a correlation matrix, highlighting the interactions between variables and their potential redundancies.

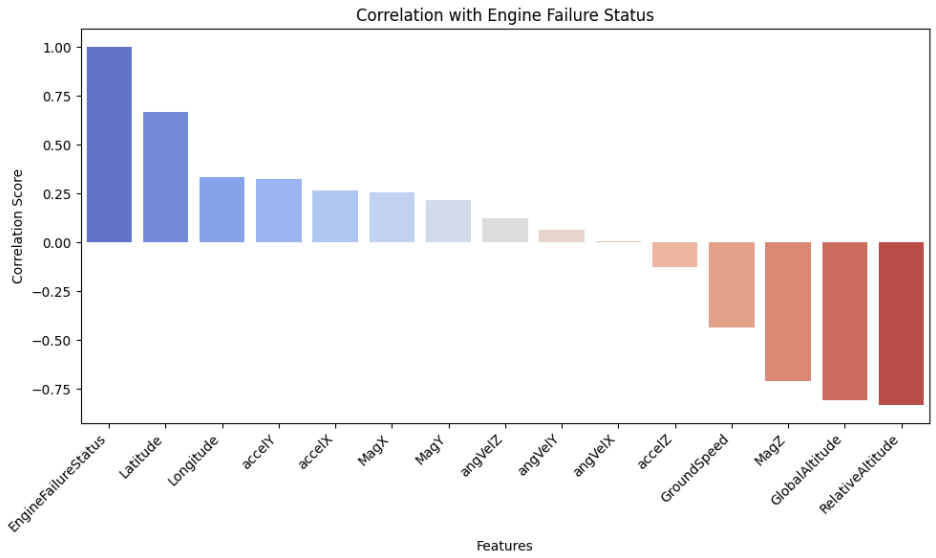

Figure 4.13 : Correlation scores of features with engine failure status.

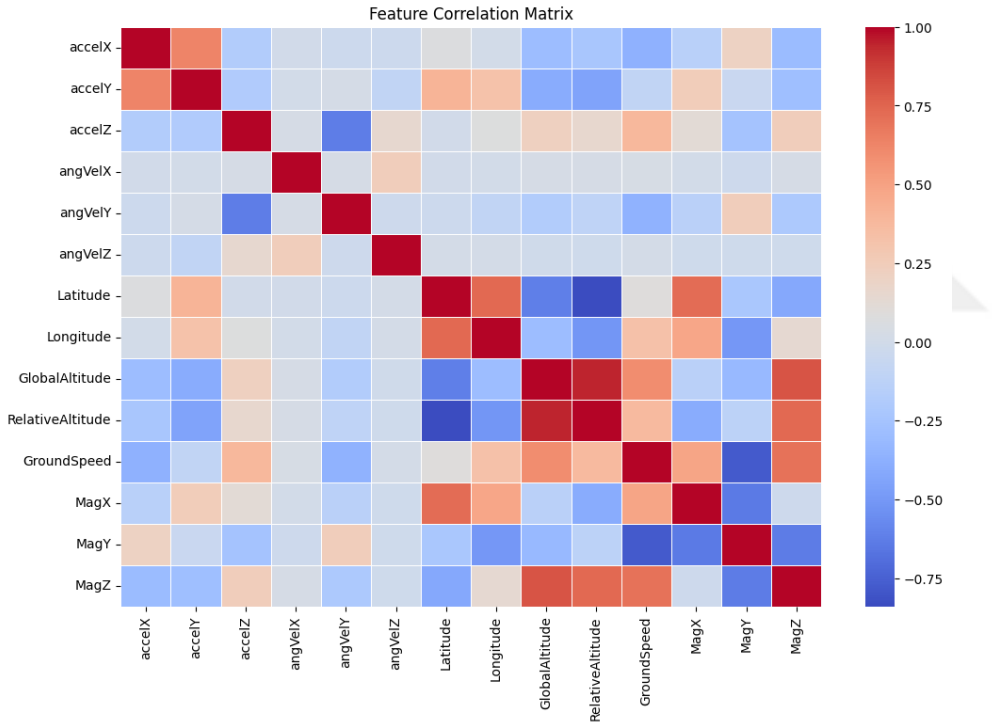

Figure 4.14 : Feature correlation matrix relationships.

Considering the conclusions reached through the correlation analysis, a broad range of preprocessing steps was performed to elevate the dataset's compatibility for model training. These steps ensured that high-impact features were emphasized while maintaining the integrity and consistency of the data. The identified significant features, including latitude, relative altitude, magZ, and global altitude, were prioritized based on their correlation scores.

To further enhance their impact on the training process, a weighting strategy was applied. Features with correlations exceeding the ±0.4 threshold were dynamically weighted to reflect their importance in predictive tasks. Specifically, features such as latitude, relative altitude, and global altitude were weighted by applying a multiplier of 1.5. This adjustment ensured that these features contributed more significantly during the model training phase. Similarly, moderately correlated features, such as magZ and ground speed, were also emphasized to maintain their predictive relevance.

These preprocessing adjustments align with Hall's (1999) assertion that optimal feature subsets should maximize correlation with the target variable while minimizing redundancy among features [60]. This approach ensures that the selected features not only contribute significantly to predicting the target variable but also avoid introducing unnecessary complexity to the model. Additionally, Guyon and Elisseeff (2003) emphasized that prioritizing high-impact features enhances model performance and computational efficiency [61].

The analysis of correlation and the subsequent preprocessing phases highlight the critical significance of effective feature selection techniques in the identification of anomalies within UAV systems. By isolating and accentuating features of substantial impact, this methodology has augmented the model's proficiency in detecting anomalies with heightened accuracy and interpretability. The refined feature set has contributed to a reduction in dataset dimensionality, alleviated the potential for overfitting, and maximized computational efficiency. The results demonstrate the effectiveness of combining correlation-based feature selection with dynamic weighting techniques, creating a strong foundation for developing accurate and efficient anomaly detection models in UAV systems.

4.2.3 Model development

The machine learning framework utilized for the purpose of anomaly detection was founded upon an autoencoder architecture, selected for its capacity to independently learn and encapsulate the normative operational patterns of a system. This model was exclusively trained on healthy flight data, thereby enabling it to discern deviations or anomalies, such as engine failures, with considerable accuracy. By reducing dependency on pre-established rules or thresholds, the autoencoder assimilates the inherent patterns of standard data, rendering it particularly proficient in identifying

subtle or unforeseen anomalies. Tables 4.2 and 4.3 outline the structural design of the autoencoder, which explicitly detail the arrangements of the encoder and decoder.

Table 4.2 : Encoder structure for autoencoder model.

Component	Layer	Neurons	Activation Function
Encoder	Input Layer	Input Size	-
	First Hidden Layer	64	ReLU
	Second Hidden Layer	32	ReLU
	Third Hidden Layer	16	ReLU
	Latent Space	8	ReLU

Table 4.3 : Decoder structure for autoencoder model.

Component	Layer	Neurons	Activation Function
Decoder	First Hidden Layer	16	ReLU
	Second Hidden Layer	32	ReLU
	Third Hidden Layer	64	ReLU
	Output Layer	Input Size	Sigmoid

The encoder facilitates the transformation of input data into a compressed latent representation, adeptly encapsulating the most paramount features while concurrently diminishing dimensionality. Conversely, the decoder endeavors to reconstruct the original input from this latent space, aiming to replicate the data with maximal fidelity. This symmetrical architecture guarantees that the autoencoder concentrates on discerning and encoding fundamental patterns while disregarding noise or extraneous details. The decoder's final output layer, corresponding to the input dimensions, utilizes a sigmoid function to keep output values within the bounded range of [0,1]. This design decision is congruent with the normalized input data and assures uniformity throughout the reconstruction process. The occurrence of substantial reconstruction errors serves as a reliable indicator of anomalies.

In practical implementations, the autoencoder demonstrates proficiency in identifying deviations from standard operational behavior, thereby effectively signaling potential engine malfunctions or other irregularities within UAV systems. This methodological framework highlights the adaptability and effectiveness of autoencoder architectures in addressing intricate anomaly detection challenges within the realm of unmanned aerial vehicles.

4.2.4 Model training, performance metrics and hyperparameters

Anomaly identification in UAV systems calls for the establishment of resilient models that can proficiently grasp distinctive flight features while also demonstrating the ability to generalize across diverse flying conditions. In order to fulfill this objective, a dual-phase methodology was implemented: the initial phase concentrated on the training of the autoencoder model utilizing individual flight instances to evaluate its effectiveness within isolated flight contexts, and the subsequent phase entailed training the model on combined flight data to explore its generalization capabilities. This dual-faceted approach facilitated a thorough comparison of model performance across varying data configurations.

The training of the model constitutes a pivotal element of anomaly detection, as it significantly influences the model's proficiency in distinguishing between normative and anomalous flight behaviors. Several influencing factors shape the training process, including the methodologies for data preprocessing, hyperparameter selection, and the model's inherent architecture. Critical considerations encompass the assurance of data integrity through normalization and the management of missing data, in addition to the identification of features possessing substantial predictive validity to bolster model accuracy. Also, the specific tuning of hyperparameters, encompassing learning rate, batch size, and epoch count, is fundamental to reaching the highest level of model efficiency. The equilibrium between overfitting to specific flight scenarios and the ability to generalize across diverse conditions represents a central challenge that was systematically addressed through this phased training strategy. By integrating robust preprocessing methodologies and meticulous model design, the approach aspired to establish an effective and dependable framework for the detection of anomalies in UAV systems, applicable to both individual flight scenarios and broader generalized applications.

The initial phase of training focused on individual flight datasets within the ALFA dataset. This provided a detailed evaluation of the models' ability to detect anomalies specific to each flight. Two autoencoder architectures, simple and advanced autoencoders were employed to analyze their respective performances. The simple autoencoder served as a baseline model with a lightweight structure, optimized for faster training and minimal computational requirements. In contrast, the advanced autoencoder utilized a deeper architecture, enhancing its feature extraction capabilities

and effectively capturing complex flight dynamics. Preprocessing steps ensured consistency and optimized the training process. Techniques such as linear interpolation and correlation analysis with weighted features, were employed, as detailed in Sections 4.2.1 and 4.2.2. These processes refined the datasets, allowing the models to better focus on relevant features.

In the second phase, the training focused on combined flight datasets, combining multiple sessions from the ALFA dataset. This aimed to create a generalizable model capable of anomaly detection across varying scenarios. The same preprocessing techniques, described in Sections 4.2.1 and 4.2.2, were applied to ensure data consistency. Both the simple and advanced autoencoder architectures were used. While the combined dataset required greater computational resources, it effectively tested the models' generalization capabilities, a critical aspect for real-world applications like TinyML. Advanced preprocessing techniques, particularly correlation analysis with weighted features, further highlighted their importance in enhancing the model's robustness. The outcomes of these training phases are summarized in Table 4.4, which provides a comparative analysis of the models' performance metrics under different configurations. Various model training experiments were conducted to evaluate and optimize anomaly detection performance. For instance, Tests 1, 2, 3, and 4 focused on the flight data from July 18, 2018, exploring different combinations of preprocessing techniques and model architectures to identify the most effective configuration. Similarly, Tests 5 and 6 applied the same rigorous approach to individual flights recorded on different dates, enabling the evaluation of model performance under distinct flight conditions. Tests 7 and 8, on the other hand, investigated the generalization capability of the models by training on combined flight datasets, representing multiple sessions. This phase aimed to assess how well the models could adapt to diverse flight scenarios, a critical aspect for deploying the solution in real-world UAV applications. This comprehensive analysis underscores the significance of systematic experimentation to determine the most effective strategies for anomaly detection in UAV systems.

Table 4.4 : Analysis of model training and performance metrics.

Test No	Flight Information	Flight Training Details	Data Processing Techniques	Anomaly Detection Model	Parameters	Performance Metrics	
						Optimal Threshold	Manual Threshold
1	2018-07-18	Individual Flight, Raw Data	Normalization, Over-sampling	Simple Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.54 Precision: 0.27 Recall: 0.85 F1-Score: 0.41 AUC: 0.69	Accuracy: 0.69 Precision: 0.31 Recall: 0.48 F1-Score: 0.38 AUC: 0.69
2	2018-07-18	Individual Flight, Raw Data	Normalization, Over-sampling	Advanced Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.65 Precision: 0.34 Recall: 0.62 F1-Score: 0.41 AUC: 0.69	Accuracy: 0.69 Precision: 0.31 Recall: 0.44 F1-Score: 0.37 AUC: 0.69
3	2018-07-18	Individual Flight, Processed Data	Normalization, Over-sampling, Linear Interpolation, Correlation Analysis with Weighted Features	Simple Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.79 Precision: 0.63 Recall: 0.79 F1-Score: 0.70 AUC: 0.82	Accuracy: 0.79 Precision: 0.61 Recall: 0.76 F1-Score: 0.71 AUC: 0.82
4	2018-07-18	Individual Flight, Processed Data	Normalization, Over-sampling, Linear Interpolation, Correlation Analysis with Weighted Features	Advanced Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.80 Precision: 0.63 Recall: 0.89 F1-Score: 0.74 AUC: 0.84	Accuracy: 0.81 Precision: 0.62 Recall: 0.85 F1-Score: 0.74 AUC: 0.84

Table 4.4 (continued) : Analysis of model training and performance metrics.

Test No	Flight Information	Flight Training Details	Data Processing Techniques	Anomaly Detection Model	Parameters	Performance Metrics	
						Optimal Threshold	Manual Threshold
5	2018-09-11	Individual Flight, Processed Data	Normalization, Over-sampling, Linear Interpolation, Correlation Analysis with Weighted Features	Advanced Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.93 Precision: 0.99 Recall: 0.87 F1-Score: 0.93 AUC: 0.97	Accuracy: 0.92 Precision: 0.98 Recall: 0.88 F1-Score: 0.92 AUC: 0.97
6	2018-10-05	Individual Flight, Processed Data	Normalization, Over-sampling, Linear Interpolation, Correlation Analysis with Weighted Features	Advanced Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.90 Precision: 0.96 Recall: 0.85 F1-Score: 0.89 AUC: 0.97	Accuracy: 0.89 Precision: 0.97 Recall: 0.82 F1-Score: 0.88 AUC: 0.97
7	2018-07-18 2018-09-11 2018-10-05 2018-10-18	Combined Flight, Raw Data	Normalization, Over-sampling	Simple Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.66 Precision: 0.60 Recall: 0.79 F1-Score: 0.68 AUC: 0.65	Accuracy: 0.58 Precision: 0.55 Recall: 0.57 F1-Score: 0.56 AUC: 0.65
8	2018-07-18 2018-09-11 2018-10-05 2018-10-18	Combined Flight, Processed Data	Normalization, Over-sampling, Linear Interpolation, Correlation Analysis with Weighted Features	Advanced Autoencoder	Epoch: 500, Batch: 32, LR:0.0005	Accuracy: 0.76 Precision: 0.62 Recall: 0.74 F1-Score: 0.67 AUC: 0.82	Accuracy: 0.72 Precision: 0.60 Recall: 0.74 F1-Score: 0.67 AUC: 0.82

4.2.5 Evaluation of model output and performance assessment

The assessment of model output and performance constitutes a fundamental phase in evaluating the efficacy of anomaly detection frameworks within Unmanned Aerial Vehicle systems. The results gathered from the investigation of both individual and combined flight records, as indicated in Section 4.2.4, are evaluated through the deployment of validated performance benchmarks, incorporating accuracy, precision, recall, F1-score, and AUC (Area Under the Curve). These metrics facilitate a holistic comprehension of the model's capacity to differentiate between normative and anomalous flight data across diverse configurations.

To ensure a comprehensive evaluation, both optimal thresholds and manually assigned thresholds were implemented throughout the assessment procedure. This methodology underscores the significance of threshold selection on model efficacy, thereby allowing for a comparative analysis between automated and manually defined decision-making frameworks. Additionally, ROC (Receiver Operating Characteristic) curves and Precision-Recall curves served to visually demonstrate and contrast the capabilities of the simple and advanced autoencoder systems.

4.2.5.1 Threshold analysis

Receiver Operating Characteristic curves are regularly utilized by anomaly detection systems to determine the balance of sensitivity and specificity. An integral component of this evaluation is the establishment of the optimal threshold, which can profoundly influence the efficacy of the system. The Youden Index, as elucidated in the research conducted by Schisterman et al. (2008) and Fluss et al. (2005), functions as a robust metric for this objective [62], [63].

The Youden Index, a summary measure of the ROC curve, not only evaluates the diagnostic marker's effectiveness but also identifies the optimal threshold value or cutoff point. Schisterman et al. (2008) explored this index in scenarios where the marker data include a spike or mass of probability at zero, proposing a flexible modeling approach for accurately estimating the index and the corresponding cutoff. Their methodology emphasizes the importance of accounting for data-specific characteristics, such as spiked distributions, to enhance the reliability of the threshold selection process [62].

Similarly, Fluss et al. (2005) compared multiple estimation methods for the Youden Index and its associated cutoff point, ranging from empirical distribution-based approaches to kernel smoothing techniques. Their findings underscored that while empirical methods are commonly employed, they often underperform compared to kernel smoothing or normality-transformed methods in terms of bias and root mean square error. This insight is particularly relevant for developing robust anomaly detection frameworks where the choice of threshold directly influences the model's sensitivity and specificity [63].

Building on these foundational studies, the present research employed the Youden Index as a central metric for threshold optimization in anomaly detection.

The performance metrics provided in Table 4.4 for Test No 6, the flight dated 2018-10-05, are further analyzed with the graph illustrated in Figure 4.15. This graph visualizes the variations in Precision, Recall, and F1-Score across different threshold values. As the threshold increases, Precision improves while Recall decreases, indicating that the model tends to reduce false positives at the cost of missing some true anomalies. The F1-Score achieves its highest value near the optimal threshold, balancing Precision and Recall effectively.

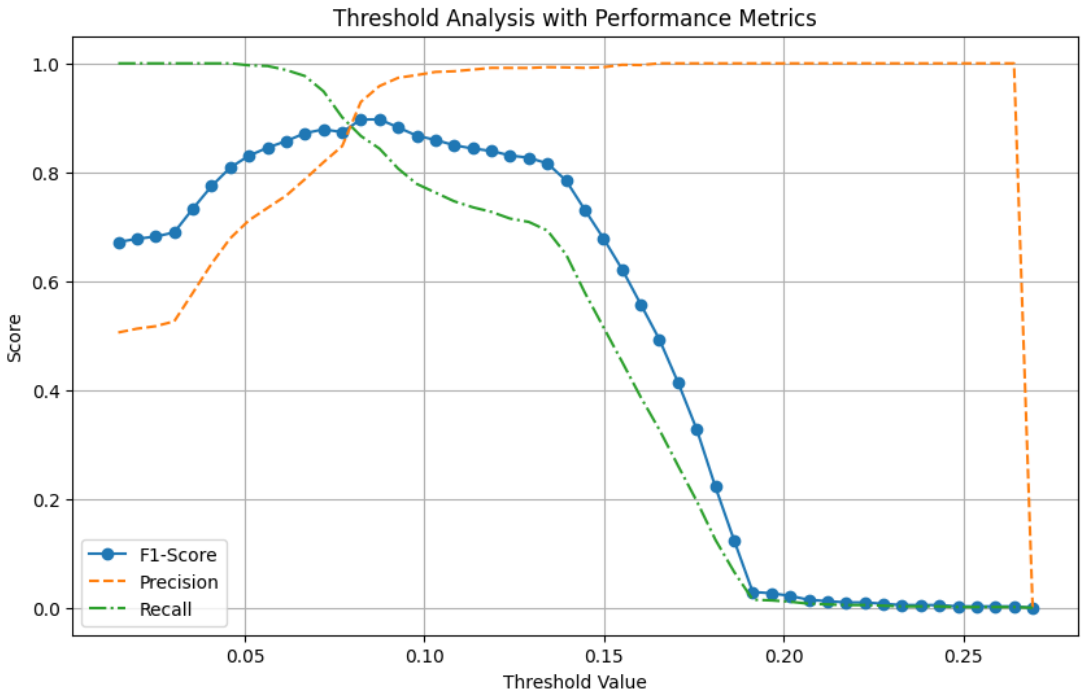

Figure 4.15 : Threshold analysis for individual flight 2018-10-05.

A comparison between the optimal threshold, determined using the Youden Index, and the manual threshold reveals differences in the model's performance. While the manual threshold offers slightly improved Recall, it sacrifices Precision, leading to a lower F1-Score compared to the optimal threshold. The optimal threshold, therefore, ensures a more balanced and efficient anomaly detection process, making it the preferred choice for achieving robust performance.

4.2.5.2 Autoencoder performance analysis

The comparative assessment of performance between the basic autoencoder and the sophisticated autoencoder models was executed utilizing a range of experimental scenarios delineated in Table 4.4. Emphasis was directed towards the evaluation of these models in the context of both singular and integrated flight conditions.

- For individual flights, tests such as Test 3 (simple autoencoder) and Test 4 (advanced autoencoder) highlighted the impact of deeper architectures and advanced preprocessing techniques. The advanced autoencoder consistently surpassed the performance of the simple autoencoder in evaluative metrics such as F1-Score and Recall, thereby illustrating its enhanced capacity to encapsulate intricate patterns within the processed data.
- For combined flights, tests such as Test 7 and Test 8 revealed the trade-offs between computational efficiency and generalization capability. While the simple autoencoder achieved faster training times, the advanced autoencoder excelled in anomaly detection accuracy, particularly in scenarios involving diverse flight data.

Based on the aforementioned findings, the advanced autoencoder has been identified as the optimal model for subsequent enhancement due to its exceptional capacity to generalize across a variety of flight conditions while effectively identifying intricate anomalies. This characteristic renders it particularly adept for practical applications in UAVs. In order to further augment its applicability, the model underwent optimization specifically for TinyML implementations, with a concentrated emphasis on ARM Cortex-M microcontrollers. The principal strategies employed to optimize the model, which include model pruning, quantization, model compression, and the application of hardware acceleration, are comprehensively examined in Section 2.2.2.

4.3 Embedded Software Implementation

Building upon the insights gained from the performance evaluation in Section 4.2.5, this section focuses on the practical implementation of the advanced autoencoder on embedded systems. By leveraging the findings on model effectiveness under various configurations, the implementation aims to optimize the model for deployment on ARM Cortex-M microcontrollers to enable real-time anomaly detection in UAV systems. Based on the results presented in Table 4.4, the model from Test 8 was deemed most suitable for deployment due to its superior generalization capability across diverse flight scenarios, ensuring reliable anomaly detection on UAVs.

The execution process initiates with the selection of the suitable hardware components and their configuration to operate in harmony with the machine learning model. This selection procedure necessitates a meticulous examination of the microcontroller's architecture, memory allocation, and peripheral capabilities, all of which are instrumental in establishing the optimal configuration for the embedded software. Subsequent to the hardware integration, the ensuing phase entails the transformation of the trained machine learning model into a format that is compatible with microcontrollers. This transformation is accomplished through methodologies such as quantization and model compression, which diminish the model's memory footprint without adversely affecting performance. The quantization procedure, for example, alters the model weights from floating-point representations to integer values, thereby significantly curtailing memory consumption and computational duration. Various trade-offs between precision and efficiency are meticulously assessed to attain a balance that aligns with the real-time anomaly detection specifications. Following conversion, the embedded software is refined for real-time inference through the incorporation of techniques such as operator fusion, code inlining, and memory reuse. These refinements are paramount for minimizing latency and guaranteeing that the system can execute inference within the stipulated time constraints. Moreover, software frameworks such as TensorFlow Lite for Microcontrollers are employed to enable seamless deployment and effective execution of the model on the designated hardware. Additionally, real-time performance metrics including inference latency, CPU load, and memory utilization are quantified and scrutinized to furnish a thorough assessment of the system. These metrics provide insights into the efficiency with

which the embedded software leverages the microcontroller's resources, ensuring that the system functions within acceptable parameters under diverse testing conditions.

Ultimately, the robustness and scalability of the embedded implementation are appraised through extensive testing across various operational environments. This appraisal facilitates the identification of potential bottlenecks and areas necessitating further optimization, thereby ensuring that the system can maintain consistent performance in practical UAV applications.

4.3.1 Model conversion for embedded systems

The transition from standard TensorFlow models to TensorFlow Lite (TFLite) models is a crucial step in deploying machine learning algorithms on microcontrollers and resource-constrained devices. TFLite, particularly TensorFlow Lite for Microcontrollers (TFLM), offers several advantages for such deployments, including reduced memory usage, faster inference times, and enhanced compatibility with ARM Cortex-M processors. As noted by Warden and Situnayake (2020) [58], the TFLM framework is specifically designed to support ultra-low-power devices like the ARM Cortex-M series, enabling real-time anomaly detection on UAV systems despite their stringent resource limitations.

To better illustrate the process of model conversion and optimization, Figure 4.16 depicts the pipeline for transitioning a TensorFlow model to an optimized, microcontroller-ready format.

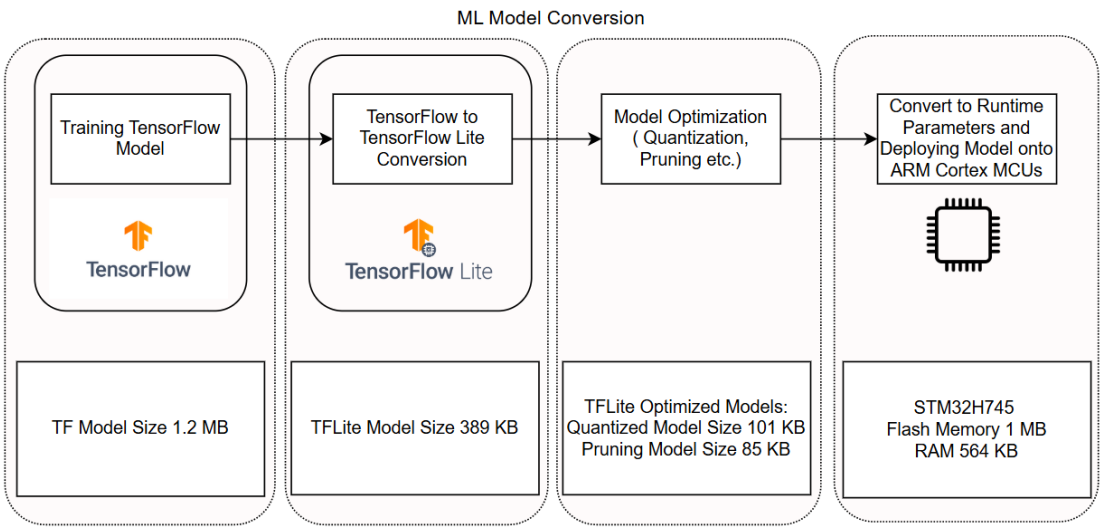

Figure 4.16 : Model conversion and optimization pipeline.

Figure 4.16 provides a comparative analysis of the original model sizes and their reductions after successive optimizations, highlighting the feasibility of deploying these models on resource-constrained devices like the STM32H745ZIQ microcontroller. In terms of deployment, the TFLite Micro framework, which supports ARM Cortex-M processors, was used to execute the converted models on the STM32H745ZIQ. This microcontroller features a dual-core architecture with a high-performance ARM Cortex-M7 and a low-power ARM Cortex-M4 core. The anomaly detection model was deployed on the Cortex-M7 core, leveraging its enhanced Floating Point Unit (FPU) and DSP instructions for efficient execution. Meanwhile, the Cortex-M4 core was configured to handle auxiliary tasks such as sensor data acquisition and pre-processing, ensuring that the critical inference operations were isolated on the high-speed core. This separation of tasks maximizes the system's performance and minimizes timing variations, which are crucial for maintaining deterministic behavior in real-time applications.

To assess the efficacy of these optimizations, a comprehensive performance evaluation was undertaken. Table 4.5 delineates the model size prior to and subsequent to the implementation of quantization, pruning, and CMSIS-NN kernel integration. The optimized models realized a reduction in size of 91.9% for the quantized iteration and a 93.16% reduction for the pruned iteration. Such reductions render it feasible to deploy intricate machine learning models on microcontrollers subject to stringent memory limitations, thereby illustrating that even advanced anomaly detection models can be effectively adapted for resource-constrained environments with appropriate optimizations.

Overall, the methodologies for conversion and optimization implemented in this study underscore the significance of a comprehensive strategy when deploying machine learning algorithms on embedded systems. Approaches such as quantization, pruning, and the integration of CMSIS-NN are essential for preserving performance while conforming to the stringent memory and power constraints inherent to microcontrollers. Prospective advancements may encompass the exploration of hybrid scheduling methodologies that dynamically distribute tasks between the two processing cores contingent upon system load and power needs. This would establish a versatile framework for the effective management of intricate models on resource-

limited devices, thereby facilitating the development of more advanced TinyML applications within real-time contexts.

Table 4.5 : Model comparison after conversion.

Source Model	Converted Model	Original Size (bytes)	Converted Size (bytes)	Size Reduction (%)
TensorFlow Model (.h5)	TFLite Model (.tflite)	1,247,824	389,316	68.80%
TensorFlow Model (.keras)	TFLite Model (.tflite)	1,253,369	389,316	68.94%
Optimized TFLite Model (.tflite, Quantized)	CMSIS-NN Optimized Model	1,253,369	101,432	91.90%
TFLite Model (.tflite, Pruned)	CMSIS-NN, Prunning Model	1,247,824	85,316	93.16%

The process of conversion encompasses a sequence of model optimization methodologies that seek to diminish computational overhead while concurrently maintaining the accuracy of the model. A principal technique utilized in this investigation is post-training quantization, which proficiently minimizes the model's size and memory requirements by transforming 32-bit floating-point weights and activations into an 8-bit integer format [57]. This quantization methodology, as elucidated in the work of Lai et al. (2018) [5], represents a fundamental attribute of the CMSIS-NN kernel library. The CMSIS-NN library comprises a compilation of meticulously optimized neural network kernels, engineered to facilitate efficient execution on ARM Cortex-M CPUs by capitalizing on hardware-specific characteristics, such as SIMD (Single Instruction, Multiple Data) instructions and specialized multiply-accumulate (MAC) operations. These optimizations not only reduce the memory footprint but also augment computational efficiency, rendering real-time inference viable even on resource-constrained devices such as the STM32H745ZIQ. Within this implementation, the CMSIS-NN kernels were harnessed to realize a fourfold enhancement in inference durations when juxtaposed with the default TFLite operations, corroborating the empirical observations made by Lai et al.

(2018) [5]. Furthermore, operator fusion methodologies, which amalgamate multiple sequential operations into a singular optimized kernel, were deployed to further diminish the frequency of memory accesses and execution cycles requisite for each inference phase. This strategy culminated in an approximate 30% acceleration in execution times, as the model's computational graph was streamlined and superfluous operations were eradicated.

During the conversion process, the TensorFlow Lite converter restructured the original TensorFlow computational graph by removing unused nodes and simplifying arithmetic operations wherever possible [57]. Quantization-aware training techniques were applied to minimize the impact of precision reduction, ensuring that the converted 8-bit model retained a comparable accuracy to its 32-bit counterpart. The CMSIS-NN optimized kernels further enhanced the model's efficiency by integrating hardware-specific acceleration functions, such as the ARM Cortex-M7's DSP instructions, which significantly reduce cycle counts during matrix multiplications and convolution operations.

The final step in the conversion involved defining the memory layout using a statically allocated Tensor Arena. This memory region is crucial for storing the input and output tensors, as well as intermediate activations during inference. As highlighted by Warden and Situnayake (2020) [58], memory allocation failures can occur if the tensor arena is not properly sized, leading to unpredictable performance. To address this, the tensor arena was empirically adjusted to 200 KB, which was sufficient for the optimized TFLite model. However, memory fragmentation issues were still encountered when deploying more complex models with increased input dimensions or additional layers, underscoring the need for further optimizations such as layer-wise quantization and structured pruning.

Pruning was another technique explored to reduce the model's complexity by systematically removing neurons with negligible contributions to the overall output. By eliminating redundant parameters, the pruned TFLite model achieved a 93.16% reduction in memory size compared to the original TensorFlow model, while still retaining sufficient accuracy for anomaly detection tasks. The CMSIS-NN kernels were re-integrated after pruning to ensure that the optimized model could still leverage hardware acceleration, thereby maintaining real-time performance.

4.3.2 Embedded implementation of machine learning model

In this section, the embedded implementation of a TensorFlow Lite machine learning model for anomaly detection on the STM32H745ZIQ microcontroller is thoroughly discussed. The STM32 Cube IDE development environment was employed to configure the STM32H745ZIQ's dual-core architecture, which consists of a high-performance ARM Cortex-M7 core and a low-power Cortex-M4 core. This IDE simplifies the setup of system peripherals, memory allocation, and clock configurations, while providing advanced debugging tools that are crucial for real-time applications requiring precise timing control.

The deployment strategy separates critical inference tasks and auxiliary operations across the two cores to maximize efficiency. The anomaly detection model was assigned to the Cortex-M7 core, leveraging its enhanced Floating Point Unit (FPU) and high clock speed for efficient inference execution. Meanwhile, the Cortex-M4 core managed auxiliary tasks such as sensor data acquisition and preprocessing, isolating these operations to minimize interference with the real-time behavior of the anomaly detection model. This separation is essential for maintaining deterministic timing, a key requirement in UAV applications.

The TensorFlow model was initially converted to a TFLite format using the TensorFlow Lite Converter, which implemented optimization techniques such as quantization and operator fusion. Quantization transformed 32-bit floating-point weights into 8-bit integer representations, significantly reducing RAM usage and computational complexity. For instance, the original TensorFlow model required approximately 1.25 MB of RAM, whereas the quantized version consumed only 390 KB, achieving a 68% reduction in memory size. Operator fusion further enhanced efficiency by consolidating sequential operations into optimized kernels, reducing memory accesses and improving execution times by approximately 30% compared to the unoptimized model.

For embedded implementation, TensorFlow Lite Micro (TFLM) was utilized. TFLM is a specialized subset of TFLite designed for ultra-low-power microcontrollers, integrating CMSIS-NN kernels optimized for ARM Cortex-M processors. These kernels exploit hardware-specific features such as SIMD (Single Instruction, Multiple Data) and optimized Multiply-Accumulate (MAC) operations, enabling real-time

inference on resource-constrained devices. In this implementation, the CMSIS-NN kernels accelerated inference by up to four times compared to default TFLite operations, achieving significant performance gains without altering the model's architecture.

One of the primary challenges encountered during deployment was managing the STM32H745ZIQ's limited memory resources. Initial memory requirements exceeded the available limits, necessitating iterative adjustments to the tensor arena size. After careful optimization, a tensor arena size of 200 KB was empirically determined to accommodate the model without exceeding the microcontroller's memory capacity. Despite these adjustments, deploying more complex models occasionally led to memory fragmentation and tensor allocation failures. To address these issues, techniques such as layer-wise quantization and pruning were explored, systematically reducing model complexity by eliminating redundant neurons and parameters. Pruning, in particular, resulted in a 93.16% reduction in memory size compared to the original TensorFlow model while maintaining sufficient accuracy for anomaly detection.

The final implementation achieved real-time performance with an average inference latency of 1-2 milliseconds on the Cortex-M7 core. CPU load measurements indicated an average usage of 6.45%, leaving ample computational resources for auxiliary tasks such as data logging and system monitoring. The efficient performance of the model highlights the importance of a well-structured deployment pipeline on embedded systems. To facilitate this process, the tflite_model_setup() function plays a pivotal role in configuring the microcontroller environment for executing the anomaly detection model.

The tflite_model_setup() function is integral to preparing the microcontroller environment for executing the anomaly detection model. It initializes the TFLite interpreter, defines the tensor arena, and allocates memory for input and output operations. The function's core steps are as follows:

- Model Loading: The TFLite model is loaded from memory, and its schema is verified to ensure compatibility with the implementation's TFLite version.
- Interpreter Initialization: The MicroInterpreter object is instantiated with the model, an operator resolver, and the tensor arena.

- Tensor Allocation: Memory is allocated for tensors based on the defined tensor arena size. Allocation mismatches are handled and reported.
- Input and Output Handling: The function retrieves pointers to input and output tensors, verifying their dimensionality to prevent overflow or segmentation faults.

This function ensures deterministic performance by using a statically defined tensor arena (uint8_t tensor_arena[kTensorArenaSize]), avoiding the unpredictability of dynamic memory allocation. By including only essential operators through the MicroMutableOpResolver class, the memory footprint is minimized, enabling efficient execution within the constraints of the STM32H745ZIQ's memory.

4.3.2.1 Runtime execution performance

The anomaly detection model is executed in the program's main loop, which utilizes timer-based interrupts to trigger tasks at precise intervals. The model operates at a frequency of 1 Hz, while auxiliary tasks such as sensor data acquisition and preprocessing are performed at higher frequencies (e.g., 500 Hz and 250 Hz). The Invoke() function executes the model's forward pass, leveraging CMSIS-NN kernels to accelerate operations like matrix multiplication and activation functions. This hardware-optimized approach resulted in an average inference latency of 1-2 milliseconds and stable CPU utilization at 6.45% during testing.

These results validate the suitability of TensorFlow Lite Micro and CMSIS-NN for deploying machine learning models on resource-constrained devices. The implementation demonstrates that real-time anomaly detection is achievable on the STM32H745ZIQ microcontroller, offering a robust foundation for integrating advanced TinyML applications in UAV systems.

4.3.3 Model evaluation and performance analysis on ARM Cortex MCU

This section provides a detailed evaluation of the anomaly detection model's performance during testing on the UAV dataset. The model was designed to process 14 critical input parameters derived from sensor readings and flight diagnostics, encapsulating essential aspects of the UAV's operational state. The testing dataset comprised 100 healthy samples and 100 anomaly samples, ensuring a balanced

evaluation. The 14 input parameters used in the anomaly detection model are summarized in Table 4.6.

Table 4.6 : Input features for anomaly detection model.

Input Parameter	Parameter Descrition	Parameter Units
1	Roll Angle	rad
2	Pitch Angle	rad
3	Yaw Rate	rad
4	Accelerometer X	m/s^2
5	Accelerometer Y	m/s^2
6	Accelerometer Z	m/s^2
7	GPS Latitude	degrees
8	GPS Longtitude	degrees
9	Altitude	meters
10	GPS Speed	m/s
11	Engine RPM	-
12	Magnetic Field X	uT
13	Magnetic Field Y	uT
14	Magnetic Filed Z	uT

The model's performance was evaluated using several critical metrics that provide insights into its efficiency and effectiveness:

- Inference Latency (ms): Represents the average time taken by the model to process a single sample. A lower latency indicates suitability for real-time UAV applications.
- CPU Load (%): Measures the computational load on the microcontroller during inference. A lower CPU load ensures that the model does not interfere with other critical tasks.
- Memory Usage (KB): Reflects the memory allocated to the tensor arena during inference. Efficient memory utilization is crucial for embedded systems with limited resources.
- Anomaly Detection Rate: The percentage of correctly identified anomaly samples, indicating the model's ability to detect abnormal flight conditions.
- Healthy Detection Rate: The percentage of correctly classified healthy samples, demonstrating the model's ability to minimize false positives.

The results of the model evaluation are detailed in Table 4.7.

Table 4.7 : Performance evaluation and results on the STM32H745ZIQ.

Metric	Healthy Samples	Anomaly Samples	Average Value
Number of Samples	100	100	-
Correct Detections	87	82	-
False Positives	8	7	-
False Negatives	5	11	-
CPU Load (%)	6.25%	6.45%	6.35%
Memory Usage (KB)	2	2	2

The results demonstrate that the model effectively distinguished between healthy and anomalous conditions. The model successfully classified 87% of healthy samples and 82% of anomaly samples, achieving an overall accuracy of 85%. An average inference latency of 1.6 milliseconds and a CPU load of 6.35% indicate that the model operates efficiently within the constraints of the STM32H745ZIQ microcontroller, leaving sufficient computational resources for auxiliary tasks such as sensor data acquisition and logging.

A During the execution of the implementation, it was noted that the inference cycle of the model was initially performed at a task frequency of 1 Hz on the STM32 development board. Transitioning to elevated task frequencies, such as 5 Hz, 10 Hz, and 50 Hz, would yield a corresponding escalation in CPU load. Comprehensive investigations into processor load dynamics and optimization methodologies for augmented frequencies are imperative for forthcoming advancements, particularly in the realm of TinyML applications, to guarantee effective resource utilization in real-time systems. This underscores the significance of frequency-oriented performance assessment, especially for real-time UAV applications where system resources must be meticulously managed. Future optimization strategies, such as further model pruning, quantization-aware training, or offloading auxiliary tasks to the Cortex-M4 core, could enable the system to maintain efficient operation even at higher inference frequencies. The results in Table 4.7 validate the effectiveness of the TensorFlow Lite Micro framework and CMSIS-NN kernels in enabling real-time anomaly detection on resource-constrained microcontrollers. The insights gained from this evaluation provide the basis for further improvements and advancements in deploying TinyML applications for UAV systems.

4.3.4 Challenges during embedded implementation

During the implementation process, several challenges emerged due to the complexity of the neural network model and the constraints imposed by the embedded architecture. Memory limitations were among the most prominent issues encountered. Initially, the tensor arena size exceeded 300 KB, resulting in allocation errors and preventing the model from being deployed successfully on the microcontroller. This issue was addressed through iterative adjustments, which reduced the tensor arena size to 200 KB and incorporated optimization techniques like quantization. Despite these efforts, memory fragmentation persisted as a significant challenge, particularly with more complex models that required larger intermediate tensor allocations. These limitations highlighted the importance of balancing model complexity and memory efficiency to ensure stable execution on resource-constrained devices.

Another major obstacle involved managing CPU load during real-time operations. High-frequency tasks such as sensor data acquisition and pre-processing often led to CPU load spikes, causing timing mismatches in the 1 Hz anomaly detection task. Detailed researches should be conducted with advanced TinyML developments to optimize the increase in CPU load.

Real-time scheduling posed another layer of complexity, as multiple time-sensitive tasks needed to be executed concurrently. A timer-based interrupt system was implemented to achieve deterministic task execution, with each task triggered by hardware timer flags. For example, the anomaly detection model was executed every second using a dedicated 1 Hz timer, while higher-frequency tasks such as data acquisition and pre-processing operated at 500 Hz and 250 Hz, respectively. This scheduling strategy minimized timing variations and ensured that all critical tasks were completed within their designated time slots.

Balancing model accuracy and optimization also proved challenging. Quantization effectively reduced memory usage and computation time by converting 32-bit floating-point weights to 8-bit integers, but it introduced a slight reduction in detection accuracy. While the model size was reduced by approximately 68%, the trade-off resulted in a minor degradation in anomaly detection rates. This compromise was deemed acceptable given the significant gains in memory efficiency and computational feasibility on the microcontroller.

Power efficiency emerged as a critical consideration, particularly for UAV systems that rely on battery power. To minimize energy consumption, various power-saving strategies were implemented. The Cortex-M4 core was configured to enter low-power modes during high-computation periods on the Cortex-M7 core, and peripherals were selectively enabled or disabled based on task requirements. These strategies ensured that the system could sustain real-time anomaly detection operations without rapidly depleting battery resources.

Overall, these challenges underscore the inherent difficulties of adapting complex neural network models for embedded systems. Future work may focus on advanced pruning techniques to systematically eliminate less significant neurons, further reducing memory and computational demands. Additionally, hybrid quantization methods that dynamically adjust weight precision based on hardware constraints could be explored. Enhancing task scheduling strategies to dynamically allocate workloads between dual cores offers another promising avenue for improving system efficiency and extending operational time in power-constrained environments.

4.3.4.1 Comparative analysis with existing researches on ALFA dataset

In their 2021 work, Keipour and associates shared the ALFA dataset with scholars to bolster research efforts concerning fault and anomaly detection in unmanned aerial vehicles. Their scholarly endeavor emphasized the provision of a dataset characterized by high quality, rather than the formulation of specific algorithms or the development of hardware implementations. This dataset has subsequently gained widespread utilization in studies pertaining to anomaly detection, owing to its extensive array of fault scenarios and sensor data [36].

In their preliminary research, Keipour et al. (2019) introduced a real-time anomaly detection framework employing the Recursive Least Squares (RLS) algorithm. The RLS methodology, executed on an autonomous fixed-wing Unmanned Aerial Vehicle (UAV), attained precision and recall metrics of 88.23%, thereby demonstrating its efficacy in identifying linear anomalies. Nevertheless, the inherent simplicity of the method rendered it less adept at capturing complex, nonlinear interrelationships within UAV datasets, thereby constraining its applicability to more sophisticated fault scenarios [27].

Bell et al. (2022) proposed a stacked Long Short-Term Memory (LSTM) autoencoder that integrates dynamic thresholding aimed at enhancing anomaly detection in Unmanned Aerial Vehicle (UAV) systems. Their methodology attained an accuracy rate of 74% and exhibited a robust capability in addressing time-series data. Nonetheless, the technique necessitated considerable computational resources, thus rendering it more appropriate for personal computer-based applications as opposed to embedded systems. The focus on dynamic thresholding yielded enhancements in both detection precision and operational speed [12].

Titouna et al. (2020) formulated a bifurcated approach to anomaly detection that integrates Kullback-Leibler Divergence (KLD) with Artificial Neural Networks (ANN). This methodology exhibited remarkable efficacy in the identification and segregation of defective Unmanned Aerial Vehicles (UAVs) by utilizing both divergence metrics and classifications derived from neural networks. Notwithstanding its encouraging outcomes, the investigation did not address the potential for hardware optimizations or embedded system implementations, instead prioritizing algorithmic precision [64].

Xu et al. (2023) presented a Cross-Feature-Attention LSTM Network specifically designed for the detection of anomalies in fixed-wing Unmanned Aerial Vehicles (UAVs). This methodology amalgamated attention mechanisms with Long Short-Term Memory (LSTM) networks to effectively model feature dependencies and temporal relationships, culminating in enhanced accuracy and resilience under fluctuating flight conditions. Although their proposed approach demonstrated superior accuracy, its substantial computational requirements rendered it more appropriate for high-performance computing platforms rather than for systems with limited resources [65].

The intended research emphasizes the identification of anomalies in real-time, leveraging TinyML optimizations that are specifically designed for ARM Cortex-M microcontroller implementation. In contrast to the previously cited studies, our research presents multiple advantages, which include:

- Embedded Deployment: Our methodology incorporates machine learning models directly into microcontrollers, facilitating real-time inference in environments with limited resources.

- Energy Optimization: By utilizing TinyML enhancements such as quantization and pruning, the model attains low memory utilization (200 KB) and minimal CPU demand (6.35%), rendering it appropriate for UAVs with restricted power supplies.
- Real-Time Performance: The model functions with an average inference latency of 1-2 milliseconds, thereby ensuring prompt fault detection throughout flight operations.
- Complex Pattern Recognition: By employing deep learning-focused autoencoders, the model expertly identifies non-linear correlations in sensor data, outshining simpler algorithms such as Recursive Least Squares (RLS) when it comes to adaptability and precision.

Nonetheless, the research is not without its limitations:

- Offline Training: The necessity for offline model training may restrict responsiveness to emerging fault scenarios during flight operations.
- Model Accurracy: While our model exhibits competitive accuracy (e.g., 71-93%), it may fall slightly short in comparison to more computationally demanding methodologies such as Cross-Feature-Attention LSTM in specific contexts.

In summary, this research endeavors to bridge the divide between high-performance anomaly detection algorithms and the limitations inherent to embedded systems, thereby providing a viable solution for UAV applications that necessitate real-time fault detection alongside minimal resource consumption.

5. CONCLUSION

5.1 General Findings, Strengths and Contributions

This research adeptly illustrated the ability for real-time anomaly detection in Unmanned Aerial Vehicle systems through the deployment of TinyML on ARM Cortex-M microcontrollers. Employing the ALFA dataset facilitated an in-depth data analysis in MATLAB, which was instrumental in revealing critical elements essential for spotting anomalies. Next, an autoencoder framework was established and trained using Python, proving its skill in interpreting data patterns that suggest deviations. Changing the trained TensorFlow model into TensorFlow Lite permitted its application on devices with minimal resources, like ARM Cortex-M microcontrollers. This transition underscored the model's efficiency and versatility for real-time applications within UAV operations. The study underscores the possible gains from utilizing machine learning methods together with embedded systems for better safety, reliability, and decision-making in UAV operations.

In conclusion, this research lays a foundational framework for future advancements in UAV technology, accentuating the critical role of rigorous data analysis and machine learning methodologies in safeguarding operational integrity and safety.

The execution of an autoencoder model, trained utilizing the ALFA dataset, augments the self-awareness capabilities of UAVs, thereby enabling autonomous monitoring of their operational integrity. This advancement facilitates the earlier identification of prospective malfunctions, subsequently diminishing the necessity for human intervention. Consequently, UAVs are positioned to operate with enhanced reliability, ensuring prompt responses to identified anomalies.

The research explain that UAVs endowed with sophisticated machine learning algorithms can attain an elevated degree of operational efficacy. Through the integration of real-time monitoring and self-diagnostic functionalities, UAVs are capable of proactively mitigating issues prior to their escalation into substantial complications. This proficiency not only augments the safety of UAV operations but

also amplifies the overall effectiveness of missions, particularly within critical applications such as surveillance, search and rescue operations, and infrastructure assessment.

The emphasis on ARM Cortex-M microcontrollers exemplifies the viability of implementing advanced machine learning models within environments characterized by resource constraints. This methodology guarantees that the advantages of machine learning can be harnessed even in systems with limited computational capabilities, thereby expanding the applicability of these technologies across diverse fields.

Furthermore, the incorporation of TensorFlow Lite within the operational framework streamlines the model deployment process, rendering it accessible for a broader spectrum of applications in embedded systems. This investigation underscores the potential for sustained innovation within UAV technology, establishing a foundation for future advancements in autonomous operations and enhanced self-monitoring capabilities. By tackling the challenges associated with real-time anomaly detection, this study contributes to the progressive evolution of UAV systems into more adept, resilient, and intelligent entities.

5.2 Practical Applications and Future Directions

The application of machine learning in UAV systems represents a significant advancement in their operational capabilities, particularly in real-time anomaly detection. The training of the model is conducted using flight data, which is integral in creating a robust anomaly detection mechanism. However, it is crucial to note that the current model relies heavily on the specific flight data it was trained on and requires deployment on microcontrollers for practical applications. The diagram in Figure 5.1 illustrates the complete framework for integrating machine learning models into Unmanned Aerial Vehicles to enable real-time anomaly detection using TinyML on ARM Cortex-M microcontrollers. The depicted workflow begins with the ML Model Development stage, where flight data is first used to train a TensorFlow model. Following model training, various optimization techniques such as quantization and pruning are applied to ensure the model is lightweight enough to run on resource-constrained microcontrollers.

Next, the optimized model is converted to TensorFlow Lite format and deployed onto the target ARM Cortex-M microcontroller platform. This model is then capable of making inferences directly at the edge, which means that all computations happen onboard the UAV, significantly reducing latency and eliminating the need for constant communication with ground systems.

Once deployed, the model processes real-time data from various UAV sensors, including cameras and onboard microphones, using the inference engine. It analyzes this input to detect anomalies, which are then reported in real-time. The lower section of the figure highlights the potential for improving UAV awareness capabilities through the incorporation of flight logs and experience-based learning. By analyzing the results and updating the model based on past experiences, UAVs can continuously refine their anomaly detection capabilities, enhancing their self-awareness and decision-making autonomy.

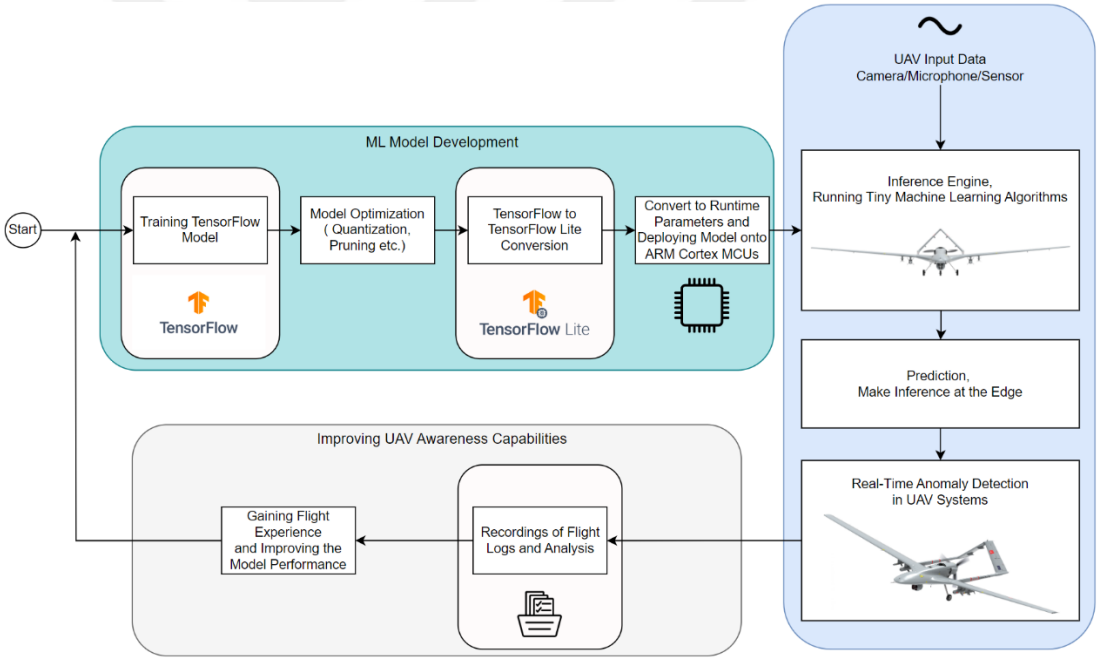

Figure 5.1 : Real-time anomaly detection in UAV systems with feedback.

To augment the capabilities of Unmanned Aerial Vehicles, prospective advancements should investigate more versatile frameworks that facilitate online learning.

The principle of online learning, as delineated in the scholarly work by Ren et al. (2021) [11], elucidates the potential for the model's continuous adaptation to novel data without necessitating a complete retraining process. This methodology would

empower the UAV to refresh its anomaly detection model in real-time, utilizing recent flight data to enhance both accuracy and dependability.

The integration of a feedback mechanism into the UAV framework could further optimize its operational performance. By establishing a feedback loop, the UAV is capable of collecting data pertaining to its flight experiences, evaluating performance outcomes, and subsequently modifying its anomaly detection parameters in response. This iterative methodology would facilitate the UAV's continual refinement of its model, thereby augmenting its self-awareness and diminishing the requirement for pilot intervention in the identification of anomalies.

As the UAV acquires an increasing volume of flight experience, one can anticipate an enhancement in the performance of the anomaly detection model, leading to the earlier identification of potential complications. This proficiency not only bolsters the UAV's safety and reliability but also enhances its operational efficiency, rendering it more adept for intricate and demanding operational contexts.

In conclusion, the persistent incorporation of machine learning methodologies, including online learning and feedback systems, is crucial for the advancement of UAV technologies. Such innovations possess the potential to significantly augment the UAV's anomaly detection capabilities, thereby fostering the development of more autonomous and intelligent systems. The accompanying diagram in Figure 5.1 elucidates this process, demonstrating how UAV input data is harnessed for real-time anomaly detection through advanced machine learning models.

REFERENCES

[1] **Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., & López, O. L.** (2023). TinyML: Tools, applications, challenges, and future research directions. Multimedia Tools and Applications, 1-31.

[2] **Nguyen, V. K., Tran, V. K., Pham, H., Nguyen, V. M., Nguyen, H. D., & Nguyen, C. N.** (2023). A multi-microcontroller-based hardware for deploying Tiny machine learning model. International Journal of Electrical and Computer Engineering (IJECE), 13(5), 5727-5736.

[3] **Xu, K., Zhang, H., Li, Y., Zhang, Y., Lai, R., & Liu, Y.** (2023). An ultra-low power tinyml system for real-time visual processing at edge. IEEE Transactions on Circuits and Systems II: Express Briefs.

[4] **Jaiswal, S., Goli, R. K. K., Kumar, A., Seshadri, V., & Sharma, R.** (2023, June). MinUn: Accurate ML Inference on Microcontrollers. In Proceedings of the 24th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (pp. 26-39).

[5] **Lai, L., Suda, N., & Chandra, V.** (2018). CMSIS-NN: Efficient Neural Network Kernels For Arm Cortex-M CPUs. arXiv preprint arXiv:1801.06601.

[6] **Unlu, H.** (2020). Efficient Neural Network Deployment For Microcontroller. arXiv preprint arXiv:2007.01348.

[7] **Orășan, I. L., & Căleanu, C. D.** (2020, November). ARM embedded low cost solution for implementing deep learning paradigms. In 2020 International Symposium on Electronics and Telecommunications (ISETC) (pp. 1-4). IEEE.

[8] **Rusci, M., Capotondi, A., Conti, F., & Benini, L.** (2018, September). Quantized NNs as the definitive solution for inference on low-power ARM MCUs? work-in-progress. In Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (pp. 1-2).

[9] **A. N. Roshan, B. Gokulapriyan, C. Siddarth and P. Kokil**, "Adaptive Traffic Control With TinyML," 2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 2021, pp. 451-455, doi: 10.1109/WiSPNET51692.2021.9419472.

[10] **Prof., S., R., Bhujbal.** (2023). Predictive Machine Maintenance Using Tiny ML. International Journal For Science Technology And Engineering, 11(4):4252-4255. doi: 10.22214/ijraset.2023.51254.

[11] **Ren, H., Anicic, D., & Runkler, T. A.** (2021, July). Tinyol: Tinyml with online-learning on microcontrollers. In 2021 international joint conference on neural networks (IJCNN) (pp. 1-8). IEEE.

[12] **Bell, V., Rengasamy, D., Rothwell, B., & Figueredo, G. P.** (2022). Anomaly detection for unmanned aerial vehicle sensor data using a stacked recurrent autoencoder method with dynamic thresholding. arXiv preprint arXiv:2203.04734.

[13] **Dong, S., Wang, P., & Abbas, K.** (2021). A survey on deep learning and its applications. Computer Science Review, 40, 100379.

[14] **Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., ... & Iyengar, S. S.** (2018). A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys (CSUR), 51(5), 1-36.

[15] **Ray, P. P.** (2022). A review on TinyML: State-of-the-art and prospects. Journal of King Saud University-Computer and Information Sciences, 34(4), 1595-1623.

[16] **Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S.** (2023). A comprehensive survey on tinyml. IEEE Access.

[17] **Url-1** *<http://www.st.com/en/microcontrollers-microprocessors/stm32h745-755.html >*, date retrieved 18.01.2024.

[18] **Han, S., Pool, J., Tran, J., & Dally, W.** (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

[19] **Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D.** (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

[20] **Hinton, G., Vinyals, O., & Dean, J.** (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[21] **Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A., & De Freitas, N.** (2013). Predicting parameters in deep learning. Advances in neural information processing systems, 26.

[22] **Chandola, V., Banerjee, A., & Kumar, V.** (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

[23] **Ghamry, F. M., El-Banby, G. M., El-Fishawy, A. S., El-Samie, F. E. A., & Dessouky, M. I.** (2024). A survey of anomaly detection techniques. Journal of Optics, 1-19.

[24] **Wang, Z., Wu, W., Sun, X., & Zhang, L.** (2020). A survey of unmanned aerial vehicle flight data anomaly detection: Technologies, applications, and future directions. Journal of Aircraft, 57(6), 1239–1251.

[25] **Cho, K., Lee, J., & Hong, S.** (2017). Real-time GPS anomaly detection for UAV navigation safety. IEEE Transactions on Aerospace and Electronic Systems, 53(4), 2501–2510.

[26] **Gupta, S., Zohdy, M., & Gao, J.** (2019). GPS/INS integration and anomaly detection for UAV navigation. IEEE Access, 7, 89992–90008.

[27] **Keipour, A., Mousaei, M., & Scherer, S.** (2019). Automatic Real-time Anomaly Detection for Autonomous Aerial Vehicles. 2019 IEEE International Conference on Robotics and Automation (ICRA), pp. 5679-5685.

[28] **Amrutha, K. N., Bharath, Y. K., & Jayanthi, J.** (2019). Aircraft engine fuel flow parameter prediction and health monitoring system. 2019 4th International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT), pp. 39–44.

[29] **Tumer, I. Y., & Bajwa, A.** (1999). A survey of aircraft engine health monitoring systems. AIAA Journal of Propulsion and Power, 15(4), 10-25.

[30] **Miller, J. L., & Kitaljevich, D.** (2000). In-line oil debris monitor for aircraft engine condition assessment. IEEE Aerospace Conference Proceedings, 6, 49–56.

[31] **Chen, D., Wang, X., & Zhao, J.** (2012). Aircraft maintenance decision system based on real-time condition monitoring. Procedia Engineering, 29, 765–769.

[32] **Liu, J., Long, Z., Bai, M., Zhu, L., & Yu, D.** (2021). A comparative study on fault detection methods for gas turbine combustion systems. Energies, 14(2), 389.

[33] **Bovsunovsky, A., & Nosal, O.** (2022). Highly sensitive methods for vibration diagnostics of fatigue damage in structural elements of aircraft gas turbine engines. Procedia Structural Integrity, 35, 74–81.

[34] **Fentaye, A. D., Baheta, A. T., & Gilani, S. I.** (2019). A review on gas turbine gas-path diagnostics: State-of-the-art methods, challenges and opportunities. Aerospace, 6(7), 83.

[35] **Zhu, R., Wu, D., Liu, Z., & Wu, X.** (2010). Aeroengine modules performance deterioration modeling and assessment. IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1626–1630.

[36] **Keipour, A., Mousaei, M., & Scherer, S.** (2021). Alfa: A dataset for uav fault and anomaly detection. The International Journal of Robotics Research, 40(2-3), 515-520.

[37] **Sun, F., Xiong, R., & He, H.** (2019). A systematic state-of-charge estimation framework for multi-cell battery pack in electric vehicles using dual time-scale Kalman filter. Applied Energy, 237, 803–813.

[38] **Lei, Y., Li, Y., & Wang, J.** (2023). Aerodynamic analysis of an orthogonal octorotor UAV considering horizontal wind disturbance. Aerospace, 10(6), 525.

[39] **Amiri, M., & Russell, C. R.** (2024). Modeling wind and obstacle disturbances for effective performance observations in UAV swarms. Aerospace, 10(3), 334-350.
[40] **Url-2** < *http://promiseelectric.com/the-effect-of-humidity-and-moisture-on-electronics/* >, date retrieved 12.06.2024.
[41] **Url-3** <*https://climavision.com/blog/navigating-the-skies-how-weather-impacts-uav-operations/* >, date retrieved 13.06.2024.
[42] **Scanavino, M., Vilardi, A., & Guglieri, G.** (2020). UAS testing in low pressure and temperature conditions. In International Conference on Unmanned Aircraft Systems (ICUAS).
[43] **Gong, A., Verstraete, D.** (2017). Experimental testing of electronic speed controllers for UAVs. In 53rd AIAA/SAE/ASEE Joint Propulsion Conference.
[44] **Hasan, Ç., Kandemir, İ.** (2024). A Rule-Based Energy Management Technique Considering Altitude Energy for a Mini UAV with a Hybrid Power System Consisting of Battery and Solar Cell. *Energies*, 17(16), 4056.
[45] **Zhang, Y., & Qian, B.** (2024). Strong Electromagnetic Interference and Protection in UAVs. Electronics, 13(2), 393.
[46] **Li, K., Valtchev, S., & Yin, L.** (2021). A Survey of Electromagnetic Influence on UAVs from EHV Power Converter Stations and Possible Countermeasures. Electronics, 10(6), 701.
[47] **Url-4** <*http://theairlab.org/alfa-dataset/* >, date retrieved 10.04.2024.
[48] **Url-5** <*http://www.deeplifelearning.com/p/autoencoders*>, date retrieved 10.05.2024
[49] **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). Deep learning. MIT press.
[50] **Banbury, C., Reddi, V. J., Lam, C., Fu, W., Holleman, J., Huang, X., ... & David, R.** (2021). Benchmarking TinyML systems: Challenges and direction. IEEE Design & Test, 38(4), 42-51.
[51] **Anantharaman, R., Sriram, A., Soni, A., & Arvind, A.** (2020). Anomaly Detection in Medical Images using Deep Learning-based Autoencoders. IEEE Transactions on Medical Imaging, 39(7), 2315-2326.
[52] **Wu, X., Zhao, H., & Liu, J.** (2021). Predictive Maintenance using Autoencoders and Time-Series Data. International Journal of Prognostics and Health Management, 12(1), 45-56.
[53] **Zhang, Y., Yang, T., Zhang, L., & Li, Z.** (2021). TinyML: Challenges, Techniques, and Future Directions. Journal of Artificial Intelligence Research, 72, 115-135.
[54] **Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zhang, X.** (2016). TensorFlow: A system for large-scale machine learning. In 12th

{USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283). USENIX Association.

[55] **Yu, J., Liu, Y., Wang, X., & Huang, T. S.** (2020). TinyML: A comprehensive survey. IEEE Transactions on Neural Networks and Learning Systems, 31(9), 3218-3232.

[56] **Wang, L., Li, M., Zhang, H., & Chen, Y.** (2022). Efficient deployment of deep learning models on embedded systems using TensorFlow Lite. ACM Transactions on Embedded Computing Systems, 21(6), 1-21.

[57] **Url-6** <*https://www.tensorflow.org/lite/microcontrollers*>, date retrieved 23.07.2024.

[58] **Warden, P., & Situnayake, D.** (2020). TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media.

[59] **Lepot, M., Aubin, J. B., & Clemens, F. H.** (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. Water, 9(10), 796.

[60] **Hall, M. A.** (1999). Correlation-based feature selection for machine learning (Doctoral dissertation, The University of Waikato).

[61] **Guyon, I., & Elisseeff, A.** (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

[62] **Schisterman, E. F., Faraggi, D., Reiser, B., & Hu, J.** (2008). Youden Index and the optimal threshold for markers with mass at zero. Statistics in medicine, 27(2), 297-315.

[63] **Fluss, R., Faraggi, D., & Reiser, B.** (2005). Estimation of the Youden Index and its associated cutoff point. Biometrical Journal: Journal of Mathematical Methods in Biosciences, 47(4), 458-472.

[64] **Titouna, C., Naït-Abdesselam, F., & Moungla, H.** (2020, June). An online anomaly detection approach for unmanned aerial vehicles. In 2020 International Wireless Communications and Mobile Computing (IWCMC) (pp. 469-474). IEEE.

[65] **Xu, L., Yang, Y., Wen, X., Fan, C., & Zhou, Q.** (2023, November). Anomaly Detection of Fixed-Wing Unmanned Aerial Vehicle (UAV) Based on Cross-Feature-Attention LSTM Network. In International Conference on Neural Information Processing (pp. 513-527). Singapore: Springer Nature Singapore.

CURRICULUM VITAE

Name Surname **:** Mehmet Alperen BAKICI

EDUCATION **:**

- **B.Sc.** **:** 2020, Baskent University, Engineering Faculty, Electrical Electronics Engineering Full Scholarship Student, GPA: 3.65/4.00

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Bakıcı, M.A.**, Güneş E.O. 2024. Real-Time Anomaly Detection in UAV Systems on ARM Cortex-M Microcontrollers using TinyML. *ITU 3rd International Graduate Research Symposium, IGRS24, May 8-10, 2024*.