Real-Time Aviation Anomaly Detection and Multi-Label Classification Using Deep Learning on Multivariate Sensor Data

by

Sakib Rayhan Yeasin
24341218
Md. Atik Hasan Rahat
22101162
Shafaat Jamil Nakib
24241257
Debjoty Mitra
22141001

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
April 2026.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sakib Rayhan Yeasin
24341218

Md. Atik Hasan Rahat
22101162

Shafaat Jamil Nakib
24241257

Debjoty Mitra
22141001

Approval

The thesis/project titled "Real-Time Aviation Anomaly Detection and Multi-Label Classification Using Deep Learning on Multivariate Sensor Data" submitted by

1. Sakib Rayhan Yeasin (24341218)
2. Md. Atik Hasan Rahat (22101162)
3. Shafaat Jamil Nakib (24241257)
4. Debjoty Mitra (22141001)

of Spring, 2026 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in 12 April, 2026.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Md. Tanzim Reza

Senior Lecturer
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor & Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

General aviation records a fatal accident rate of approximately one per 100,000 flight hours, with loss-of-control and stall events remaining leading preventable causes. Existing flight safety systems rely on fixed expert-defined thresholds and cannot detect complex multi-sensor anomaly patterns or identify specific event types in real time. This thesis proposes a lightweight two-stage deep learning framework for real-time aviation anomaly detection and multi-label event classification on raw flight sensor data from the NGAFID General Aviation Training Set. Stage 1 uses an ensemble of two novel Transformer architectures, the Cross-Sensor Patch Transformer (CSPT) and the Hierarchical Cross-Sensor Transformer (HiCST), each incorporating a cross-sensor multi-head attention module that explicitly models inter-sensor dependencies before temporal processing. Root Mean Square ensemble fusion achieves an anomaly-class F1-score of 0.8815, recall of 0.9076, and AUPRC of 0.9523 on a test set with a 339:1 class imbalance ratio. Stage 2 uses MHANet, which introduces per-sensor independent linear projections to classify each anomalous timestep into any combination of ten simultaneous event types, achieving a macro F1 of 0.9563 and subset accuracy of 0.9627. The complete pipeline runs in 6.08 milliseconds per sensor reading on a standard CPU, confirming real-time feasibility. Both stages outperform all established deep learning and classical machine learning baselines while using significantly fewer parameters, demonstrating that domain-aware architectural specialization consistently outperforms general-purpose approaches for aviation safety monitoring.

Keywords: Aviation Anomaly Detection, Multi Level Classification, Transformer, Hierarchical Framework, Time-Series Data

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The aviation industry has seriously changed over the last 20 years in terms of technological advancement and modern aircrafts are fitted with advanced sensor systems which continuously check the flight parameters, engine performance and aircraft behaviours. These systems produce huge amounts of data at every flight operation and gather vital data on aircraft behavior, environmental factors, and any anomalies that might affect flight safety. A representative of the most informative sources of flight telemetry data, the National General Aviation Flight Information Database (NGAFID), which summarizes information about a variety of general aviation flights, includes different types of aircraft, their operational conditions and different stages of the flight.

Figure 1.1: Cessna 172S Aircraft

The general aviation that constitutes a significant part of the global civil aviation operations is characterised by special safety concerns as opposed to commercial aviation. Smaller aircraft, which can be owned by an individual or small company, can offer pilots with tight operational situations with which they have to be extremely aware and make decisions. High-altitude stalls, low-altitude stalls, the loss of airspeed during critical stages of flight, and poor pitch or roll control are critical events that have continued to pose a risk in the general aviation operation. The challenge of identifying these anomalies becomes even more significant with an account of the fact that every event has unique manifestations in the various phases and states of flight, and that anomalies in most cases manifest themselves in a multi-complex way, making it a combined threat that needs to be identified concurrently.

Conventional methods of detecting anomalies in aviation have been based on their post-flight analysis and by their safety personnel and exploiting the domain knowledge and the past accident data bases. But this reactive solution will in itself not stop accidents but will help to learn with experience about previous accidents.

The advent of machine learning and deep learning algorithms has created fresh opportunities in real time anomaly detection and now proactive identification of dangerous flight conditions can be made in advance before turning into a killer event. The shift towards the use of aviation safety as a predictive paradigm, based on data, as opposed to a reactive, incident-focused approach can be seen as a paradigm shift in the viewpoint of the aviation community about flight risk reduction.

The use of time series analysis in the aviation industry poses some unique challenges as flight operations are dynamic. The aircraft behavior is influenced by numerous interacting variables such as altitude, airspeed, pitch, roll, acceleration, on the other hand, fuel status and the vertical speed amongst others that have complex and non-linear effects on each other. More often than not, traditional statistical approaches are inadequate in that they fail to explain these complex interdependencies. Integrated with learning of time patterns and sensor interactions in aviation data, transformer-based architectures that have overtaken the state of the art over the last several years in natural language processing and sequential data analysis have potential opportunities. The capability of the attention mechanisms to weight various sensors and time steps would allow models to learn which combinations of sensors readings would actually indicate any of the dangerous flight conditions.

Moreover, the multi label property of aviation anomalies in which two or more types of events may happen at the same time during a flight or even within a small time frame necessitates specific machine learning techniques. A pilot who is stalled in high air has a compound risk which is neither of the two conditions. The same event during approach phases has different implications as compared to the same sensor values during cruise flight. The combination of binary anomaly detection (uncovering the abnormal and normal functioning of the system) with multi-label classification of events (discovering a particular type of event) is a holistic approach to dealing with the range of aviation safety issues.

1.2 Rationale of the Study and Motivation

This research is driven by the combination of three urgent conditions, namely: the ongoing problem of flight safety in general aviation, the emergence of deep learning technology excited over a short time, and the possibility of obtaining full flight telemetry data and having it accessible to research projects.

There is still a considerably high rate of accidents in general aviation compared to commercial aviation, despite decades of research and regulation development in the field of aviation safety. Based on aviation safety statistics, incidents of general aviation accidents often depict situations that could have been avoided by anomaly detection and alerting the pilots during real-time. Accidents like accidental stalls, stalls at low altitude during the take off and landing and failure to maintain air speed at critical stages of flight are some of the recurrent accident causal factors. The drive to build automated detection systems has been influenced by the appreciation that the perception of human pilots has specific constraints even after several years of training in the face of the cognitive load of the flight

operations with the sensor monitoring task. An efficient automatic system could notify the pilots about emerging harmful conditions which may give them decisive seconds to take correction.

The second driving force is in regard to technological development. Transformer based architecture, albeit a revolution in the field of natural language processing, constitutes a comparatively unexplored domain when it comes to aviation applications. Attention across sensors including mechanisms that allow models to learn the combinations of sensor readings that are most significant can provide new opportunities to comprehend the dynamics of aircraft. Some transformer architectures, in contrast to conventional statistical methods, do not require explicit engineering of features based on domain knowledge, instead learning feature values that are potentially invisible to human domain experts. Patch-based time modeling combined with cross sensor attention offers a conceptually and empirically robust solution to the multivariate and multiproject problem of detecting aviation anomalies.

The third driving power focuses on the availability of data and the research opportunity. This has a research opportunity due to the presence of the NGAFID dataset that categorizes flight records based on clear labels of anomalies and the classification of events. High quality and size labeled datasets like these are highly uncommon in aviation applications, and are usually limited by privacy policies, proprietary issues, and the fundamental challenge that thorough flight data are difficult to access. This data will form the basis required to build and test advanced machine learning models without having to create synthetic data and simulation-driven approaches that might not be reflective of reality.

In addition to solving the immediate practical problems, this study gives insights to underlying questions regarding the design of deep architectures to be used in safety-critical applications. These include: how to manage severe class imbalance (where dangerous events constitute less than 0.2% of all flight data and carry high safety costs); how to ensure performance consistency across event types with varying budgets; how to achieve interpretability acceptable for aviation regulatory acceptance; and how to optimize the precision-recall trade-off when both false alarms (unnecessary alerts) and false negatives (missed detections) carry significant safety and operational consequences.

Besides, using binary anomaly-detection along with multi-label event classification as the multi-stage approach is operational. Pilots must not just be aware that something is wrong but what the issue is so that they can take necessary corrective measures. A pilot alerted with the abstract principle of an anomaly will not be able to react positively but a pilot alerted with the development of the high-altitude stall with explicit altitude, airspeed and pitch data will be able to consult known emergency procedures. This two-stage structure emulates the ways humans solve problems and makes it more useful.

1.3 Problem Statement

- **Core Challenge: Real-Time Anomaly Detection in General Aviation:**
  Regulatory oversight and safety programs notwithstanding, general aviation accidents are still a major cause of aviation fatalities. Out of the multiplicity of causal elements that are discovered during the investigation of accidents, in general aviation, about 15-25% of accidents result in either a loss of control incident, stall, or extreme attitude incident, that are all present as distinctive patterns in aircraft sensor data. However, the current operating procedures offer the pilots post hoc safety analysis, but in real time it did not show anomaly indicators. It is the time delay between when the anomaly takes place and the time of awareness which is the root of the problem, that pilots in general aviation planes are often not aided by real-time decision support systems that would give them a chance to notice hazardous flight situations developing.

- **Technical Complexity: Multivariate Sensor Integration:**
  Flight dynamics of aircrafts are a result of interaction between a variety of sensors predetermining the measurement of diverse physical events. Altitude above ground level, altitude lag differential, indicated airspeed, normalized acceleration, pitch, roll, stall index, total fuel and vertical speed, which are the nine main sensors considered in this research, are used to give information on various aspects of aircraft state. Dangerous flight conditions do not however manifest as extreme data on one or other sensor but, conversely, arise due to certain combinations and time series patterns of sensor data. An example of this is the development of a stall condition by a characteristic sequence of airspeed reduction, pitch increase, drop in altitude, and subsequent patterns of characteristic acceleration. However, differences in aircraft type, configuration, weight and environmental considerations result in the specific sensor values and transition rates to change significantly. Explicitly defined rule-based systems, capturing domain knowledge as explicit if-then statements, do not scale to this variation, prompting the creation of learning-based systems that can uncover the appropriate patterns directly out of data.

- **Severe Class Imbalance Problem:** Class imbalance in flight data is extreme in aviation safety scenarios: in an overall data set of flights, anomalous, or dangerous flight conditions occupy a small percentage of the total time in the air, which is typically 10 per cent or less. This asymmetry undermines the traditional solutions of machine learning. By predicting 100 percent of samples to be normal flight, a naive classifier with 90 percent of the samples being correct has 0 safety value and 90 percent accuracy. The default loss, in contrast to cross-entropy, inverts the weights of all wrong decisions, and results in a poorly performing anomaly class which is seldom seen but critically important. It becomes more severe in a multi-label setting where the frequencies of various types of events are orders of magnitude apart some events are found with frequencies below 1% of the flights and the result is a severe label imbalance that cannot necessarily be resolved by traditional means.

- **Multi-Label Classification Complexity:** Causes of aviation anomalies of-

ten occur together. A stall that is developing at high altitude needs to be handled differently than high-altitude stall, but both stalls have stall dynamics. Several types of events may compose the same flight leg during which a pilot losing airspeed during approach in an unusual pitch attitude may face simultaneous dangers which have to be identified concurrently. Standard binary classification models treat events separately and are not able to represent these interactions. Multi-label classification methods are required to classify any number of zero up to ten potential event types in each flight segment, where event types have vastly different base frequencies. Also, various types of events have diverse implications with regard to operations; a stall when heading towards the approach will require urgent attention, and the same speed rate in the cruise will be completely normal. This demands not only a detection but also precise event-type identification.

- **Temporal Pattern Recognition:** Aviation sensor data is inherently temporal, in contrast with many machine learning applications that use a single set or a single unit of features, e.g. a feature vector. Hazardous flying conditions are normally formed within particular time range stalls are formed within seconds, but the fuel rate trends within the span of hours. Detectors need to be implemented on time windows that are large enough to reflect relevant dynamics and small enough to issue timely alerts. The choice of the window size is inherently a trade-off because such short windows lose the entire pattern of development of an anomality, and, conversely, very long windows distort an anomaly signal with much normal-flight data. Moreover, each of the three temporal patterns exists at several scales at the same time, and needs models capable of modeling not only the fine-scale dynamics but the longer-term trends as well.

- **Sensor Heterogeneity and Calibration:**

  Sensors have a variety of measurement ranges, sensitivities, as well as physically defined meanings. The altitude scales in to the thousands of meters and the angle of pitch are usually between -90 and +90. Acceleration could be within a range of -2g to +2g. In the absence of a suitable normalization of the array of sensors, training standard neural networks would assume that they carry equal significance, whereas the difference in relative magnitudes is due to measuring scaling as opposed to measuring importance. Also, sensor calibration may be different depending on the aircraft type and operation and introduces distribution shift problems. The models should not just identify anomalies, but do so well in a wide variety of sensor configurations and calibration conditions.

- **Precision-Recall Trade-off in Safety-Critical Context:**

  The precision-recall trade-off should be optimized carefully in aviation applications. False alarms (signaling the dangers that are not actually there when the pilots are alerted) cause the alert fatigue, which in turn lowers the attentiveness of the pilots to the real alerts and damages the credibility of the system. On the other hand, false negatives (lost real anomalies) are actual safety dangers. The way machine learning is typically developed focuses on maximizing accuracy or F1-score without directly taking into account the asymmetric impact of

types of errors. The trade-off that the different operational contexts and event types might require different precision-recall operating points may need to be clearly managed in aviation safety. The system should allow threshold settings allowing varying precisionrecall decisions based on the type of anomaly and the device phase.

- **Model Interpretability for Regulatory Acceptance:**

  Air transport is considered as one of the most controlled industries in the world. Any automated anomaly detection system to be incorporated into aircraft operating environments would need to meet high-regulatory standards that requires evidence of reliability, robustness and failure modes. "Black-box" deep learning models can potentially perform better but there is regulatory scepticism. The regulatory authorities are interested in the manner and reason of decision-making by a system, how they address edge cases, and the failure modes. This introduces a basic conflict pitting model complexity (which tends to enhance performance) against interpretability (which must be certified by regulatory authorities). This system should be capable of competitive performance but still have adequate transparency to enable its acceptance by the regulators.

1.4 Research Gap

Although many recent studies have improved aviation anomaly detection using deep learning, there are still some important problems. Most of the existing works like Rong et al. [15] and Zhao et al. [33] mainly focus on binary classification (normal vs anomaly), which is not enough for real-world use because it does not explain what type of problem is happening. Also, many studies depend on external data such as ADS-B instead of using full internal sensor data, so they miss early-stage system failures. Another issue is that models like LSTM cannot properly capture long-duration problems since they work on short time steps instead of meaningful time segments. In addition, some advanced models require extra retraining steps after detecting anomalies, which makes them slow and not suitable for real-time systems as shown by He et al. [7]. Finally, aviation datasets are highly imbalanced, but most works do not properly handle this, leading to poor detection of rare events as discussed by Mejri et al. [26]. So, there is a clear need for a fast, real-time model that can use full sensor data and give multi-label outputs for better decision making.

1.5 Research Objectives

- **Develop a binary anomaly detection system for general aviation sensor data:**

  Tuned and trained a Transformer-based ensemble model that can be used to learn anomalies in real-time using 30-timestep multivariate sensor windows. The system has to reach an anomaly-class F1-score of greater than 0.88 on held-out test data whilst it has to manage the extreme class imbalance of general aviation flight records without synthetic data additions.

- **Develop a multi-label event classification system for anomalous timesteps:**

  Hypothesize a model which categorizes all detected anomalies into any combination of ten concurrent aviation event categories. The system needs to score above 0.95 macro F1, should be able to achieve similar per-class results between common and rare events, and should have interpretable per-event probability results that should be used to communicate with pilots.

- **Address class imbalance without data augmentation:**

  Discover and implement algorithms and methods of loss function design and sampling that can be used to effectively detect and classify rare anomaly events on highly imbalanced real flight data, without introducing artificial samples that do not necessarily represent realistic flight physics.

- **Ensure computational efficiency for real-time deployment:**

  Design architectures can be made lightweight enough to execute single-sample inference latency on standard CPU hardware below a second or less for per sensor reading, and can be deployed on board or to ground-stations without any specialized avionics hardware.

- **Evaluate performance against established baselines:**

  Compare the suggested architectures systematically to popular time-series deep learning models and classical machine learning solutions, on the basis of training protocols and metrics of measures of performance and efficiency to quantify the performance and efficiency benefit of domain-sensitive design decisions.

- **Investigate ensemble strategies for Stage 1 detection:**

  Test various ensemble fusion strategies between the two independent Stage 1 models, and determine whether a significant increase in the precision-recall trade-off can be achieved by combining complementary architecture with any single model.

1.6 Methodology in Brief

- **Dataset:**

  The experiment relies on the General Aviation Training Set (GATS) provided by the NGAFID, i.e. the Cessna 172S one. The dataset includes more than 22 million rows of timesteps of 4,481 flights which are measured at a frequency of 1 Hz. After feature selection, there remains nine sensor channels: altitude above ground level, lagged MSL altitude difference, reported airspeed, normal acceleration, pitch, roll, stall index, total fuel, and vertical speed. The data

is highly skewed, with 99.83 of the rows being classified as normal flight and only 0.17 as anomalous based on 10 types of in-flight event. Missing values are dealt with using a two-tier imputation approach: linear interpolation of the temporal gaps in the rows and the median column filler of the complete column gaps of sensor channels.

- **Data partitioning:**

  The data is divided at the flight level in order to avoid data leakage where all the timesteps of a flight are allocated to a single partition. There is an 80/10/10 train-validation-test split. In Stage1, event-stratified partitioning means that the proportion of each type of event Watcher anomaly is the same in all splits. In Stage 2, a greedy multi-label stratification algorithm is used to deal with the multi-occurrence of a large number of simultaneous events in a flight. In the case of Stage 1, 30-timestep sliding windows are created that have a stride of 1 and the label of the last timestep is the window label.

- **Stage 1: Binary anomaly detection:**

  Stage 1 identifies the existence of any attack in a particular 30-timestep sensor window. This task is done with the development of two independent Transformer based models. The Cross-Sensor Patch Transformer (CSPT) originally runs a cross-sensor multi-head attention module on the nine sensor channels at each timestep to provide inter-sensor dependencies, and processes the time sequence into 9 overlapping patch tokens (patch length 6, stride 3) and processes them through 3 Pre-LayerNorm Transformer blocks with a CLS token. The Hierarchical Cross-Sensor Transformer (HiCST) uses the cross-sensor attention module but uses each of the 30 timesteps as an independent token and reduces the sequence with distillation convolutions (Conv1d, BatchNorm, ELU, MaxPool) one after another, shrinking the sequence to 15 and then 8 tokens across three encoder layers. Both methods have an internal SensorNorm layer which maintains training-set statistics of normalization and the restored readings are fed directly into the model at inference time with no external precharacterizations. The two models are trained on Focal Loss with a focusing exponent of 2.8 and dynamically computed class weighting parameter, a Weighted Random Sampler to oversample anomalous windows, OneCycleLR scheduling where we use a 10 percent warmup and Exponential Moving Average weight smoothing, as well as Automatic Mixed Precision training. The best decision threshold is chosen per epoch by sweeping candidate values over the validation set and selecting the best candidate which maximizes binary F1. The last ensemble is a hybrid of both models via Root Mean Square fusion of their predicted anomaly probability where a little more weight is put on the more confident model at any given prediction.

- **Stage 2: Multi-label event classification:**

  When Stage 1 detects an anomalous timestep in Stage 2, it determines which 10 types of events are active at that point in time. Stage 2 input is not a

temporal window but a single row of sensor values (shape 1x9) of sensor values (not a temporal window) as the identity of an active aviation event is only determined by the sensor configuration at that moment and not its temporal history. MHANet is a Stage 2 model, which maps the scalar values of each sensor to 64-dimensional tokens using nine independent linear projections, one per sensor, so that each sensor learns its own physically meaningful representation. A positional encoding that can be learned is introduced to differentiate the sensor positions. It uses two blocks of Post-LayerNorm Transformer that apply multi-head self-attention to the sensor tokens to learn cross-senor relationships and then a three-layer classification head that resolves ten independent sigmoid outputs. MHANet is trained with Asymmetric Loss, which places higher focusing exponent on the easy negative samples when compared to the positive ones and using inverse-frequency class weights plus manually boosting the most rare event classes. The learning rate scheduler is the CosineAnnealingWarmRestarts, and individual per-class decision levels are trained on the validation set.

- **Evaluation:**

  Stage 1 is assessed on the held-out test set in terms of anomaly-class precision, recall, F1-score and AUPRC. The overall accuracy is not applicable since the 339: 1 imbalance in classes makes it uninformative. The stage 2 is measured by macro F1 (the main measure, all weight of the 10 event classes), micro F1, subset accuracy, and Hamming loss. The proposed models are contrasted with seven existing deep learning baselines in time series classification, as well as four classical machine learning methods, which are also trained using the same training strategy and tested on the same test set.

The complete methodology can be briefly illustrated in the following figure.

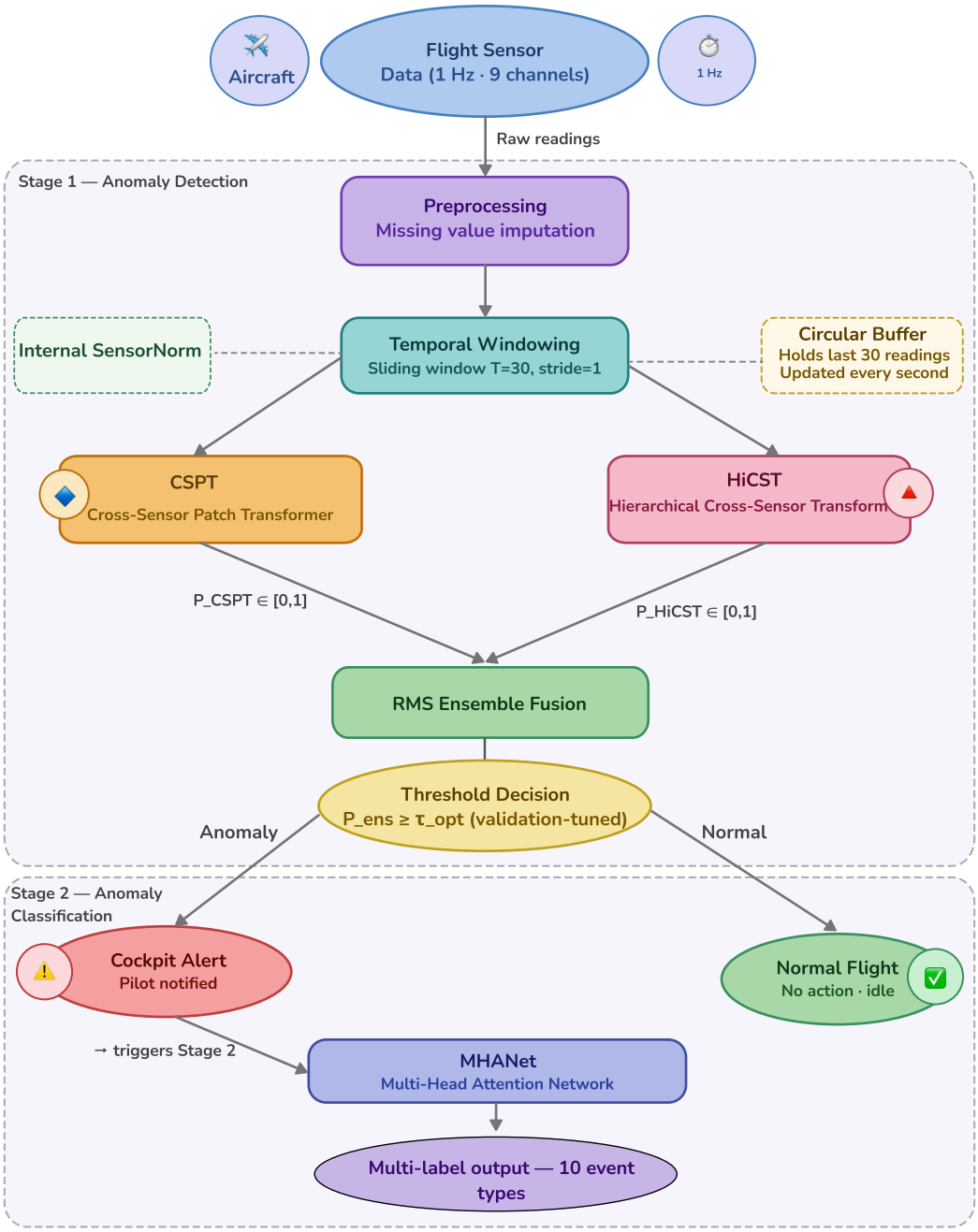

Figure 1.2: Methodology Overview

1.7 Scopes and Challenges

1.7.1 Scopes

The scope of this research includes 10 aviation anomaly events types of data tracked during the flights of general aviation involvement: High Altitude Stall, Low Airspeed on Climbout, Low Airspeed on Approach, VSI on Final, Roll, Low Pitch, Low altitude Stall, High altitude Spin, High pitch, and Low fuel. The dataset entails some 500,000 training windows, 100,000 validation windows and 200,000 test

windows assembled by individual flight records in the GAFID database. Multi-year operational data across the different aircraft types, the level of skills of pilots and the conditions of operations falls under the temporal scope.

The sensor is defined to have 9 physical measures namely: altitude above the ground level, altitude lag differential, reported airspeed, normalized acceleration, pitch, roll, stall index, total fuel, and vertical velocity. This is the key parameter that is taken by these sensors as the most important to detect anomaly and still within the normal range of capabilities of a general aviation aircraft. Other aspects such as weather data, maintenance history and pilot attributes are not included in the scope of the research since it only emphasizes real time sensor-based anomaly detection.

1.7.2 Challenges

The most primordial problem is excessive imbalance in classes which is inherent in the aviation safety applications. Dangerous flight conditions are a minor constituent of the overall time in flight in any large set of flights. In rare events, such as High Altitude Spin or Low Fuel states, anomalies can be $< 1\%$ total data, and simple types of anomaly prediction, like always predict normal strategies, can be performed with high accuracy by naive algorithms. This objective demands specialized loss functions (Focal Loss, Asymmetric Loss) and training schemes (weighted sampling, sensitive choice of threshold) to make sure that models acquire meaningful patterns on the rare occasions instead of getting caught in the trap of trivial solutions.

A related issue relates to identifying real irregularities and normal variation in flight operations. Normal flight operations Aircraft deliberately enter steep turns and climb-outs, landings with different pitch and roll angles involve steep turns, climb-outs and approaches all with superficial similarities to angle conditions when anomalies unintentionally occur. An algorithm has to be taught that same pitch/roll combinations are normal functioning situations in some situations but hazardous in other situations. This pattern-dependent definition of anomaly demands that the model should not be guided by rules based on threshold, but rather be learnt based on temporal patterns and combinations of multiple sensors.

Real-world, operational deployment of aircraft places strong requirements on model complexity and speed of inference. Avionics systems are generally run on limited computing resources, and have strict power budgets. The model should have a latency of inference of at most of one milliseconds to allow real time alerting, and will need to fit into a normal avionics memory space. This need to be efficient and yet accurate is an inherent problem, and may necessitate delicate tradeoffs in architecture design and quantization.

The multi-label classification phase introduces even greater difficulties: models must perform with high accuracy on typical events, but also on typical events without making mistakes on rare events; what is more, this is susceptible to 10 different labels at a time. It is a common practice of using standard methods that aim to attain good performance on average on all labels and have underperforming labels. The consistency among labels requires advanced loss functions and appropriate

training methods.

Lastly, some of the core issues relate to regulatory acceptance. There is a requirement of not only the systems to work, but also that engineers should be able to reveal how and why to work, failure modes, and how edge cases are handled by the system. Although black-box decision-making can be appealing under regulatory scrutiny, deep learning architectures pose regulatory challenges due to their potential capability to perform more effectively. The challenge requires striking a balance among the model complexity (which tends to enhance performance) and interpretability (which regulatory approval requires) and validation rigor (whether it is consistent across a broad range of operational conditions).

Chapter 2

Literature Review

2.1 Preliminaries

Aviation safety analysis focuses on understanding patterns within flight data to identify abnormal behavior that may lead to hazardous events. Modern aircraft generate large volumes of continuous multivariate sensor data, making automated analysis systems essential for real-time monitoring and decision support. During flight, parameters such as airspeed, altitude, roll, pitch, and vertical speed are recorded at high frequency, forming a multivariate time-series representation of aircraft behavior. Instead of analyzing independent data points, this study adopts a sliding window approach, where a sequence of consecutive timesteps is processed together to capture temporal dependencies and detect patterns that precede anomalies. This approach is particularly important as many safety-critical events develop over a period of time rather than occurring instantaneously.

In addition to temporal relationships, aviation anomalies are often caused by the interaction of multiple sensors rather than a single parameter exceeding a threshold. Therefore, understanding inter-sensor relationships is crucial. To address this, attention mechanisms are utilized, enabling the model to learn how different flight parameters influence each other under both normal and abnormal conditions. This is further enhanced through Cross-Sensor Attention, which explicitly models the co-activation and dependency between variables such as altitude, vertical speed, and pitch, reflecting the physical dynamics of flight.

To efficiently process long sequences of flight data, Transformer-based architectures are employed, as they are capable of capturing long-range dependencies while maintaining computational efficiency. Unlike traditional recurrent or convolutional models, Transformers use self-attention to dynamically focus on the most relevant parts of the sequence, making them well-suited for complex temporal pattern recognition. Additionally, Patch-Based Tokenization is used to divide long sequences into smaller temporal segments, allowing the model to focus on sustained conditions rather than momentary noise, which is critical for identifying aviation hazards.

Another important concept in aviation anomaly detection is multi-label classification. In real-world scenarios, multiple anomalies can occur simultaneously (e.g., low airspeed and high descent rate), making single-label classification insufficient.

A multi-label approach allows the model to assign multiple event labels to the same time window, providing a more realistic and diagnostic understanding of flight conditions. Furthermore, the dataset is highly imbalanced, with normal flight data dominating and anomalies being rare. To address this, techniques such as Asymmetric Loss and weighted sampling are used to ensure that the model learns effectively from rare but critical events.

Finally, ensemble learning is applied to combine the strengths of multiple models. By integrating predictions from different architectures, the system achieves more stable and robust performance, balancing precision and recall. This is particularly important in safety-critical applications, where both missed detections and false alarms must be minimized. These foundational concepts collectively provide the necessary background to understand the methodologies and advancements discussed in the subsequent literature review.

2.2 Review of Existing Research

The research by Li et al. [36] is an interesting strategy addressing the problem of aircraft security but with a particular focus on the susceptibility of onboard sensors to environmental risks and internal failures. The authors acknowledged that classic models of deep learning can have difficulties with venturing the complex intertwining connections between various sensors, hence proposing an improved approach to fault diagnosis, which combines GraphSage with advanced attention systems. Their approach is rather novel; they have introduced a stylized rendering of raw sensor values into Sensor Data Image (SDIs) to graphically capture both cross-sensor and temporal relationship and then interpreted these in the form of branches by the application of graph structures using K-nearest neighbor and Radius classification algorithms. The model allowed extraction of fault features to be refined significantly as it was capable of adjusting the significance of neighboring nodes by factoring in an attention mechanism. The empirical findings were remarkable, as the model suggested was able to attain a staggering accuracy of 97.80% and F1 score of 97.20% even when applied on a factorial dataset entailing actual-flight UAV data. This was an extremely higher performance than other related models, e.g. the CNN+LSTM model achieved only an accuracy of 90.63 per cent, and it also had a very fast testing time of only 0.07 seconds thus it can be incorporated in real-time application. Although the successes were made, the authors admitted that they should conduct further research to determine the application of the graph network technology to even more intricate data representations in the future. This is where our study is mentioned with reference to the fact that Li et al. concentrated rather on particular sensor errors but our research about flight anomaly detection is going to fill the gap that they found in complex data projection. We will overcome their constraints by creating a stronger structure that would be able to accommodate more extensive and complex flight deviations, and guarantee more comprehensive safety measures.

To determine the feasible maturity of the state-of-the-art techniques, Mejri et al. [26] examined a variety of unsupervised anomaly detection of time-series data

as factorial and critical. Their approach consisted of a stringent comparison of 9 different algorithms, both conventional machine learning algorithms, such as OC-SVM, as well as modern deep learning models such as MTAD-GAT on five different datasets, including spacecraft telemetry. In contrast to the existing studies that only used standard precision and recall, the authors introduced a single experimental procedure including range based measurements, model stability examination, and memory consumption measurements. It was their large-scale testing that determined that hybrid deep learning models such as MTAD-GAT tended to be really high performing - better than the rest in the case of such datasets as SWaT and MSL- but that the traditional methods were actually quite competitive and more stable. Most importantly, they have shown that the commonly used "Point Adjustment" protocol is inflating the measure of performance to a great extent that is giving an illusion of accuracy. The conclusion made by the authors was that a single method is not universally better and the selection of algorithm means that it relies heavily on the nature of the anomalies and hardware limitations at hand. One key shortcoming was noted to be the inability to be realistic to some synthetic benchmarks as well as challenges most models experienced to identify collective trend anomalies. Our investigation on flight anomaly detection will directly take these implications into account by being concerned with the unique environmental and mechanical constraints of aviation data; we will mitigate the established instability of deep learning models in safety-critical systems and create a framework that does not deteriorate under the artificial inflations of point adjustment, as gaps in this study have been indicated.

The issue of insufficiently available data obstructs the aviation safety domain, and Memarzadeh et al. [13] propose addressing it with a semi-supervised deep learning model that seeks to uncover the indicators of safety incidents. Their approach cleverly combines unsupervised feature engineering, namely, the use of Convolutional Variational Auto-Encoder (CVAE), with the supervised classification, and two different models were used, the generative M1 + M2 and the clustering-oriented CCLP. On binary takeoff and multiclass landing data based on Flight Operational Quality Assurance (FOQA) data, they experimented with this algorithm using a huge sea of unlabeled one-dimensional examples and a small island of labeled ones. The empirical results were startling; on binary problems; M1+M2 model achieved a precision of about 97.5 with a recall of 70 where only 500 labeled samples were used. Besides, the CCLP model also performed well in the multiclass scenario with a high accuracy of 72.2% on a small number of 100 labeled data, which is much greater than the state-of-the-art supervised DT-MIL model, which retained an exceptionally low accuracy of 31.4 in the same setting. Though successful, the authors realized that their validation depended much on the anomalies that are characterized by an easy threshold of exceedance, which may not accurately represent the nonlinear dynamics of the flight that are too complex to model. They recommended that subsequent versions would have to develop to be able to accept open-set or totally unknown forms of anomaly. This is where our study comes in, with their work having been limited to known forms of anomaly, our study will seek to overcome this limitation by creating a framework that can automatically identify complex, undefined anomalies without pre-defined threshold levels, and this will provide a better safety net to the current aviation environment.

Passarella et al. [27] developed a specific machine learning system that could improve the safety of aviation by detecting abnormalities in the patterns of commercial flights based on the Automatic Dependent Surveillance-Broadcast (ADS-B) data. Their approach built strongly on the idea of supervised learning where they collected a huge dataset of 167,844 737 MAX 8 flights and created a very strict protocol of labeling them that proceeded by expert pilots: a flight was termed as abnormal when it had a drop in altitude over 100 feet in a 10 second period around either takeoff or climbing. That is why, despite a rigorous benchmarking of 25 algorithms, Quadratic Discriminant Analysis (QDA) model turned out to be the most successful, with a good fit value of 93 percent accuracy of the models on the training and the testing sets. The strength of the model was also shown with its Area Under the Curve (AUC) scores, of 0.97 in abnormal detection and 0.96 in normal flights and, was even at 97 percent when it was tested against the unseen real-life accident data in a stress test. Nevertheless, the authors were open to admit that a major limitation that existed in their design was the fact that the model was mainly based on one target variable, that is, baristic altitude, and did not consider all the possibilities of flight parameters, such as velocity and heading, that according to pilots, can give a full safety picture. This weakness will give us a clear start point to our research because unlike their univariate interest in the study, our study on flight anomaly detection will incorporate a multivariate model that will investigate the altitude, speed and heading of the plane simultaneously. In such a way, we would be able to overcome their single-variable restriction and define such complicated and many-dimensional anomalies which can be overlooked with a mere check on the altitude.

The article by de Silva et al. [5] presents a physics-informed, but data-driven model to process sensor faults in a system of complex dynamics, e.g., aircraft. Their approach is a data-only learning of a linear time-invariant (LTI) model of the system dynamics by synthesizing three independent elements, namely, they use Dynamic Mode Decomposition with control (DMDc) and time-delay embeddings to learn the model. It is this learned model that provides a Kalman filter to produce real-time estimates of the state; this difference between the estimates and actual measurements, the covariance of the innovation- is the key indicator of abnormalities. Lastly, a decision tree classifier is a process that dynamically uses this covariance, and other sensor measures to classify when to signal a fault and effectively the decision process is automated. Using real flight test data on commercial flights, the authors indicated that this method was good at detecting faults in a range of 5 to 12 seconds after occurrence and the false positive was usually short lived. On synthetic tasks with two yeast models of the dynamics of Goman-Khrabrov and longitudinal flight dynamics, respectively, the system demonstrated very high accuracies of about 98 and 99 percent in case of a decision tree that was trained on a complete set of failure modes. Nevertheless, the authors did point to some major limitations: the approach is essentially supervised, and one needs labeled training data that might be limited, and the approach fails to be generalizable when the distribution of testing data changes sufficiently with respect to the training set. They also limited their existing models to the monitoring of one scalar state at once. Our study of flight anomaly detection comes into action,

here, we will solve these particular limitations by designing a structure to lessen the reliance on fully-labeled data and being capable of detecting broader, multivariate anomalies of the system other than singular, scalar sensor-based detection values provided.

The article by Jasra et al. [10] presents a powerful and hybrid approach to the aviation safety improvement, which is purposefully created to identify anomalies among the deep and voluminous flow of flight operational quality assurance (FOQA) data. They use a philosophy of divide and conquer as the basis of their methodology; to identify particular point anomalies, they first use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to cluster the flights into discrete groups based on normal working patterns, and then to train individual One-Class Support Vector Machine (OC-SVM) models over each cluster. This was a two step, unsupervised system that was firmly tested with both a synthetic input of NASA DASHlink and real world operational input of a partner airline. These results were very promising and the model made a precision of 98, a recall of 96 as well as an F1-score of 97 on the synthetic benchmark. Moreover, the system was able to detect 14 of 15 familiar safety events when applied to the real-world dataset, which equals to 93.3 per cent detection rate and the false-positive rate was low in contrast to the standalone techniques. The authors however openly acknowledged that this hybrid is computationally costly since it requires to train several OC-SVM models and also the sensitivity of the performance is very much sensitive to the value hyperparameters such as epsilon and gamma. Such a computational bottleneck offers an obvious niche in our own area of research because the authors of this study only talked about accuracy as a complex clustering, whereas ours is an attempt to deal with these peculiarities in order to create a less complicated, and more useful, computational framework that does not require the manual optimization of hyperparameters, which will enable the expedited and more scalable application in a real-time setting.

Ismail Fawaz and the research team [2] proposed a significant advancement in Time Series Classification (TSC) through their paper "InceptionTime: Finding AlexNet for Time Series Classification," aiming to establish a deep learning benchmark for time-series data with both high accuracy and scalability. The core idea of their work is the InceptionTime architecture, which is an ensemble of five deep Convolutional Neural Network (CNN) models built using Inception modules that apply multiple filters of different lengths (10, 20, and 40) simultaneously to capture both short-term and long-term temporal patterns; the model further incorporates bottleneck layers for dimensionality reduction, residual connections to enable deeper networks without vanishing gradients, and Global Average Pooling combined with ensemble voting to stabilize predictions. The results demonstrate strong performance and efficiency, where InceptionTime achieved accuracy comparable to the state-of-the-art HIVE-COTE while being significantly faster, requiring only one hour of training on a dataset of 1,500 samples compared to eight days for HIVE-COTE, and across 85 UCR datasets it recorded a Win/Tie/Loss of 40/6/39 against HIVE-COTE and 54/8/23 against ResNet(5) ($p < 0.01$), while also showing scalability by processing 8 million time series in just 13 hours. However, the authors noted limitations such as instability in individual models due to random

initialization and the risk of overfitting on smaller datasets, suggesting future work in transfer learning and fine-tuning to improve generalization across domains. This work is highly relevant to our research as it validates the effectiveness of ensemble learning and multi-scale feature extraction, but our framework extends beyond its limitations by replacing fixed convolutional kernels with patch-based attention in the Cross-Sensor Patch Transformer (CSPT) to capture sustained temporal conditions, introducing multi-label classification via MMoE for simultaneous diagnosis of multiple aviation events instead of single-sequence classification, incorporating Cross-Sensor Attention to learn inter-sensor relationships critical for flight dynamics, and improving real-time practicality by combining heterogeneous architectures (CSPT and Informer) rather than relying on a homogeneous ensemble.

Memarzadeh et al. [3] introduce the innovative, unsupervised deep learning model that is capable of detecting safety vulnerabilities in the high-dimensional time-series of Flight Operational Quality Assurance (FOQA) data. They acknowledge that the principles in aviation of using labeled data are not practical and instead their approach uses a Convolutional Variational Auto-Encoder (CVAE) to encode window-based flight data into a low-dimensional latent space through convolutional operations and then decode it back. Their new feature is the added hyperparameter of regularization, denoted as 0 improvement as 0 that trades off the bias and variance of the model, so that the model avoids overfitting to oddities and microsystems but ultimately learns nominal patterns. The empirical observations were probabilistic; in experiments where the CVAE was applied to a case study of commercial flight take-offs the model performed much better than known baselines, such as Generative Adversarial Networks (GANs) and standard Auto-Encoders and improved precision and recall by a factor of 20 and 26 percent, respectively. Additionally, they established that the training on nominal data alone could potentially kick-start their performance and that the improvement in accuracy could be seen to triple to a minimum of 37 percentage points. Although these resources have been developed, the authors have also admitted a severe weakness: their existing architecture cannot well combine heterogeneous types of data, namely binary and categorical features, and cannot give adequate interpretability to the representations in the latent space. This will present a perfect research opportunity with regards to our flight anomaly detection project; we will address these limitations directly by coming up with a more flexible framework that can still process various types of data; include discrete cockpit switch inputs, but also give more understandable results regarding what anomaly is detected.

Sun et al. [29] suggested an original deep learning system, which is named PS-AE-LSTM to closely evaluate the risks of the safety of flight that can be extracted through time-series sensor data. Their approach is a success story: they decided to build a statistical Probability Severity (PS) model to produce risk labels of high quality according to the making of normal data on flight parameters and to supplement their model with an Autoencoder (AE) to reconstruct the data and enable the inclusion of temporal information in their model with the help of a Long Short-Term Memory (LSTM) network. As a solution to challenges caused by lack of data, they adopted the Monte Carlo technique to model a strong dataset based on the authentic Quick Access Recorder (QAR) files. The experiment

findings were strong, whereby the proposed model attained the accuracy of 86.45% and the F1 score of 89.59% in the process of identifying risks of making a hard landing. It was far superior to seven baseline models, such as standard LSTM and AE-LSTM who achieved only 61.78% and 68.51 accuracy levels respectively. Nevertheless, the authors also admitted that their framework is extremely reliant on a premise that flight parameters are distributed normally and they also stated that in the future the model would need to be extended with various inputs to perform a comprehensive assessment related to flight phase. This particular weakness forms a strategic initiation point to our study; we are unlike their method in that they based their study on the strict statistical requirements of the PS model to generalize labels and in the process this would necessitate that this particular study consents to develop a framework that is able to detect anomalies in data that do not always imply that such data follows the normal distribution rule and this would place the study in a much broader setting of applicability.

Memarzadeh et al. [12] discuss the data scarcity issue as a critical bottleneck in the problem of aviation safety by proposing RESAD, a powerful and comparable semi-supervised deep learning model aimed at identifying abnormalities in flight operations. Their approach is novel in its fusion of an autoencoder model with a classifier where they optimize a composite loss term which does not only impose accurate classification and data reconstruction but in addition ensures compact clustering of the classes in the latent space through graph-based label propagation. This method enables the model to learn efficiently that one has a massive ocean of unlabeled data and only uses a very small portion of labeled instances. The empirical findings were remarkable, RESAD trained on 5.5% labeled data produced an average accuracy of 86 which was significantly high compared to the fully supervised DT-MIL model which was performing at 54.2. Moreover, RESAD achieved 65.3% accuracy even on noise levels above 10 percent, which is far above the 56.8 and 53 percent baselines of semi-supervised models (CCLP and AE + C) in tests of strong party to adversarial perturbation (notated as bb). Although these have been achieved, the authors described that their current model rests on an assumption of a closed set that is, it forces new, unknown anomalies to be clustered into existing categories as opposed to indicating them as new unknowns and it can only cluster instances under a single type of anomalies. This particular restriction opens the way to our study; although their research is outstanding at categorizing familiar risks, our study on flight anomaly detection seeks to create an open-set ability to recognize, and mark, already unidentified, or multivariate anomalies, which cannot be categorized according to existing safety standards.

The framework introduced by Zinnari et al. [23] is a highly complex stacking ensemble that is aimed at automated Flight Condition Recognition (FCR) as an essential part of the helicopter structure health and fatigue observation. These supervised learning models are developed based on the principles of a two-step architecture: first, they use a Random Forest complex analyzing the multivariate sensor data on sliding windows of different lengths (between 100 and 800 samples) to identify both multi-scaled predictions of the time types of maneuvers; and second, a smart aggregation of these multi-scaled predictions by the use of a company of small-scale predictions into a single final maneuver type is done

using a Supervisor model of logistic regression. The system was strictly tested against a huge dataset of 460 Leonardo Helicopters load survey flights and the macro-averaged accuracy of 96 and F1-score of 0.94 across 49 different flight regimes were impressive compared to simple voting schemes and one-window models. Their work also has a significant error however, learned under strict supervision, which required the manual cleaning process with a lot of labor to fix the training data mislabeled, and only allowed the model to recognize a limited number of standard maneuvers. This particular limitation can be used as a launch pad to our research endeavor; as compared to the study done by Zinnari et al. where the researchers emphasized on categorizing established behaviors of operations, our flight anomaly detection study will be oriented towards an open-set environment. We will break their weakness by creating a framework that does not only group regimes as known, but independently determines deviations and unknown abnormalities which are totally out of the spectrum of existing labels.

Yang et al. [17] address the lack of publicly available benchmarks in the field of aviation predictive maintenance with the introduction of a new data set, NGAFID-MC, as well as a new deep learning architecture. Their approach involves the construction of a large, real-world dataset consisting of more than 7,500 labeled flights performed by Cessna 172S airplanes and specifically two types of common problems (intake gasket leaks and rocker cover damage) analyzed under the conditions of pre- and post-maintenance status. They designed the Convolutional Multiheaded Self-Attention (Conv-MHSA) model to examine these long multivariate time-series they employed a 1D convolution to downsample time and then use attention mechanisms to pick up longer temporal interactions that 1D RNNs tend to overlook. Another way that they made image augmentation works, including cutout, mixup, and cutmix, in time-series prevented overfitting. These findings were convincing; Conv-MHSA model both outdid Conv-LSTM models in classification performance (e.g., an ROC-AUC of 0.826 in the larger dataset cluster) and was at least 15-fold computationally more efficient to train. Nevertheless, the authors have mentioned that there is a coloring in of the possible mislabeling in the maintenance records and the flight occurring on the very day of maintenance, and they have been doing the given approach only in respective binary classification of individual, known problems. It is this particular constraint, namely, the dependence on supervised labels of known maintenance issues, that our research overlaps; on the contrary to their target, predictive maintenance, we are interested in an unanticipated and wider-ranging framework, one that is unsupervised and able to identify an irregular behavior without prior awareness of given modalities of defective operation.

The article by LaBella et al. [34] presents a proposed novel time-series dataset, GATS, based on the National General Aviation Flight Information Database (NGAFID) to promote the development of machine learning studies in an area of general aviation safety that has not yet been thoroughly investigated. Their main methodology consisted of curating more than 10,000 hours of anonymized flight data on three different plane types such as Cessna 172S and Piper PA-44 and benchmarking this provided data satisfactorily on two tasks: plane classification and data reconstruction of missing data. To determine these benchmarks, they used a combination of supervised learning (ConvMHSA) with self-supervised methods

(SimCLR and Masked Autoencoders) assessing the ability of both architectures to deal with the complexity of sensor data of real-world contexts. The empirical findings were a dichotomy of difficulty, in that the supervised models were able to identify aircraft models with near-perfect accuracy of 99 percent, whilst the missing data reconstruction task was much more difficult the best Mean Absolute Error (MAE) of the Masked AutoEncoder was 0.34 as compared to the disastrous MAE of 4.44 of the contrastive approach. The authors very frankly acknowledged multiple limitations such as the existence of derived columns that made the models learn trivial shortcuts, a large score in missing values (NaNs), the bias of the dataset towards the student pilot behaviors implying to the commercial sector. They particularly emphasised that their existing models did not give any meaningful reconstructions using sensor failures and created a void on the safety-critical analysis. This weakness is directly applicable to our study of flight anomaly detection; we seek to go beyond their concerns of performing single-purpose classification and reconstruction to create a robust anomaly detecting system that is capable of resisting the noise and absent data that they observed, and thus be able to unlock to automatic recognition of unsafe flight patterns, of which they made only single-purpose solutions.

The paper by Nanyonga et al. [19] approaches the combination of Natural Language Processing (NLP) with aviation safety and suggests a deep learning model to automatically differentiate the flight stages using accident descriptions made in unstructured language. They used Aviation Safety Network (ASN) dataset as their methodology where they pre-processed more than 4,000 reports and trained several Recurrent Neural Network (RNN) variants, such as LSTM, GRU, and BiLSTM, separately and in hybrid similarities to find a temporal relationship among the text. The findings have revealed the effectiveness of hybrid models themselves, whereas the single BiLSTM model delivered the accuracy of 64% and recall of 64% the combined LSTM+BiLSTM architecture gave higher results and the top accuracy of 67% with the ability to distinguish between such phases as 'Landing' and 'En route'. Although these have positive results, the authors did display an honest confession that limitations of the dataset used in studying caused the models to struggle to learn more complex patterns hence limiting the effectiveness of the models. The given limitation in itself is the focus of our research since their study was limited to a limited set of texts of narrative descriptions, whereas our research work on detecting flats in the air will focus on filling this data limitation by a far more comprehensive dataset. We will address the concerns of their scalability, to create a powerful system that will have the ability to notice the anomality more accurately and generatively, using a bigger amount of data.

Mori [8], in the study "Anomaly Detection and Cause Analysis During Landing Approach Using Recurrent Neural Network," focuses on how to improve the aviation safety during the landing phase. The primary goal of the research is to find "untypical flights" that seem normal in standard conditions but may have some hidden risks. So the main idea is to model normal flight behavior and find the difference between the actual and predicted flight behavior. They used the aircraft's track angle over a 16-second window to calculate the Lateral Stability Index Ω. Also, in order to predict the index they use a GRU(512 units) along with a two

layer FFNN (256 nodes each). They use multiple inputs such as wind components, altitude, track angle, roll angle, and localizer deviation in their model. Using the sigma value, they measure the difference between the predicted and actual values and when the value is large, then it is marked as an anomaly. Their result shows a high level of performance for detecting rare type anomalies. The model was used to detect abnormal flights based on 128 A320 flight records and 5-fold cross-validation with sigma values of 0.01 to 9.99. The sigma value of one flight was approximately 10.0, which is a high deviation. It was further realized that false localizer signals influenced the autopilot behavior, leading to many anomalies. However, there are some limitations. It only focuses on lateral movement and ignores vertical flight behavior. The model also has the ability to identify anomalies but fails to automatically provide an explanation of the reasons why it occurred, so they need to do this manually. The author recommends that the model should be extended to cover the vertical parameters and bigger datasets. Also, they worked only on the landing phase. But in our work, we improve this because in our approach it identifies the anomalies throughout the flight and not only during landing. Also, here we can handle more complex, multivariate anomalies without the need for manual analysis, which is more complete and efficient solution for flight safety.

In the article "Flight Data Anomaly Detection and Diagnosis with Variable Association Change," He et al. [7] focus on a more sophisticated form of anomaly in aviation systems. They do not just identify the abnormal values, their work focuses on the change in relations among various sensors, which is significant to detect the important and underlying faults of the system. The main idea to detect the anomaly is to analyze the change in the sensor relationship over time. They suggested a model named DAVAC (Detection and Diagnosis of Anomaly with Variable Association Change). The methodology consists of three primary modules: Residual Network (ResNet) is trained to learn single sensor patterns; a nonlinear Granger causal graph is trained to learn sensor relationships and regression is used to forecast future values. While detecting an anomaly using the high prediction error, a second model is trained on anomaly data and then using the difference between normal and abnormal graphs it is possible to identify which sensor relationships have changed. The result shows DAVAC (Detection and Diagnosis of Anomaly with Variable Association Change) performs better than the existing methods currently have. In 40-100 variable simulations, it obtained approximately 13 percent higher AUPR than the best model (OmniAD). It was able to detect root causes of various abnormal flights on real flight data of 47 sensors, unlike the models such as the MSCRED and cLSTM. It also showed more than 85 percent of improvement in identifying sensor relationship changes. However, there are some limitations. In their study, they assume that anomalies do not change with time but this might not be the case with dynamic faults. In addition, for the diagnosis process, it needs to gather several abnormal samples, which causes real-time diagnosis delay. They focus on the importance of both detection and diagnostics, but their method requires a two-step process and is less suitable for real-time use. Our approach enhances this by detecting and classifying without retraining. Moreover, our solution will be able to manage dynamic and changing anomalies and identify problems during the entire flight, which will be more faster and practical in a real-time aviation safety system.

In the paper titled Aircraft Flight Movement Anomaly Detection using Automatic Dependent Surveillance-Broadcast, Ajhari and Negara [9] discuss the application of the ADS-B data to identify flight anomalies. They mainly focus on identifying unusual flight movement patterns using the publicly available real-time flight tracking data. Their main goal of the research is to identify the anomalies through deep learning models on the basis of flight movement. The authors gathered ADS-B data in Flightradar24 and examined 14 actual crash scenarios, such as Sriwijaya Air SJ182 and Lion Air JT610. They directly can not use the raw data so the methodology includes data cleaning, interpolation, and a sliding window approach (20-second window with 5-second step). There are seven features which are latitude, longitude, altitude, speed, heading, vertical speed, and time. They used four models LSTM, GRU, Bi-LSTM, and Bi-GRU and from the result they found out that the bidirectional models work better than the standard ones. The Bi-LSTM model was found to have 99.44% accuracy and 99.51% F1-score and a precision of 99.48 and a recall of 99.55. It had a high effectiveness as it only made 65 mistakes out of 11,661 test samples, which are movement-based anomalies. However, there are some limitations. They are working with the external movement data with some limited features as a result they are unable to detect internal aircraft faults. They also mostly focused on the trajectory-based anomalies of the aircraft without finding deeper system-level issues. Their work shows that deep learning can effectively detect flight anomalies but their approach is limited to external movement patterns. In our work, we overcome these limitations by using internal multivariate sensor data allowing to detect deeper internal faults. Also, our model not only can do binary classification but also identify different types of anomalies. Furthermore, it can work throughout the whole flight and work with the real-world data imbalance which makes our approach more suitable for the aviation safety system.

Rong et al. [15], in their work tried to improve the aviation safety by detecting anomalies from high-dimensional flight data recorded by Quick Access Recorders (QAR). In order to detect the anomalies, the main challenge they address is the challenge of handling large and complex flight datasets. In their work, they combined different deep learning models and proposed a hybrid framework consisting of three parts, which are a Variational Autoencoder (VAE) for feature extraction and dimension reduction, an LSTM to capture temporal patterns, and a Multihead Self-Attention mechanism to focus on important time steps. In their approach, they used only the normal flight data to train the model, and to detect the anomaly, they used the reconstruction error between the original and predicted values. Their model works good in different flight phases for example in the climb phase models F1 Score is 0.9503 (precision 0.9145, recall 0.9833). Also, during the cruise and descent phases, the model achieved a recall of 1.0 which the model was able to detect all true anomalies. Moreover, because of using attention, it helps to reduce training time (around 1.2–1.75 minutes per epoch) which makes it more efficient than similar models. However, there are some limitations. To train their model they heavily need to depend on the clean training data also any hidden anomalies in the dataset can affect the model's performance. Then they mainly focus on the engine-based parameters and use a simple threshold-based detection

approach. From their work it shows how attention-based models can be effective in anomaly detection. But their approach is limited to the binary detection and uses a small feature set. Here, our study improves this by using multivariate sensor data and detecting multi-label classification which helps to identify different types of anomalies. There is a huge imbalance in the aviation dataset our model handles this imbalance more effectively and detects anomalies throughout the whole flight, which makes it more suitable for real-world aviation safety applications.

Qin et al. [14], in their study work on improving the aviation safety through moving from post-accident analysis to early detection of anomalies, especially during the landing phase. Their main approach was to combine deep learning with the clustering techniques. Their proposed Time-Feature Attention-based Convolutional Autoencoder (TFA-CAE) has two types of attention one is time attention for detecting when anomalies occur and another is feature attention to identify which sensors are responsible for the anomalies. The HDBSCAN clustering algorithm showed a strong performance in feature extraction and anomaly detection, without requiring the labeled data, where it achieved a very low reconstruction error of 0.0009, which outperforms models like SA-CAE (0.0013) and GRU-AE (0.0016). They used 14,195 flight records, and among them, around 825 anomalies were detected by their system. Also, it can find the difference between different landing patterns and identify real issues such as sensor errors and low-power approaches. However, there are some limitation like they can only detect errors for different runways and work only for airport level. Then the clustering result needs manual checking which limits real-time usability. Though their work highlights the attention mechanism and multivariate data but their clustering results also need human involvement, and they only work for airport-level anomalies. In our study we provide automatic classification of anomalies without human intervention. Also our model is capable of capturing more complex temporal patterns and can handle the data imbalance more effectively. Then our system can detect anomalies throughout the whole flight, not only for a specific phase.

Wang and Li [32] in order to improve time series anomaly detection, worked on capturing the information from the past and future data points. Also, they focus on overcoming the limitations of traditional one-directional models. Their main idea was to use a hybrid deep learning model called BiTCN-MHA, where it has three main components. The first one is a Bidirectional Temporal Convolutional Network (BiTCN) used to extract the features from both backward and forward sequences, then to improve the learning stability they used an ELU activation function and lastly a Multi-Head Attention (MHA) mechanism so that the model can focus on important features and reduce the noise. Also, to improve the robustness the model uses data enhancement by adding white noise as a result, their model performs better than many existing methods. Across six benchmark datasets, it improved Precision by 6.10%, Recall by 10.16%, AUC by 4.06%, and F1-score by 8.50%. Also, they achieved an AUC of 0.93453 and an F1-score of 0.89206 on the NAB dataset, outperforming many models like TranAD and USAD. but there are some limitations like their models did not show good performance on larger datasets. Also, as the model depends on standard loss functions it can not handle complex data patterns effectively. This is why the authors suggested to improve the loss

function and model structure in the future. In their work they highlighted the importance of temporal modeling and attention mechanisms but focused only on binary anomaly detection. Here, our study improves this by intriducting multi level classification and identifying specific types of anomalies. Also, our model can interact between multiple sensors and capture longer temporal patterns more effectively. We also handled real-world data imbalance and anomaly detection throughout the whole flight effectively.

Liu et al. [37], in their work introduce a new framework called FLIGHT2VEC for analyzing flight trajectory data. This can be used to learn the general representations of flight movement, which can be used to do prediction and perform multiple tasks, including the detection of anomalies. Their main idea was to improve how flight trajectory data is processed so they proposed a behavior-adaptive patching mechanism, where important segments of the flights are given more focus and less important parts are reduced. Also, they introduce motion trend learning which is a self-supervised method that predicts flight direction categories for better understanding of flight behaviour. Using this approach shows strong performance across different tasks as it helps the transformer model to learn the meaningful patterns instead of raw coordinates. The model achieved an AUC of 0.9201 and AUPR of 0.9072, outperforming other models like DDM for anomaly detection. Also because of its short processing time of 3.11 ms it is highly efficient for fast applications. But there are some limitations to their model, which mainly work on the trajectory and movement data but do not work on the internal sensor data of an aircraft. As a result, it can not detect the internal anomalies of a flight. But in our study, we improve this by using multivariate internal sensor data, which helps to detect deeper system faults. Also, our model can perform multi-level classification and can work throughout the whole flight, making it more appropriate for real-world use.

In aviation, the time series multivariate data is very complex and it is one of the reasons for the reduction of model performance. In order to solve this problem, Zhao et al. [33] in their study focus on improving anomaly detection in high-dimensional sensor data. Their main idea was to combine graph-based learning with transformer-based forecasting. So they proposed a hybrid model where they use a Graph Attention Network (GAT) with a GRU that will be used to capture short-term patterns and relationships between sensors. This will be combined with an informal model to learn long-term temporal dependencies. In their model, they use a joint optimization approach, where they calculate the short-term and long-term prediction losses, and based on that prediction error, they will improve the anomaly detection. The results show that their approach performs good on the industrial dataset. They achieved an F1 score of 0.82 on the SWaT dataset and 0.65 on the WADI dataset without any adjustment. But with point adjustment, the results show much improvement where the F1 score increases to 0.91 (SWaT) and 0.87 (WADI). Moreover, the model improves recall by 30.33% and F1-score by 14.03%, which indicates it performs well on the high-dimensional data. They suggested the use of informer and attention-based models for multivariate data. But there are some limitations, like the model they used is much more complex and uses a large number of parameters which can affect the efficiency and real-time use. As a result, they suggest creating a lighter version of the model in future work.

Then in their study, they focus only on anomaly detection, but we improve this in our study by performing multi-label classification, where we can identify a specific anomaly type. Also, our model can handle data imbalance and work throughout the whole flight, making it more practical for aviation safety applications.

He and Zhao [1], in their study, work on anomaly detection on multivariate time series data using deep learning. They focus on improving the efficiency and performance compared to the traditional sequentia models. Their idea was to use a Temporal Convolutional Network (TCN) to learn the normal patterns from the flight time series data and detect anomalies based on prediction errors. In their methodology, they include causal convolutions to prevent future data leakage for capturing long-term dependencies, including dilated convolutions and residual connections for stable training. They used multi-scale feature mixture where they combine outputs from multiple layers to capture patterns at different levels. They detect anomalies by comparing the predicted and actual values, and also calculate the anomaly score using Gaussian distribution. The result shows that the model they proposed, a multi-scale TCN, can perform better than the standard TCN models. After applying the model on the ECG dataset, they achieved a precision of 0.946 and an F-score of 0.908 also when they used it on the Space Shuttle dataset it achieved a precision of 0.880 and an F-score of 0.874. Though the recall was low, indicating many anomalies were missed. In their work, they supported the use of temporal modeling for anomaly detection. However, there are some limitations there model is sensitive to window size as a result, it can affect the performance. Also, as this is unsupervised, the model cannot understand the complex anomalies. Here, our model improves this by capturing the complex relationship between multiple sensors more effectively. Also, our model can perform multi-label classification and identify a specific type of anomaly. We also improve recall and can detect anomalies throughout the whole flight time.

Gopali et al. [6] in their work focus on comparing different deep learning models for anomaly detection in multivariate time-series data and found that whether LSTM or TCN performs better in such tasks. As a result, their main goal was to compare LSTM (Recurrent Neural Network) and TCN (Temporal Convolutional Network) models. In order to do so they used the SWaT dataset with 14,996 data points and built three models: a TCN, a 1-layer LSTM, and a 3-layer LSTM. They mainly used a threshold-based method for anomaly detection, where a data point is considered to be abnormal if the prediction error value is more than 1.75 times the standard deviation. The result of the TCN model is comparatively better than LSTM, where it achieved an F1-score of 0.920 (precision 0.939, recall 0.903) with a training time of about 5 hours 55 minutes, but LSTM achieved an F1-score of 0.901 but took longer (6 hours 28 minutes). Though there are some limitations, like the LSTM models are slow and computationally expensive also for the complex dynamic patterns, the TCN may struggle to give better performance. As a result, they suggested to use hybrid model in the future. Though they show how the convolution-based models are more effective than the recurrent ones but their work is limited to binary anomaly detection using simple thresholds. Here, our study improves this by using attention-based models that can capture complex relationships between multiple sensors. Furthermore, our model can perform multi-label classification and identify

a specific type of anomaly throughout the entire flight.

2.3 Summary of Key Findings

The overall analysis of the available literature shows that there is a definite change in aviation safety research, where the conventional models of aviation safety are replaced by the new systems of anomaly detection which are based on deep learning. The earlier approaches were largely based on distribution modeling and clustering, including the hybrid DBSCAN and OC-SVM method by Jasra et al. [10], which worked well on simple anomalies but failed with high-dimensional flight data and could not even interpret complex safety events. In recent times more modern architectures such as Recurrent Neural Networks such as LSTM [29], Temporal Convolutional Networks as investigated by He and Zhao [1] and Gopali et al. [6], and Transformer-based models such as FLIGHT2VEC by Liu et al. [37] have been introduced. These models have shown a great result in capturing nonlinear temporal patterns that are present in flight telemetry data.

The other significant trend that can be seen throughout the literature is the shift of univariate analysis to multivariate time-series analysis. Previous research mainly focuses on only one parameter such as altitude as in with ADS-B trajectory-based solutions [27], whereas recent research points out that aviation anomalies are typically brought about by the combination of several sensors. With this advancement, numerous studies rely on external surveillance information because they do not have internal systems-level data to identify failures at their initial stages [27], [37]. Conversely, those that make use of internal QAR data, including Rong et al. [15], demonstrate high detection capabilities but are frequently restricted to particular data, such as engine data. This implies that there is a need to have models that are capable of handling full multivariate sensor inputs and that kind of data in order to gain a better understanding of the system-level.

From the literature, it is seen how architectural improvement is important also the use of hybrid models and attention mechanisms. In order to focus on important time steps and learn relationship between different sensor techniques like time feature attention and cross-sensor attention gives good results. This has been successfully applied to models like TFA-CAE by Qin et al. [14] and the GAT-Informer framework by Zhao et al. [33]. Nevertheless, there is still a significant drawback as most of these systems are binary classification, and it just can only detect whether a data point is normal or abnormal. This is not enough in the real world aviation situations, where it is important to know the type of anomaly. As Li et al. [36] said, no models that offer automatic diagnostic results or do multi-label classification throughout the whole flight exist.

Another significant issue that is addressed in the literature is class imbalance. Flight data is very imbalanced, and anomalies are very uncommon. Research like Memarzadeh et al. [13] indicate that because of this problem, supervised and semi-supervised approaches are used, which may not be able to identify minor anomalies. Also, the use of evaluation methods in certain works are misleading. Mejri et al. [26] showed that in such method, techniques like point adjustment can

artificially raise the performance matrix. It is also observed in other studies [5], [29] that models using strict assumptions like normal distributions or fixed physical constraints can fail to work in real world situations where the behavior of flights is nonlinear and complex.

Lastly, most of the existing works are only focused on certain stages of flight, such as takeoff or landing [27], [14], or other flight conditions as observed in [23]. These approaches are not capable of detecting throughout the flight. In addition, most of the models need manual analysis once they detect any anomaly, which makes them less useful in real-time processes.

These results demonstrate the necessity of a more comprehensive framework that will be able to accept multivariate internal sensor data, manage extreme imbalance in classes, and enable multi-label diagnostic results in real time. Such a system would not just detect anomalies, but also enhance the aviation safety analysis.

Chapter 3

Requirements, Impacts and Constraints

3.1 Final Specifications and Requirements

This section summarizes the technical, methodological and resource needs that informed the design and development of the two-stage aviation anomaly detection framework. The specification discusses the hardware environment on which the training and experimentation will occur, the software stack and its critical dependencies, the desired behavior that the deployed system will assume, and the performance goals at the beginning of the research.

3.1.1 Hardware Requirements

All the model training, hyperparameter optimization, and assessment were executed on a local workstation. The hardware configuration was chosen to enable large-batch deep learning training with mixed-precision computation, high throughput loading of data in a high-throughput storage device, and to have enough memory to store the entire dataset, multiple model checkpoints in memory at once.

Table 3.1: Hardware configuration used for model training and evaluation.

Component	**Specification**
Processor (CPU)	Intel Core i7-14700K (20 cores / 28 threads, up to 5.6 GHz boost clock)
Graphics Card (GPU)	NVIDIA GeForce RTX 4070 Super, 16 GB GDDR6X VRAM
System Memory (RAM)	64 GB DDR5
Storage	1 TB NVMe SSD
Operating System	Windows 11

The GPU was used exclusively during training to accelerate matrix operations through PyTorch's Automatic Mixed Precision (AMP) backend, which performs forward and backward passes in float16 and stores weights in float32. All inference latency measurements reported in this thesis were taken on the CPU only, since the target deployment environment is a standard ground-station computer or embedded

avionics hardware where a GPU is not guaranteed to be available.

3.1.2 Software and Framework Requirements

The framework was developed entirely in Python. The following libraries and tools were required to reproduce the training pipeline, evaluation, and visualization components of this research.

Table 3.2: Software stack and key dependencies.

Library / Tool	**Version**	**Purpose**
Python	3.10	Core runtime environment
PyTorch	2.1 (CUDA 12.1)	Model definition, training, and inference
scikit-learn	1.4	Preprocessing, metrics, and threshold sweep
pandas	2.1	Dataset loading and flight-level partitioning
NumPy	1.26	Numerical computation and window construction
Matplotlib / Seaborn	3.8 / 0.13	Training curves, confusion matrices, and analysis plots
CUDA Toolkit	12.1	GPU acceleration for training

3.1.3 Methodological Requirements

The following technical requirements were established to ensure that the system is correct, reproducible, and suitable for the aviation safety domain.

- **Reproducibility:**

  All random sources (Python, NumPy, PyTorch CPU, and PyTorch CUDA) are fixed with a common seed of 42 at the start of every training run, ensuring that weight initialization, data shuffling, and dropout patterns are identical across runs. Deterministic behavior is a requirement in safety-critical research where results must be independently verifiable.

- **Flight-level data partitioning:**

  All timesteps from a given flight must be assigned to exactly one split (train, validation, or test). Mixing timesteps from the same flight across splits would allow label leakage and produce optimistic evaluation results that do not reflect genuine generalization to unseen flights.

- **Class imbalance handling without augmentation:**

  The framework must address the 339:1 normal-to-anomaly imbalance in Stage 1 and the per-class imbalance in Stage 2 entirely through loss function design and sampling strategies. Synthetic augmentation of time series sensor

data was excluded because augmented samples may not reflect realistic flight physics, which could cause the model to learn patterns absent from real operations.

- **Internal normalization:**

  All three models must store sensor normalization statistics as fixed non-learnable parameters so that raw sensor readings can be passed directly at inference time. This eliminates the external preprocessing dependency that would otherwise be required to match the training-time scaling applied to the data.

- **Validation-only threshold selection:**

  Decision thresholds for both Stage 1 and Stage 2 must be selected using the validation set exclusively. The test set must never be used for threshold tuning, ensuring that reported test metrics reflect performance on genuinely unseen data.

3.2 Societal Impact

- **General Aviation Safety Enhancement:**

  The key social contribution of this study is how to make general aviation safer as this is an ongoing issue in the fight against aviation accidents. Accident rates in general aviation are by no means less than those in commercial aviation, which has the same regulatory controls and the same fundamental operation procedures. Loss of control-based accidents and incidents of stalls and extreme attitudes are a common cause factor that can be prevented by taking prompt actions to rescue the situation by the pilot. This anomaly warning system supports pilots with real-time decision-making dealing directly with these causal factors of accidents.

  The benefit to the society is not just limited to accident avoidance but also pilot confidence and accessibility of operations. Instrument meteorological conditions, high-altitude flight Pilots practicing under difficult conditions at night, under the influence of cognitively demanding conditions are forced to monitor numerous parameters at the same time. This workload can be minimised through real-time anomaly alerts whereby pilots can be able to concentrate on strategic decisions, as automated systems keep a lookout on possible tactical hazards. In the case of pilots who receive less training (recreational or private pilots) automated alerting helps offset the lack of experience, enhancing margins in safety and allowing them access to more operations.

- **Democratization of Advanced Safety Technology:**

  The second impact on society will be to democratic and advanced safety technology that was once accessible to only commercial aviation operators. Commercial aviation spends a lot of money on complex flight monitoring systems,

terrain awareness and warning systems (TAWS), traffic collision avoidance systems (TCAS), and predictive maintenance systems. Such systems are very inaccessible to general aviation (cost and complexity of certification). This study brings about anomaly detecting research specifically tailored to cost-efficient retrofit to the currently used general aviation airplanes and extend advanced safety technologies to the wider community of aviation. The resulting system is much cheaper than its commercial counterparts but it meets general aviation-specific safety issues.

- **Data-Driven Safety Culture:**

  Installing such system facilitates a shift towards data-driven safety culture in general aviation. In the past, the safety in general aviation has been improved based on incident reporting systems and accident investigations which bring about lessons learned throughout the community. Although useful, such a reactive means cannot be used to avoid accidents, but rather help to learn the lesson of the past mistakes. The capability to detect anomalies through automated methods facilitates strategic monitoring of safety so that the emerging trends of safety can be detected before they lead to accidents. Statistically, the aggregated, anonymized data of several planes offers power to realize patterns of safety and emerging risks, as well as evidence based safety interventions.

- **Economic and Social Accessibility:**

  Its economic effects are felt in the availability of general aviation service. General aviation may play vital roles such as agricultural, emergency medical services, search, and rescue, and regional connectivity in under-served regions. The better safety systems can help insurance premiums can be lowered, as well as the reliability of operations, minimizing costs and increasing service availability. In countries with low development levels especially, general aviation offers vital connectivity; enhanced safety decreases obstacles to growth of such vital services.

3.3 Environmental Impact

- **Operational Efficiency Improvements:**

  Although the main idea of this anomaly detection system is to improve safety, there are the secondary environmental effects arising due to the improvement of operational efficiency. An appropriate anomaly detection to help pilots identify emerging dangerous scenarios and take corrective measures will lead to efficient flight paths and lower unwarranted altitude corrections or deviations. Smooth flight operations help decrease the consumption of fuel, which has a direct negative effect on the amount of carbon emissions and the environment as a whole. Furthermore, accident prevention removes environmental expenses such as clean up costs of accidents, cost of recovered debris and possible environmental pollution of aircraft wreckages in isolated locations.

- **Lifecycle and Energy Considerations:**

  The computational infrastructure that will enable the use of this system will need energy to model train, process inferences, and store data. Training

involves computation intensive in terms of GPUs; but this training is only performed once in system development. Operational inference operates on relatively small GPUs (consumer grade NVIDIA GPUs which use 100-300 watts) or even CPU based systems, which is a small increment over the power requirements of avionics systems that already exist on aircraft. The model sizes (must be less than 6MB in total) and latency in inference (less than 1ms) imply small amounts of energy usage in operational use.

The production of computing hardware used to implement this system has embodied environmental costs which are common in electronics production. Nonetheless, the reuse of the system by several aircraft fleets offers the benefit of amortization: the same designs can be used by hundreds of aircraft, sharing the development environmental cost over a large number of units. Additional factors include long service life (10+ years of avionics systems), decreasing per-flight-hour environmental cost.

- **Accident Prevention Environmental Benefits:**

  Accident prevention gives a great advantage to the environment. The airplane accidents in remote or environmentally sensitive locations cause aircraft fuel, hydraulic fluids, and materials pollution. Collateral environmental damage may occur as a result of rescue and recovery operations done in case of accidents. The system reduces these environmental costs because it avoids accidents, which offer net environmental benefits, even after the energy used in the operation of the system.

- **Sustainable Aviation Considerations:**

  With aviation slowly moving towards a sustainable aviation fuel (SAF) and electrification, anomaly detection systems gain significance concerning the efficiency of operations in limited-energy conditionings. The lack of energy onboard and the impossibility of inefficient flight operations make electric aircraft expensive. Real-time anomaly detection that allows optimum flight procedures will be even more useful as conditions of aviation sustainability become more stringent.

3.4 Ethical Issues

- **Algorithmic Bias and Fairness:**

  One of the major ethical issues is the possibility of findings anomalies due to algorithmic bias in anomaly detectors. Machine learning algorithms may implicitly capture biases that exist on training data. When training data is skewed towards some type of aircraft or state of operation, the model will not be as predictive in less-represented conditions, posing unsafe scenarios to pilots flying low-represented aircraft. The system deals with this by ensuring consistency of performance of all monitored types of event (per-event F1 variance ≤ 0.05) using explicit performance validation per event and by testing across different aircraft types in beta deployment.

But, there are more fundamental fairness issues, does the system deliver the same benefit in safety, even in case some groups of pilots (e.g., rural general aviation operators, international operators) have dissimilar accident profiles or operational features? Ethical implementation involves methodical research to understand whether the system will offer equitable safety improvement among varied groups of pilots and varied operations.

- **Pilot Autonomy and Over-Reliance:**

  The secondary ethical issue is connected with possible deficiency of situational awareness in pilots in case of excess use of automated alerts. In case pilots get to rely solely on the anomaly detection system, defeating degradation would diminish the safety margins. Also, false alarms may unintentionally compromise trust in the system by pilots, which may result in pilots turning off the alerts, or ignoring the warning (alert fatigue). Ethical deployment needs to be closely human factors engineered, with alert design that minimizes false positive rate and pilot training that stresses that automated alerts are additive, not substitive.

- **Data Privacy and Monitoring Implications:**

  Onboard anomaly detection systems implementation will produce continuous flight data, which increases privacy concerns. Flight data were traditionally stored on board aircraft and could only be accessed by pilots. It opens opportunities to monitor pilot behavior with integration with ground-based systems to perform analysis, monitor performance, or compliance with regulations. Ethical deployment demands they mask information processing methods, pilot approval procedures ought to be clear, and data security strong to block data use in unethical manners like evaluation of pilot performance or penalty action instead of focusing on flight data as a safety measure.

- **Regulatory and Liability Considerations:**

  This leads to ethical issues about responsibility distribution in case of failure or warning errors by the anomaly detection system. Is it the people working on the system, aircraft makers, or pilots that are liable to such unfortunate events? Existing liability frameworks are not yet developed around AI-based safety-critical systems. Ethical deployment would also be impossible without explicit regulatory schemes that determine the allocation of responsibility and insurance against agreements with improper liability on all stakeholders.

- **Accessibility and Equity in Safety Innovation:**

  A major moral aspect is a fair access to high-level safety technology. When the system is introduced on higher-end aircraft and not yet implemented on low-end aircraft flown by recreational pilots or flight schools, safety gains are clustered among well-off operators without benefiting high-risk operators. Deployment in an ethical manner focuses on providing technology to all classes of aircraft and operating categories, making sure that a safety enhancement is provided to the aviation community as a whole, rather than just benefit concentrated.

- **Informed Consent in Data Usage:**

  The validating training and testing of this system needed historical flight data with definite labels of anomalies. Pilots who have provided this data might not have given a direct consent that their data can be used to develop a machine learning model. Although anonymization is deemed to offer a certain level of privacy, some ethical issues about informed consent exist. To improve transparency in communication on data utilization, overt consent procedures, and coherent explanation of machine learning algorithms and possible restrictions, frontline implementation should take priority in the future.

3.5 Project Management Plan

The project follows a structured four-phase approach over 16 months:

Phase 1: Model Development and Validation (Months 1–4):
Develop and validate CSPT v8, HiCST, and MHANet v2 architectures. Evaluate 10 ensemble strategies, optimize decision thresholds, and establish performance baselines.

Phase 2: Integration and System Development (Months 5–8):
Integrate the two-stage system architecture, define avionics interfaces, implement real-time inference engine, and test on avionics hardware.

Phase 3: Certification and Regulatory Approval (Months 9–12):
Obtain DO-178C Level C certification, conduct failure mode and effects analysis (FMEA), develop pilot training materials, and secure regulatory approval.

Phase 4: Pilot Testing and Refinement (Months 13–16):
Deploy on 5–10 instrumented research aircraft, collect real-world performance data, integrate pilot feedback, and document lessons learned.

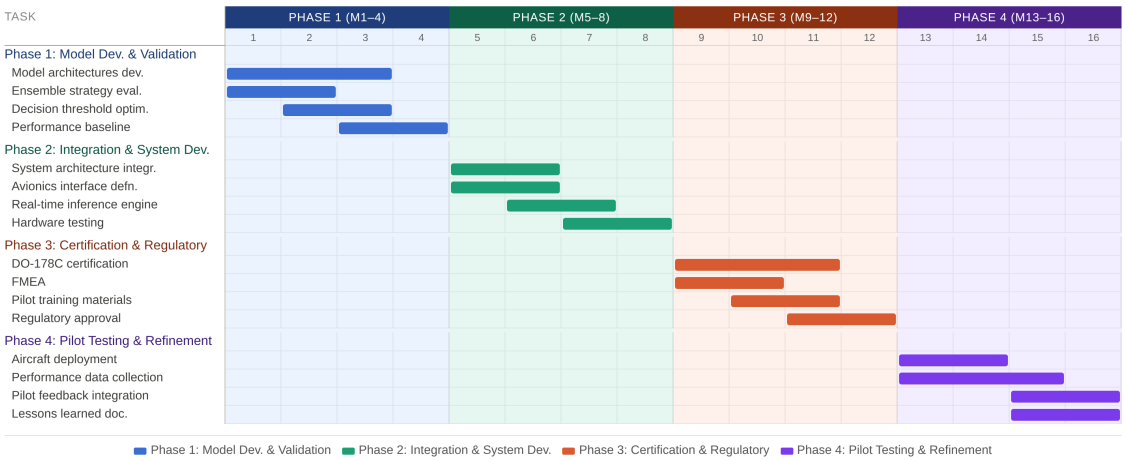

Figure 3.1: Gantt Chart of Project Management Plan

3.6 Risk Management

3.6.1 Technical Risks

3.6.1.1 Model Performance Degradation in Operational Deployment

- **Description:** Despite achieving target performance on test data, model performance may degrade when deployed to operational aircraft due to distribution shift (different sensor characteristics, maintenance states, aircraft configurations not represented in training data)
- **Probability:** Medium (40%)
- **Impact:** High (critical safety function degraded)
- **Severity:** High
- **Mitigation:** Implement continuous model monitoring tracking performance metrics on operational flights; establish retraining pipeline enabling rapid model updates if performance degrades; conduct systematic testing across representative aircraft variants during beta phase; maintain human-in-the-loop procedures for alert validation
- **Contingency:** Fallback to previous validated version; restrict deployment to aircraft types with adequate validation data

3.6.1.2 False Positive Rate Higher Than Acceptable

- **Description:** System generates excessive false positive alerts (incorrectly identifying normal flight as anomalous), causing alert fatigue and pilot disregard of genuine alerts
- **Probability:** Medium (35%)
- **Impact:** High (undermines system effectiveness and pilot trust)
- **Severity:** High
- **Mitigation:** Implement aggressive false positive testing; optimize decision threshold balancing precision and recall; introduce alert hysteresis (requiring 2+ consecutive seconds of anomaly classification before alert); conduct pilot usability testing evaluating alert frequency and pilot responses
- **Contingency:** Increase decision threshold increasing precision at cost of recall; implement pilot-configurable alert frequency settings

3.6.1.3 Critical Events Missed by Detection System

- **Description:** System fails to detect rare but critical events (High Altitude Spin, specific combinations), resulting in missed safety alerts
- **Probability:** Low (15%)
- **Impact:** Critical (direct safety consequence)

- **Severity:** Critical
- **Mitigation:** Implement per-event performance monitoring ensuring F1 $\geq$ 0.81 for all events; conduct systematic testing of rare event combinations; maintain ensemble backup detection approaches; establish explicit minimum performance thresholds for each event type with automatic degradation warnings if thresholds approach
- **Contingency:** Revert to simpler rule-based detection for events where ML performs inadequately; implement conservative default behavior (alert) for undetected patterns

3.6.1.4 Inference Latency Exceeds Operational Requirements

- **Description:** Real-time inference on avionics hardware exceeds 1ms target, introducing unacceptable delays between event occurrence and pilot alerting
- **Probability:** Low (20%)
- **Impact:** Medium (delayed alerts reduce effectiveness)
- **Severity:** Medium
- **Mitigation:** Conduct early latency testing on target hardware; implement model quantization (FP32→INT8) if needed; establish parallel processing pathways; optimize implementations for target GPU architecture
- **Contingency:** Deploy on ground-based system processing delayed data stream; reduce temporal resolution or window size reducing detection capability but enabling latency targets

3.6.2 Regulatory and Certification Risks

3.6.2.1 Regulatory Disapproval or Certification Delays

- **Description:** Regulatory authorities (FAA, EASA) may find ML-based anomaly detection insufficiently validated for safety-critical applications, delaying or preventing certification
- **Probability:** Medium (30%)
- **Impact:** High (prevents operational deployment)
- **Severity:** High
- **Mitigation:** Early regulatory engagement through Advisory Circular process; engage regulatory consultants experienced with AI certification; implement comprehensive validation exceeding minimum requirements; maintain detailed documentation of design rationale and validation; conduct independent third-party verification
- **Contingency:** Reposition system as non-essential advisory (Level D certification) rather than safety-critical function; restrict deployment to experimental/research aircraft under special flight permits

3.6.2.2 Liability and Legal Challenges

- **Description:** Accidents involving false alerts or missed detections trigger liability claims or regulatory investigations creating legal and financial consequences
- **Probability:** Low (20%)
- **Impact:** Very High (extensive legal/financial consequences)
- **Severity:** Very High
- **Mitigation:** Maintain comprehensive liability insurance; establish clear disclaimers regarding system limitations; document all testing and validation procedures; maintain clear responsibility allocation between developers, manufacturers, and operators; establish clear operating procedures and limitations
- **Contingency:** Maintain legal defense fund; implement conservative operational restrictions until liability precedents clarified

3.6.3 Operational and Safety Risks

3.6.3.1 System Enables Unsafe Operational Practices

- **Description:** Pilots relying on anomaly detection system may push aircraft beyond intended operating envelopes, enabling operations that would be unsafe without system backup
- **Probability:** Medium (35%)
- **Impact:** High (system enables unsafe operations)
- **Severity:** High
- **Mitigation:** Implement clear system limitations documentation; conduct comprehensive pilot training emphasizing that system supplements rather than replaces judgment; establish automated operational envelopes enforcing aircraft limits regardless of system status; maintain pilot awareness training
- **Contingency:** Implement regulatory restrictions on operations enabled by system; establish gradual deployment allowing operational practices to normalize before full deployment

3.6.3.2 Alert Fatigue and Pilot Disregard

- **Description:** Excessive or poorly calibrated alerts cause pilots to disregard system warnings, reducing effectiveness during genuine emergencies
- **Probability:** Medium (40%)
- **Impact:** High (system loses effectiveness)
- **Severity:** High

- **Mitigation:** Aggressive false positive optimization; implement alert prioritization (critical/high/medium) matching alert importance; implement temporal hysteresis reducing alert chatter; conduct pilot usability studies; establish alert customization enabling pilot configuration
- **Contingency:** Implement automatic alert suppression after repeated false alerts, with manual override enabling reset; conduct emergency alert retraining when alerts suppressed

3.6.4 Data and Privacy Risks

3.6.4.1 Data Breach or Unauthorized Access to Flight Data

- **Description:** Flight data containing flight parameters and pilot behavior attracts security interest; unauthorized access could enable surveillance or competitive intelligence
- **Probability:** Medium (25%)
- **Impact:** High (privacy violation, potential safety compromise through tampering)
- **Severity:** High
- **Mitigation:** Implement end-to-end encryption for all data transmission; establish access controls limiting data access to authorized personnel only; conduct regular security audits and penetration testing; implement data anonymization for non-real-time analysis; establish incident response procedures
- **Contingency:** Implement local processing without external communication; fallback to onboard-only data storage without ground transmission

3.6.4.2 Pilot Privacy Concerns and Consent Issues

- **Description:** Pilots express concerns regarding continuous monitoring and data collection, creating operational acceptance problems or regulatory pushback
- **Probability:** Medium (30%)
- **Impact:** Medium (reduces operational acceptance and deployment)
- **Severity:** Medium
- **Mitigation:** Establish transparent data handling policies with clear pilot communication; implement explicit consent procedures; provide pilots access to their flight data; establish data retention limits; conduct privacy impact assessment; engage pilot representatives early in development
- **Contingency:** Implement opt-out procedures for pilots declining monitoring; restrict deployment to willing operators until concerns resolved

3.6.5 Business and Market Risks

3.6.5.1 Market Adoption Slower Than Projected

- **Description:** Aircraft owners may resist retrofit costs or perceived operational complexity, limiting market adoption below business case projections
- **Probability:** Medium (35%)
- **Impact:** Medium (reduced revenue and business viability)
- **Severity:** Medium
- **Mitigation:** Conduct market research validating customer demand; establish competitive pricing; partner with aircraft maintenance organizations enabling easy integration; develop clear business case demonstrating return on investment through reduced insurance premiums and improved operational efficiency
- **Contingency:** Pursue alternative deployment through flight schools, air taxi operators, or agricultural aviation where safety premium justifies higher costs

3.7 Economic Analysis

3.7.1 Cost-Benefit Analysis

The economic viability of this system rests on comparison of implementation costs against safety and operational benefits:

3.7.1.1 Implementation Costs:

- Hardware (GPU module, integration hardware): $3,000–5,000 per aircraft
- Software licensing and support: $500–1,000 annually per aircraft
- Installation and certification labor: $2,000–4,000 per aircraft
- Total first-year cost per aircraft: $5,500–10,000
- Recurring annual costs: $500–1,000

3.7.2 Safety Benefits:

General aviation experiences approximately 1 fatal accident per 100,000 flight hours. The economic cost of a fatal aviation accident exceeds $10 million (considering aircraft value, emergency response costs, legal settlements, and social costs). If the anomaly detection system prevents even 1 fatal accident per 2,000 aircraft over 10 years of operation, the expected value of accident prevention is:

(1 accident prevented/2,000 aircraft)×$10 million = $5,000 per aircraft over system lifetime

Over 10-year operational life, this alone justifies $500 per year safety investment, exceeding recurring software costs.

3.7.2.1 Operational Benefits:

Beyond accident prevention, anomaly detection provides operational efficiency improvements:

- Reduced aircraft damage from averted emergency situations: $\sim \$200 - -500$ per aircraft annually
- Reduced insurance premiums through demonstrated safety improvements: \$500–1,500 per aircraft annually
- Reduced pilot training requirements through automated alerting: \$200–400 per aircraft annually
- Improved operational availability through early anomaly detection enabling maintenance before failures: \$300–600 per aircraft annually

Total annual operational benefits: \$1,200–3,000 per aircraft

3.7.2.2 Return on Investment (ROI):

With total annual benefits of \$1,700–4,000 and annual costs of \$500–1,000, annual net benefit per aircraft is \$1,200–3,500. For a \$5,500–10,000 initial investment, payback period is 1.6–8.3 years depending on benefit realization and actual costs. Conservatively, considering only documented benefits and conservative cost estimates, 3–5 year payback periods appear achievable.

3.7.3 Market Size and Deployment Scenarios

The potential market for aviation anomaly detection in general aviation comprises:

- Active general aviation fleet: ∼180,000 aircraft globally (US: ∼170,000; Europe: ∼8,000; other: ∼2,000)
- Retrofit candidates (10+ year old aircraft with glass cockpit capacity): ∼80,000 aircraft
- New production aircraft (compatible models): ∼5,000–7,000 annually

Conservative deployment scenarios project:

- **5-Year Market Penetration:** 10–15% of retrofit candidates = 8,000–12,000 aircraft
- **Cumulative Revenue at \$6,500 average selling price:** \$52–78 million
- **Supporting Services and Training Revenue:** \$15–25 million
- **Total 5-Year Market Value:** \$67–103 million

3.7.4 Competitive Positioning and Differentiation

The anomaly detection system differentiates from existing general aviation safety systems through:

1. **Cost:** $6,500 total cost vs. $15,000–30,000 for commercial equivalents
2. **Specifically Designed for General Aviation:** Rather than adapted from commercial aviation, the system is purpose-built for general aviation sensor configurations and operational patterns
3. **Real-time Processing:** Unlike ground-based analysis systems, provides immediate pilot alerting
4. **Deep Learning Performance:** Superior performance compared to rule-based detection systems

Competitive positioning suggests strong market differentiation and pricing power, supporting business viability projections.

3.7.5 Long-Term Economic Considerations

Beyond initial deployment, long-term economic considerations include:

- **Fleet Modernization:** As older aircraft retire and newer aircraft (with integrated anomaly detection) enter service, retrofit market gradually shifts toward factory-installed options with higher profit margins
- **Regulatory Integration:** As anomaly detection becomes standard, potential regulatory credit (insurance premium reductions, maintenance interval extensions) enhances economic value
- **Technology Evolution:** Future systems may integrate additional sensors (weather, traffic) or advanced decision capabilities, creating upgrade pathways and recurring revenue streams
- **International Expansion:** Current analysis focuses on developed aviation markets; expansion to developing aviation markets could substantially increase long-term opportunity

3.7.6 Funding and Financial Requirements

Project financial requirements:

- Development budget: $850K–1.2M (16 months)
- Manufacturing setup: $500K (for pilot production)
- Initial marketing and sales: $300K (year 1)
- Total pre-revenue funding requirement: $\sim \$1.65 - 2.0\text{M}$

Projected funding timeline:

- Initial seed funding: $500K (months 1–4 model development)
- Series A funding: $1.2M (months 5–12 integration and certification)
- Revenue realization: Month 18+ (pilot testing complete, market launch)

Break-even analysis suggests positive cash flow achievable within 24–36 months assuming deployment targets, supporting venture capital viability.

Chapter 4

Methodology

4.1 Methodology Overview

This chapter presents the full methodology for developing a real-time aviation anomaly detection and multi-label classification system using deep learning on multivariate flight sensor data. The chapter is structured in the following way. Section 4.2 describes the model design, such as the architectures of the three purpose-built deep learning models and how each was trained. Section 4.3 explains the dataset, including its origin, pre-processing, the nature of its imbalance and the data splitting and windowing methodologies that were used to prepare the data to be used in the model during training. Section 4.4 shows the real time application of the entire two stage pipeline and its applicability to operational aircrafts.

The dataset used in this study is the General Aviation Training Set (GATS), which is extracted from the National General Aviation Flight Information Database (NGAFID). Only the Cessna 172S subset of GATS is used in this study, where it contains 4,481 flights and more than 22 million rows of multivariate sensor data that were recorded at a frequency of 1 Hz. For feature selection, nine sensors were taken, including AltAGL, AltMSL Lag Diff, IAS, NormAc, Pitch, Roll, Stall Index, Total Fuel, and VSpd. The dataset has a huge data imbalance problem. Normal flight conditions are found in approximately 99.83% of the total number of rows of timesteps, and abnormal conditions are found in only 0.17% of the total number of timesteps, with a ratio of about 1:603. There are 18 types of anomalies, and these are unevenly distributed. High Altitude Stall, the most common event, has over half of all occurrences of anomaly, whereas the least common events occur less than five times in the whole dataset. Two sensor features have missing values which are addressed with a two-step imputation strategy. To begin with, linear interpolation is used to fill the temporal gaps between rows. Then, median imputation is used in that cases where an entire sensor column is missing. The data is separated at the flight level, i.e. each row of a flight belongs to just one split. In stage 1 an event-stratified partitioning was used to ensure proportional representation of events in training, validation and test sets. To deal with the multi-label nature of the data and the co-occurrence of events, a greedy optimization algorithm is used in Stage 2. Both phases are based on training, validation, and testing in 80/10/10 ratio. For Stage 1, temporal windows of 30 consecutive timesteps have been created using a sliding window approach with stride = 1. This is important as anomaly de-

tection relies on comparing sensor values through time, but not individual timesteps.

The overall problem is divided into two sequential stages instead of solving it with a single end-to-end model, and this is done for both technical and practical reasons. Stage 1 deals with a binary detection problem where it is provided with a sensor window of length 30 timesteps, and the model decides whether there is an anomaly or not. Stage 2 handles the multi-label classification task where after detecting an abnormal timestep, the model predicts which 10 most common types of events are happening simultaneously. Because of this two stage approach in detecting the anomaly, each step has become more optimized and has done its work properly. Stage 1 focuses on learning temporal patterns under extreme binary imbalance, and in Stage 2 it focuses on distinguishing multiple events at the same time under per-class imbalance. This design helps to implement a trigger-based system, where Stage 2 will be turned on only in case Stage 1 detects an anomaly. As a result, the system is in idle condition approximately 99.7% of normal flight time.

Stage 1 involves a combination of two independently trained deep learning systems: the Cross-Sensor Patch Transformer (CSPT) and the Hierarchical Cross-Sensor Transformer (HiCST). Both models take an input tensor of size $B \times 9 \times 30$ and output an anomaly probability. They share a common initial design component, which is the Cross-Sensor attention module. This module represents the interrelationship among the nine sensor channels at a particular timestep, before any temporal processing occurs. This is important because many anomalies occur due to combined changes across multiple sensors, not just a single sensor. After this they follow different architectures. CSPT uses a patch-based tokenization approach, where consecutive timesteps are grouped into overlapping patches (patch_len = 6, stride = 3), resulting in 9 tokens. The tokens are then forwarded to three Pre-LayerNorm Transformer blocks, and a CLS token is used to make the final classification. On the other hand, HiCST keeps one token per timestep (total 30 tokens) and applies a hierarchical encoding strategy. Following both the initial two Transformer layers, a distillation convolution block (`Conv1d` $\rightarrow$ `BatchNorm` $\rightarrow$ `ELU` $\rightarrow$ `MaxPool`) decreases the sequence length by 30 timesteps to 15 and then to 8 timesteps. This enables the model to acquire higher-level temporal features in a progressive manner. This difference in architecture is the main reason for using an ensemble. CSPT model captures local temporal patterns with patch-based features whereas HiCST model captures more global patterns with hierarchical compression. Because they learn different types of features, their errors are also different, and combining them helps improve overall performance. In this study Root Mean Square (RMS) fusion have been used to combine the outputs of the two models, assigning a bit more weight to the more confident model. The final ensemble probability is then compared against a threshold τ_{opt}, which is chosen using validation data, to give the final binary anomaly decision. The complete Stage 1 data pipeline is shown in Figure 4.1.

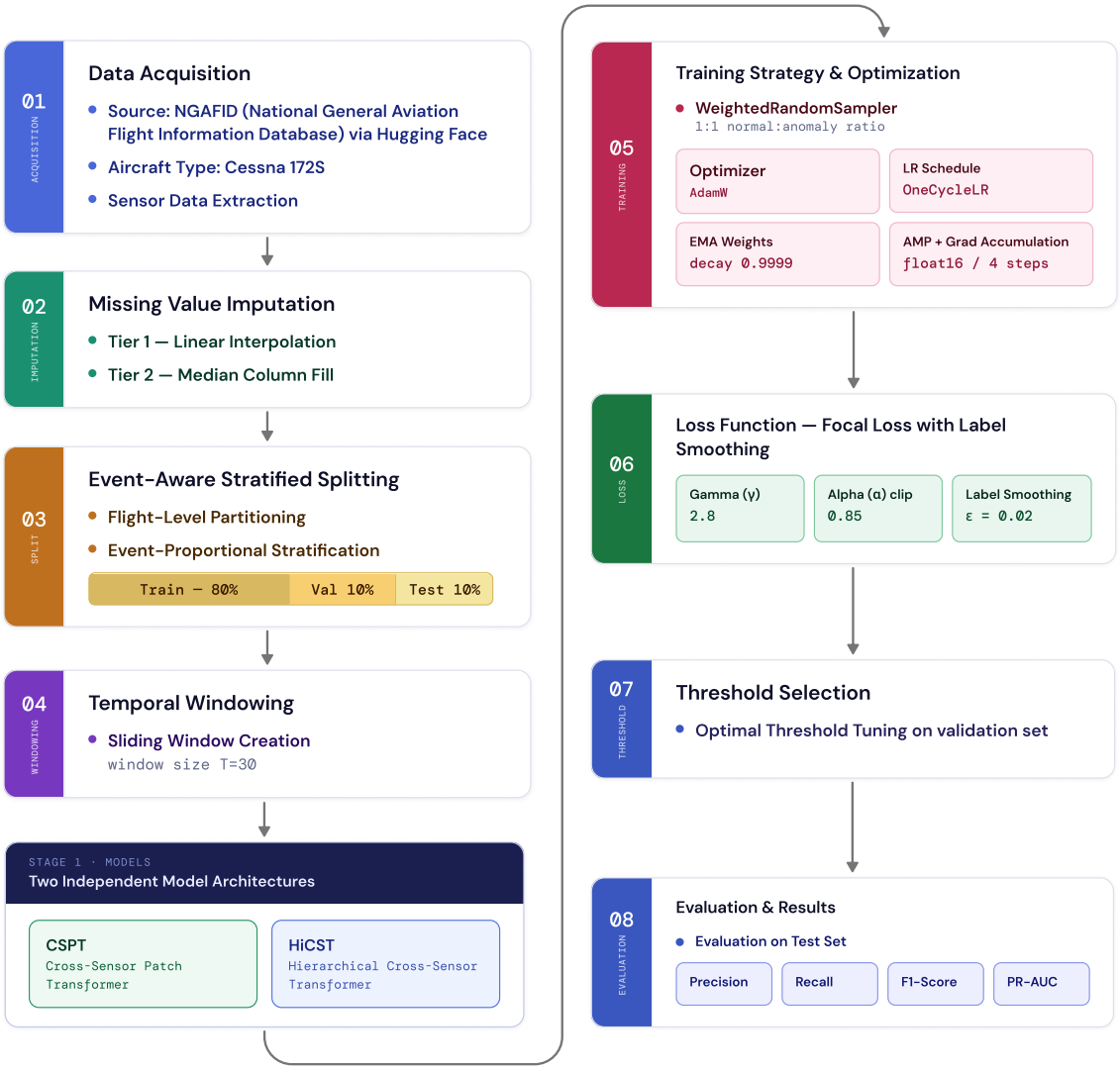

Figure 4.1: End-to-end Stage 1 methodology pipeline

Stage 2 uses a single model called MHANet (Multi-Head Attention Network), which is specifically designed for the multi-label event classification task. MHANet operates over individual timesteps rather than temporal windows which is opposite of Stage 1 models. It requires an input of size B $\times$ 9, which is a set of sensor values at one time. This design is suitable because identifying the type of anomaly depends on the current sensor state for example, whether the pitch angle is too high, fuel level is too low, or the stall index is elevated rather than the sequence of changes that happened before. The main architectural idea in MHANet is its per-sensor independent embedding. The model employs nine distinct linear projection layers, one per sensor, instead of a single shared linear layer with all sensors. This enables every sensor to discover its own representation according to its physical meaning, without being influenced by other sensors with different value ranges or meanings. In order to transform these probabilities into the final predictions, each class has its own thresholds. These thresholds are optimized on the validation set separately for each of the 10 event types. The end result is a multi-hot vector, with each value corresponding to the presence or absence of a particular anomaly. Figure 4.2 shows the entire Stage 2 data pipeline.

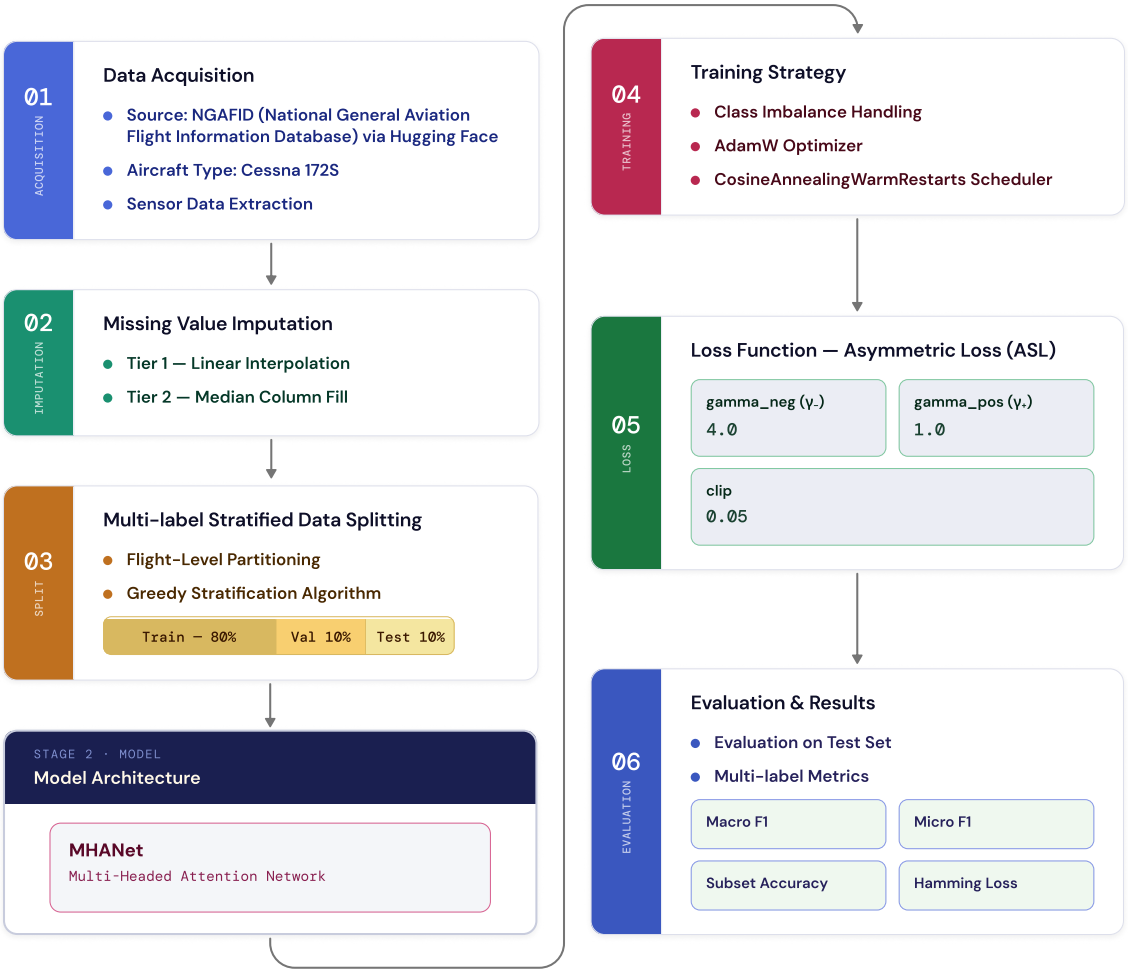

Figure 4.2: End-to-end Stage 2 methodology pipeline

The main goal in designing a two-stage pipeline was for real time aircraft deployment. Sensor data is received in 1 Hz in an operational environment and is

stored in a circular buffer which contains the latest 30 readings. This buffer is sent to the Stage 1 ensemble model every second. When the ensemble probability is above the predetermined threshold, a cockpit alert is activated, and the nine sensor values of the current timestep are immediately transmitted to MHANet to be classified as an event. The predicted event types are then displayed to the pilot and documented in the flight data recorder as well as the start timestamp, event type name, and duration. Because of its special design properties, this system is suitable for embedded avionics hardware. Both Stage 1 models have an internal SensorNorm layer, enabling raw sensor inputs to be fed into it without any external preprocessing. The models themselves are also lightweight and have a small embedding size (d model = 64) and small Transformer architectures, so inference can be run in milliseconds on a CPU, which is much less than the one-second time limit. Moreover, all the normalization parameters and decision thresholds are stored as part of the model, so no additional computation or external connection is needed at runtime. Also, because of having a trigger-based activation in stage 2, it enhances efficiency. Stage 2 is not engaged until Stage 1 has detected an anomaly, the classification model is not active during normal flight, minimizing the computation and power consumption. All these design characteristics enable the system to be self-contained, low-latency, and suitable when resources are limited in avionics applications.

4.2 Preliminary Design and Model Specification

4.2.1 Model Design

The pipeline of anomaly detection and classification that has been used in this paper is organized into two sequential steps. Stage 1 is works with binary detection task, in which the model decides whether it is an abnormal event or normal flight operation in a specific multivariate sensor window. Stage 2 works only on the windows that have been detected as anomalous by Stage 1 and classifies the specific event types that exist. Both stages work independently and Stage 2 is only provided with the data that has already passed the detection threshold of Stage 1. Stage 1 uses an ensemble of two models, where their predictions are combined to improve the robustness of anomaly detection. The first one in this set is the Cross-Sensor Patch Transformer (CSPT) and its architecture is detailed in this section. Figure 4.1 represents the entire pipeline, and Figure 4.3 depicts the architecture in the CSPT model in detail.

4.2.1.1 Stage 1: Model 1 CSPT (Cross-Sensor Patch Transformer)

The Cross-Sensor Patch Transformer (CSPT) is a custom deep learning model designed for anomaly detection in multivariate flight sensor data. It accepts fixed-length sliding windows of sensor readings as an input and generates a binary output per window, whether it is normal or anomalous. This architecture is constructed based on three key concepts. First, explicit cross-sensor relationship modelling before any temporal processing. Second, patch-based tokenization is

used such that every Transformer token has local temporal context; and third, a CLS token is used to encode global sequence information to be used in the final classification.

The model takes an input tensor of shape B $\times$ 9 $\times$ 30, where B is the batch size, 9 represents the number of sensor channels (AltAGL, AltMSL Lag Diff, IAS, NormAc, Pitch, Roll, Stall Index, Total Fuel, and VSpd), and 30 is the number of timesteps in the window. The architecture consists of six key stages these are SensorNorm, CrossSensorAttention, PatchEmbedding, a stack of three TransformerBlocks, CLS token extraction, and a final Classification Head.

Sensor Normalization:
The Sensor Normalization is used in the first stage. The raw input is then processed by a SensorNorm layer, which independently normalizes each channel of the nine sensor channels to a z-score using statistics calculated on the training set:

$$\hat{x}_c = \frac{x_c - \mu_c}{\sigma_c} \tag{4.1}$$

Here $\hat{x}_c$ is the normalized value of sensor channel c, and μ_c and σ_c are the mean and the standard deviation of the channel using the training data. Once normalized, the output tensor will be of the same shape, $B \times 9 \times 30$.

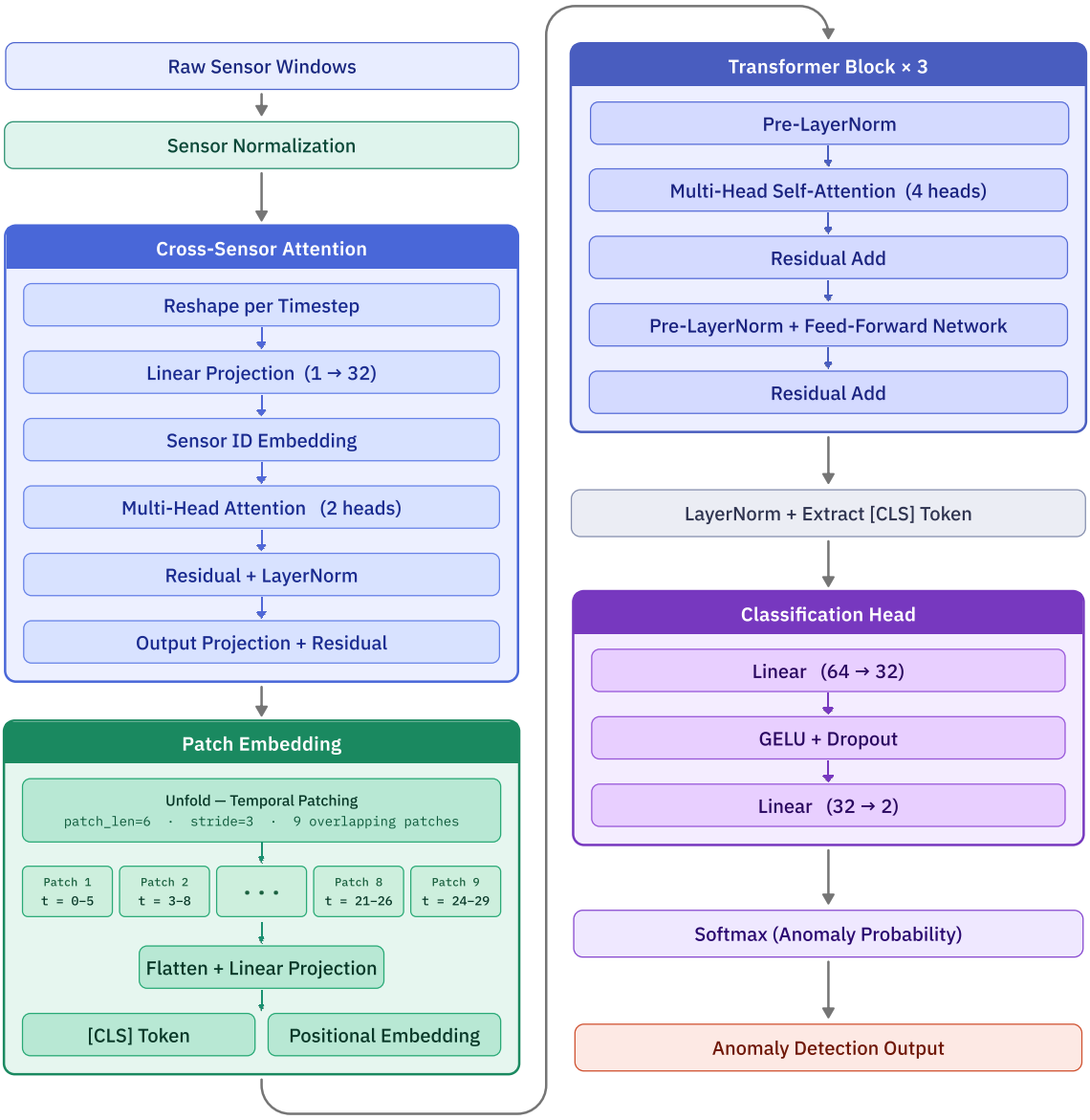

Figure 4.3: Architecture of the proposed CSPT (Cross-Sensor Patch Transformer)

This normalization step is necessary because the nine sensors operate on fundamentally different physical scales. AltAGL is measured in thousands of feet, NormAc spans roughly ±2g, and Stall Index is bounded between 0 and 1. Without normalization, sensors with large absolute magnitudes dominate the gradient signal during training, causing sensors with smaller scales to be effectively ignored by the model. The mean and standard deviation are stored as fixed non-learnable buffers rather than as trainable parameters.

Cross-Sensor Attention:
Cross-Sensor Attention is the most structurally novel component of the CSPT architecture. Its purpose is to explicitly model the relationships between different sensors at each timestep before any temporal processing takes place. The most important reason is that anomalies cannot be identified with only one sensor. As an illustration, a Low Altitude Stall needs to monitor several conditions simultaneously AltAGL should be low, IAS should be below a critical level, and the Stall Index should be high. No individual sensor can represent this condition alone; the relationship between sensors carries the actual diagnostic signal.

The CrossSensorAttention layer is independent at each timestep. The B × 9 × 30-shaped input tensor is reshaped by combining the time dimension with the batch dimension to create a (B×30) × 9 × 1-shaped tensor. The B×30 samples now reflect the nine sensor values at a particular timestep. This transformation isolates the inter-sensor (spatial) dimension, allowing the attention mechanism to focus only on how sensors relate to each other at that moment. The single sensor values are then projected to a higher-dimensional space with a learned linear transformation between dimension 1 and dimension 32:

$$h_i = W_{\text{proj}} \cdot x_i, \qquad h_i \in \mathbb{R}^{32} \tag{4.2}$$

A single number carries too little information and cannot provide sufficient information for a meaningful attention computation. By projecting it into a 32-dimensional space, the model can represent inter-sensor relationships as geometric relationships between vectors. However, a problem arises after this projection. When the normalized value of two sensors is equal, the projected vectors of the sensors will be the same and the model will not be able to differentiate between them. To solve this, a learnable sensor identity embedding is added to each projected vector:

$$e_i = h_i + \text{emb}_i, \qquad \text{emb}_i \in \mathbb{R}^{32} \tag{4.3}$$

The nine sensors have their own learnable embedding vectors of size 32. The sensor index is a categorical variable, meaning the values 0 through 8 do not have any ordinal relationship. Due to this, a learnable embedding is employed to map every sensor identity to a meaningful vector representation. The resulting representations are then sent through a two-head Multi-Head Self-Attention (MHSA) layer and then the projection sensor values are added to their identity embeddings. Here, each sensor listens to all the other sensors to get to know about inter-sensor relationships:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.4}$$

Here, Q, K, and V are all derived from the same combined embedding, meaning this is self-attention. The scaling factor (sqrt d k), with (d k = 16) per head, is to ensure that the dot-product values do not get too large, which would cause the softmax function to enter a region where gradients are very small. Instead of four attention heads, two are applied since the sequence length is 9 (number of sensors). In case of four heads, the heads will be working on an 8-dimensional subspace, which is too small to learn meaningful relationships. It is more appropriate with two heads, each head has a 16-dimensional space. This enables various heads to specialize in various kinds of relations, such as one head might study altitude-airspeed relations, and another might study pitch-roll relations. The attention layer output is then subjected to Layer Normalization and a residual connection. The residual connection makes sure that in case the attention mechanism fails to contribute any useful information, the original sensor representation can still flow through without alteration. The reason behind the use of LayerNorm over BatchNorm is that the flight data of various stages, including takeoff, cruise, and landing, have highly different statistical characteristics, and LayerNorm normalizes each sample separately.

Finally, the resulting 9×32 representation for each timestep is projected back into a 9-dimensional sensor space using a linear layer. This is then fed back to the original input by another residual connection. This implies that the CrossSensorAttention module does not substitute the input, but rather introduces a correction term, which encodes inter-sensor relationships. The ultimate output shape is B $\times$ 9 $\times$ 30, which is identical to that of the input.

Patch Embedding:
After CrossSensorAttention layer enriches each timestep with inter-sensor context, the next stage converts the time-series into a sequence of tokens that can be processed by a Transformer. With timesteps considered as individual tokens, then the token would not have a temporal context, i.e. it would not know what preceded or followed that moment. To address this, CSPT relies on patch-based tokenization, in which several successive time steps are clustered into a single token. This enables every token to be a local temporal trajectory rather than a single point in time. Two hyperparameters are used to control the patching process: patch_len = 6 and stride = 3. Each patch contains 6 consecutive timesteps, and a new patch starts every 3 timesteps. The total number of patches extracted from a 30-timestep window is given by:

$$N_{\text{patches}} = \left\lfloor \frac{T - \text{patch\_len}}{\text{stride}} \right\rfloor + 1 = \left\lfloor \frac{30 - 6}{3} \right\rfloor + 1 = 9 \tag{4.5}$$

Since the stride (3) is smaller than the patch length (6), the patches overlap. For example, Patch 1 covers timesteps 0–5, Patch 2 covers 3–8, Patch 3 covers 6–11, and so on. If an anomaly occurs near the boundary between two patches, it would be captured only partially without overlap potentially being too weak a signal in either patch to be detected. With overlapping patches, any meaningful anomaly is fully captured within at least one patch. The patches have 9 sensors in 6 timesteps, which make a 9 x 6 block. This block is flattened into a 54-dimensional vector and projected into a 64-dimensional space with a learned linear transformation:

$$\text{token}_j = W_{\text{embed}} \cdot \text{flatten}(\text{patch}_j), \qquad \text{token}_j \in \mathbb{R}^{64} \tag{4.6}$$

This linear projection acts like a patch encoder, transforming raw multivariate time-data to a dense 64-dimensional representation that can be processed by Transformer.

A special learnable CLS (classification) token of dimension 64 is then added at the beginning of the token sequence, increasing the total sequence length from 9 to 10. This CLS token processes each patch token during Transformer processing and collects significant information in the entire sequence. At the end of the Transformer, only the CLS token is used to make the final classification decision. This is better than average pooling since pooling considers all patches as equal when anomalies are typically concentrated in a few patches. The CLS token, through attention, can focus more on the important patches and ignore less relevant ones. To preserve positional information, a learnable positional embedding of dimension 10×64 is added to the token sequence before it is passed into the Transformer. Without this, the self-attention mechanism is permutation-invariant and cannot distinguish Patch 1 (early in the window) from Patch 9 (late in the window). As the positional embedding size is fixed because the length of the sequence is 10 tokens. So, a Dropout layer with probability p = 0.20 is applied after the positional embeddings and the sequence is then fed into the Transformer stack.

Transformer Block:
The token sequence B $\times$ 10 $\times$ 64 is fed through three stacked Transformer Blocks. Each block has the same structure but uses its own set of parameters, allowing the model to learn different levels of representation. The first block mainly captures local features at the patch level, the second learns relationships between patches (temporal patterns), and the third builds a global understanding of the entire 30-timestep window. In all three blocks, the shape of the tensors does not change, and it remains as B $\times$ 10 $\times$ 64.TransformerBlocks have two primary sub-layers: Multi-Head Self-Attention (MHSA) and a Feed-Forward Network (FFN). Both sub-layers have a Pre-LayerNorm design. In the original Transformer architecture, Post-LayerNorm was used, where normalization is applied after the residual connection. However, this approach can lead to gradient instability, especially in deeper networks. The reason is that gradients in later layers are required to pass through the normalization step prior to reaching earlier layers, which can diminish them. Pre-LayerNorm keeps the residual path free of normalization allowing gradients to flow more directly and producing more stable convergence.

Multi-Head Self-Attention:
The token sequence is first passed through a LayerNorm operation. Normalized output is then projected linearly into Query (Q), Key (K) and Value (V) matrices all with shape B x 10 x 64. These matrices are then divided into 4 attention heads, each head working in a 16-dimensional subspace. The attention computation for each head is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{16}}\right)V \tag{4.7}$$

Each of the four attention heads learns to focus on different types of inter-patch

relationships. An example is that one head can learn short-term patch-patch dependencies, and another can learn long-term temporal sequence-sequence dependencies. After computing attention separately in each head, the outputs are concatenated and projected back into a 64-dimensional space. Softmax is used to compute attention weights because the temporal information is distributed across all patches. Each patch contributes some information to the overall picture, and a hard sparse attention mechanism would discard the soft temporal dependencies which are important for detecting gradually developing anomalies. Lastly, the residual connection is used to add the attention output to the original pre-normalized input.

Feed-Forward Network.
The output of the attention sub-layer is first normalized again using LayerNorm, and then passed through a two-layer Feed-Forward Network (FFN):

$$\mathrm{FFN}(x) = W_2 \cdot \mathrm{GELU}(W_1 \cdot x + b_1) + b_2 \tag{4.8}$$

The first linear layer in this structure increases the feature dimension by $d_{\text{ff}} = 256$ which is the standard expansion ratio of 4 in Transformer models. This 256 dimensional representation is then projected by the second layer of linear back to 64 dimensions. The extension into a new higher-dimensional space enables the network to learn more complex and richer non-linear transformations of the attention output. The activation function employed in this case is GELU (Gaussian Error Linear Unit) rather than ReLU. ReLU forces all negative values to zero, which can lead to dead neurons and loss of useful information. GELU applies a smooth, probabilistic gate that preserves a small gradient even for negative inputs. This is important in the case of flight sensor data, where negative values can be physically meaningful, e.g. a negative pitch angle or descent rate. So this is impractical to totally eliminate these values because it would lower the capability of the model to identify anomalies. To improve generalization and prevent overfitting, Dropout with probability (p = 0.20) is applied after the GELU activation and again after the second linear layer. And then the result of the FFN is fed back to the input of the FFN via a residual connection, so that the model is able to remember the original information but apply the learnt transformations.

Final LayerNorm and CLS Token Extraction:
After the three TransformerBlocks, a final LayerNorm is applied to the entire B $\times$ 10 $\times$ 64 sequence to stabilize the representations before classification. The CLS token (index 0 of the sequence) is then extracted, producing a tensor of shape B $\times$ 64. The other nine patch tokens are discarded. The CLS token is used as the only input to the classification head because, throughout the three Transformer blocks, it has attended to all patch tokens and gathered the most important information from them. As a result, this single 64-dimensional vector acts as a compact and globally informed representation of the entire 30-timestep multivariate sensor window, containing the key features needed for anomaly detection.

Classification Head:
The 64-dimensional CLS vector is passed through a two-layer classification head. The initial linear layer will decrease the number of dimensions to 32. This is followed by a GELU activation and a Dropout layer with probability (p = 0.20). The

32-dimensional representation is then mapped by the second linear layer to 2 output dimensions to generate two raw logits, one of which is the normal class and the other the anomaly class. It is better to use two logits with a Focal Loss objective instead of a single sigmoid output. This configuration enables the model to learn directly the relative confidence between the two classes, which performs better in a highly imbalanced dataset than typical binary cross-entropy with a single output. Both linear layers in the classification head are initialized using Xavier normal initialization, defined as:

$$\mathrm{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}} \tag{4.9}$$

This ensures that the variance of activations is approximately preserved as signals pass forward and backward through the layer. Initialization can be unstable at this point, causing unstable gradients at the start of training, before the Transformer backbone has acquired meaningful representations. During inference, the two logits are passed through a Softmax function to produce class probabilities. A window is considered anomalous when the probability of the anomaly class is greater than a threshold τ_{opt}, which is determined by maximizing F1 score on the validation set.

4.2.1.2 Stage 1 Model 2 HiCST (Hierarchical Cross-Sensor Transformer)

The second model in the Stage 1 ensemble is the Hierarchical Cross-Sensor Transformer (HiCST). While CSPT uses patch-based tokenization followed by a flat stack of Transformer blocks, HiCST follows a different architectural approach. It treats each individual timestep as a separate token and gradually compresses the temporal sequence across layers using learned distillation convolutions. This creates a hierarchical encoder that processes the data at multiple levels of temporal abstraction starting from raw timestep relationships and moving toward a compact global representation before making the final classification decision.

Like CSPT, HiCST accepts an input of shape B $\times$ 9 $\times$ 30, where B is the batch size, 9 is the number of sensor channels and 30 is the number of timesteps in the window. The architecture can be divided into seven consecutive stages: SensorNorm, CrossSensorMHA, Temporal Projection, CLS token with Positional Embedding, a Hierarchical Transformer Encoder with two Distillation Convolutions, a final LayerNorm with CLS extraction, and a Classification Head. Figure 4.1 depicts the entire pipeline and Figure 4.4 depicts the architecture of the HiCST model.

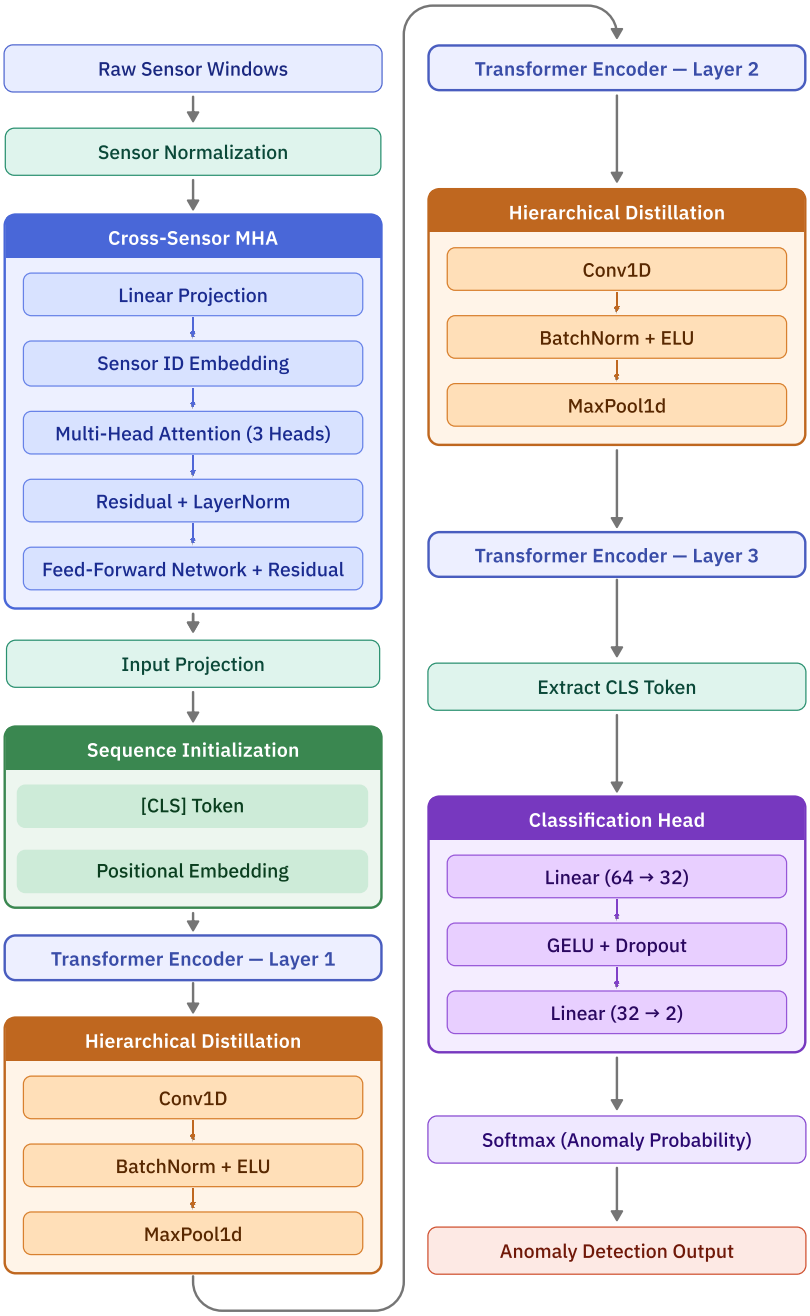

Figure 4.4: Architecture of the proposed HiCST (Hierarchical Cross-Sensor Transformer)

SensorNorm:
The SensorNorm layer in HiCST is identical in both design and function to the one used in CSPT. It applies channel-wise z-score normalization independently to each of the nine sensor channels, using statistics computed from the training dataset and stored as fixed, non-learnable parameters:

$$x_c = \frac{x_c - \mu_c}{\sigma_c} \tag{4.10}$$

The reasoning behind this is the same as in CSPT. The mean (μ_c) and standard deviation (σ_c) are calculated only from the training data and then kept constant during training and inference to avoid data leakage. After normalization, the tensor shape remains unchanged at $B \times 9 \times 30$.

CrossSensorMHSA:
The second stage, CrossSensorMHA, serves the same overall purpose as the CrossSensorAttention module in CSPT: it models the relationships between different sensors at each individual timestep before any temporal processing is applied. This is significant since most of the types of anomaly can be detected only under multiple sensor conditions and not by individual sensors. However, CrossSensorMHA is a more advanced and expressive component compared to its CSPT counterpart. Whereas CSPT uses attention first and a simple projection, CrossSensorMHA uses a full Transformer encoder layer on sensor dimension, employing attention with a separate Feed-Forward Network. This allows it to capture more complex, non-linear inter-sensor relationships. Similar to CSPT, the input (B $\times$ 9 $\times$ 30) is reshaped by combining the time dimension with the batch dimension to create a (B$\times$30) $\times$ 9 $\times$ 1 tensor. This guarantees that every timestep is computed separately, and the attention mechanism is only concerned with the relationships between sensors at that time, without introducing temporal dependencies. Each scalar sensor value is then projected into a higher-dimensional space using a learned linear transformation:

$$h_i = W_{\text{proj}} \cdot x_i, \qquad h_i \in \mathbb{R}^{48} \tag{4.11}$$

The projection dimension here is 48, which enables the model to more accurately capture the intricate inter-sensor interactions. Also, to distinguish sensors with identical normalized values a learnable 48-dimensional identity embedding is added element-wise to each projected sensor vector:

$$e_i = h_i + \text{emb}_i, \qquad \text{emb}_i \in \mathbb{R}^{48} \tag{4.12}$$

Each of the nine sensors has a dedicated embedding in an nn.Embedding$(9, 48)$ table. Three attention heads are used for the cross-sensor self-attention. With a projection dimension of 48 and three heads, each head operates in a 16-dimensional subspace. The self-attention computation follows the standard scaled dot-product form:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{16}}\right)V \tag{4.13}$$

The attention output is followed by a residual connection and LayerNorm (the first of two in this block). Dropout ($p = 0.10$) is applied to the attention output before

the residual addition.

Feed-Forward Network.
After the attention sub-layer, a two-layer FFN processes each sensor's representation independently:

$$\mathrm{FFN}(e) = W_2 \cdot \mathrm{GELU}(W_1 \cdot e + b_1) + b_2 \tag{4.14}$$

The first linear layer expands from 48 to 96 dimensions (a 2× expansion), and the second projects back to 48 dimensions. GELU is used as the activation function for the same reason as in CSPT. A second residual connection and LayerNorm follow the FFN output. This FFN is the key structural addition over CSPT's cross-sensor module. After the attention and FFN sub-layers, the (B×30) × 9 × 48 representation is flattened to (B×30) × 432 and projected back to 9 dimensions via a linear layer. Dropout (p = 0.10) is applied before this flattening. The 9-dimensional output is reshaped and permuted back to B × 9 × 30, then added to the SensorNorm output as a residual correction:

$$\text{output} = x_{\text{normed}} + \Delta_{\text{cross-sensor}} \tag{4.15}$$

CrossSensorMHA does not replace the input. It adds a learned correction term encoding cross-sensor context. If the cross-sensor information is not useful for a particular window, the model can push Δ toward zero and rely on the original sensor values. The output shape is B × 9 × 30.

Temporal Projection and Token Construction:
After CrossSensorMHA, the tensor of shape B × 9 × 30 must be converted into a token sequence for the Transformer. This is where HiCST makes its most fundamental departure from CSPT. HiCST treats each individual time step as a separate token. Every token therefore represents a single moment in time, but contains the enriched cross-sensor representations from all nine sensors combined. The tensor is first permuted from sensor-first (B × 9 × 30) to time-first layout (B × 30 × 9), where each row corresponds to a single time step and its nine sensor values form the feature vector. A linear projection then maps the 9-dimensional sensor vector at each time step to a 64-dimensional Transformer embedding:

$$\text{token}_t = W_{\text{input}} \cdot x_t, \qquad \text{token}_t \in \mathbb{R}^{64} \tag{4.16}$$

The input is only 9-dimensional because each token covers a single time step rather than a multi-step patch. The projection lifts each timestep's sensor readings into a richer 64-dimensional space where the Transformer can operate. The output shape is $B \times 30 \times 64$, 30 tokens, one per time step, each carrying a 64-dimensional representation of that moment's cross-sensor-enriched sensor state.

CLS Token and Positional Embedding:
A learnable CLS token of dimension 64 is prepended to the 30 timestep tokens, bringing the sequence length to 31. It attends to all other tokens throughout the Transformer layers and accumulates a global summary of the entire window for use in the final classification step. HiCST initializes it using a truncated normal distribution with standard deviation 0.02:

$$\text{CLS} \sim \mathcal{TN}(0,\, 0.02) \tag{4.17}$$
$$\text{truncated at } \pm 2\sigma$$

Initializing from zero means the CLS token has no distinguishable relationship with any other token at the start of training, which can slow down the early stages of learning meaningful attention patterns. A small random initialization gives the CLS token an initial direction in the embedding space, helping it establish useful attention relationships from the first training steps. The truncation at $\pm 2\sigma$ ensures the initial values do not grow large enough to destabilize early training.

After prepending the CLS token, a learnable positional embedding of shape $1 \times 31 \times 64$ is added to the full sequence. Each of the 31 positions (one CLS and 30 timestep positions) has a dedicated 64-dimensional learnable vector initialized with small random values. Self-attention is permutation-invariant, so without positional embeddings the Transformer cannot distinguish early timesteps from late ones critical information for detecting anomalies that develop over time. Learnable positional embeddings are preferred over fixed sinusoidal encodings because the sequence length is always fixed at 31 tokens, and learnable embeddings can adapt to the specific positional structure of this task. The output shape is $\text{B} \times 31 \times 64$.

Hierarhical Transformer Encoder with Distillation:
This stage is the defining structural innovation of HiCST. HiCST uses a hierarchical encoder where the temporal sequence is progressively compressed after each of the first two Transformer layers by a distillation convolution. The motivation is grounded in the nature of flight data: not all 30 time steps in a window carry equal information. Anomaly signals are typically concentrated in a small subset of timesteps within the window. By compressing the sequence progressively, HiCST encourages the model to identify which temporal regions contain the most activated patterns, and to carry those forward in compact form for higher-level reasoning. The CLS token is maintained separately throughout this process and is never compressed at each layer.

Before the hierarchical layers begin, the CLS token and the 30 timestep tokens are separated. The CLS token (shape $\text{B} \times 1 \times 64$) is managed independently, while the timestep sequence (shape $\text{B} \times 30 \times 64$) passes through the distillation pipeline. Before each Transformer layer, the CLS token is prepended to the current timestep sequence; after each layer, it is separated again for the distillation step.

Layer 1 and Distillation 1: The CLS token is concatenated with the 30-timestep sequence to form a $\text{B} \times 31 \times 64$ input. This passes through the first TransformerEncoderLayer, which uses a Pre-LayerNorm design, four attention heads (each operating in a 16-dimensional subspace), a Feed-Forward Network with 256 hidden dimensions ($4\times$ expansion) and GELU activation, and Dropout ($\text{p} = 0.15$). Pre-LayerNorm is used for to keep the residual path free of normalization. After the layer, the CLS token and timestep sequence are separated again. The updated CLS token ($\text{B} \times 1 \times 64$) has now attended to all 30 raw timestep representations. The 30-timestep sequence then passes through the first Distillation Convolution,

which applies four sequential operations:

Conv1d (kernel=3, padding=1, bias=False): A one-dimensional convolution with kernel size 3 slides across the time axis. Each output position integrates information from its immediate temporal neighbors, providing local temporal smoothing that complements the global attention in the Transformer. Padding of 1 ensures that boundary timesteps are processed with the same receptive field as interior ones. The bias is omitted because the subsequent BatchNorm layer subsumes any additive offset.

BatchNorm1d: Batch normalization is applied channel-wise to the convolutional output. While LayerNorm is the natural choice for Transformer token representations, BatchNorm is the standard practice for convolutional feature maps, where the same channel produces comparable statistics across samples in a batch. This combination LayerNorm within the Transformer, BatchNorm within the convolution applies the most appropriate normalization strategy to each component.

ELU activation: Exponential Linear Unit activation is used here rather than GELU or ReLU. ReLU maps all negative inputs to exactly zero, discarding any signal. GELU smoothly attenuates negative inputs but still drives them toward zero. ELU allows negative inputs to saturate at a small negative value rather than zero, preserving a signal that indicates absence or suppression. In the distillation context, "this timestep shows no anomaly signal" is itself meaningful information, and ELU retains that negative signal rather than silencing it.

MaxPool1d (kernel=2, `ceil_mode=True`): The sequence is halved by taking the maximum value in each pair of adjacent timesteps. MaxPool preserves the most activated and therefore most anomaly-relevant timestep in each pair, rather than averaging them together as AvgPool would. `ceil_mode=True` ensures that if the sequence length is odd, the final timestep is included in a pool rather than dropped. After Distillation 1, the sequence has been compressed from 30 to 15 timesteps, giving an output shape of $B \times 15 \times 64$.

Layer 2 and Distillation 2: The CLS token is prepended to the 15-timestep sequence (B $\times$ 16 $\times$ 64) and passed through the second TransformerEncoderLayer with the same configuration. The CLS token now attends to 15 compressed tokens, each of which summarizes the most activated features from a pair of original timesteps. The second layer therefore learns relationships at a coarser temporal resolution than the first rather than individual timestep patterns, it captures mid-level dynamics such as how the first half of the window relates to the second half. After the layer, the 15-timestep sequence passes through the second Distillation Convolution with the same four operations. MaxPool with `ceil_mode=True` on 15 timesteps yields $\lceil 15/2 \rceil = 8$ timesteps. After Distillation 2, the sequence shape is B $\times$ 8 $\times$ 64.

Layer 3 (no distillation): The CLS token is prepended to the 8-timestep sequence (B $\times$ 9 $\times$ 64) and passed through the third and final TransformerEncoderLayer,

again with the same configuration. By this point, each of the 8 tokens represents a compressed summary of approximately 3–4 original timesteps, containing the most activated features from those timesteps. The CLS token attends to these 8 highly abstract temporal representations and produces its final global summary of the 30-step window. No distillation follows this layer. After the layer, the CLS token is extracted (B $\times$ 1 $\times$ 64) and the 8 timestep tokens are discarded.

Across the three layers, the hierarchical encoder builds representations at three levels of abstraction: Layer 1 captures local patterns across 30 raw timestep tokens, Layer 2 captures mid-level temporal relationships across 15 compressed tokens, and Layer 3 synthesizes a global window summary from 8 highly compressed tokens. The CLS token's representation is progressively refined at each level.

Final LayerNorm and CLS Token Extraction: After the third TransformerEncoderLayer, a final LayerNorm is applied to the extracted CLS token. The Pre-LayerNorm design used within each Transformer layer normalizes the input before attention and FFN, but leaves the residual output unnormalized. Before the CLS representation enters the classification head, this final LayerNorm stabilizes it to a consistent scale. The CLS token's size-1 sequence dimension is then removed, yielding a tensor of shape B $\times$ 64, one 64-dimensional summary vector per sample in the batch.

Classification Head: The 64-dimensional CLS vector passes through a two-layer classification head that is structurally identical to the CSPT classification head. The first linear layer maps from 64 to 32 dimensions, acting as a bottleneck that compresses the rich CLS representation into the most class-discriminative features. A GELU activation and Dropout (p = 0.15) follow. The second linear layer maps from 32 to 2 dimensions, producing two raw logit scores, one for the normal class and one for the anomaly class. Both linear layers in the classification head are initialized using Xavier normal initialization, ensuring a stable initial weight scale that neither amplifies nor attenuates signals passing through the head. This initialization is particularly important here because the classification head connects directly to the loss, and unstable gradients early in training can prevent the deeper Transformer layers from receiving any useful learning signal at all. The output shape is B $\times$ 2.

4.2.1.3 Stage 1 Ensemble: RMS Fusion of CSPT and HiCST

CSPT and HiCST are trained independently on the same training data but follow fundamentally different architectural strategies. CSPT captures temporal anomaly patterns through overlapping patch tokens, while HiCST builds progressively abstract representations through hierarchical compression. Because their architectures differ so substantially, their error patterns differ as well cases where one model is uncertain or incorrect tend to be cases where the other is more confident and correct. The purpose of ensembling the two models is to exploit these complementary strengths: by combining their outputs, the ensemble can correct for the individual blind spots of each model and produce more reliable anomaly

probability estimates than either model alone.

The ensemble pipeline is illustrated in Figure 4.5. Both models receive the same raw sensor window ($B \times 9 \times 30$) and independently produce an anomaly probability. These probabilities are fused using Root Mean Square (RMS) fusion, and a final binary decision is made by applying an optimal threshold learned on the validation set.

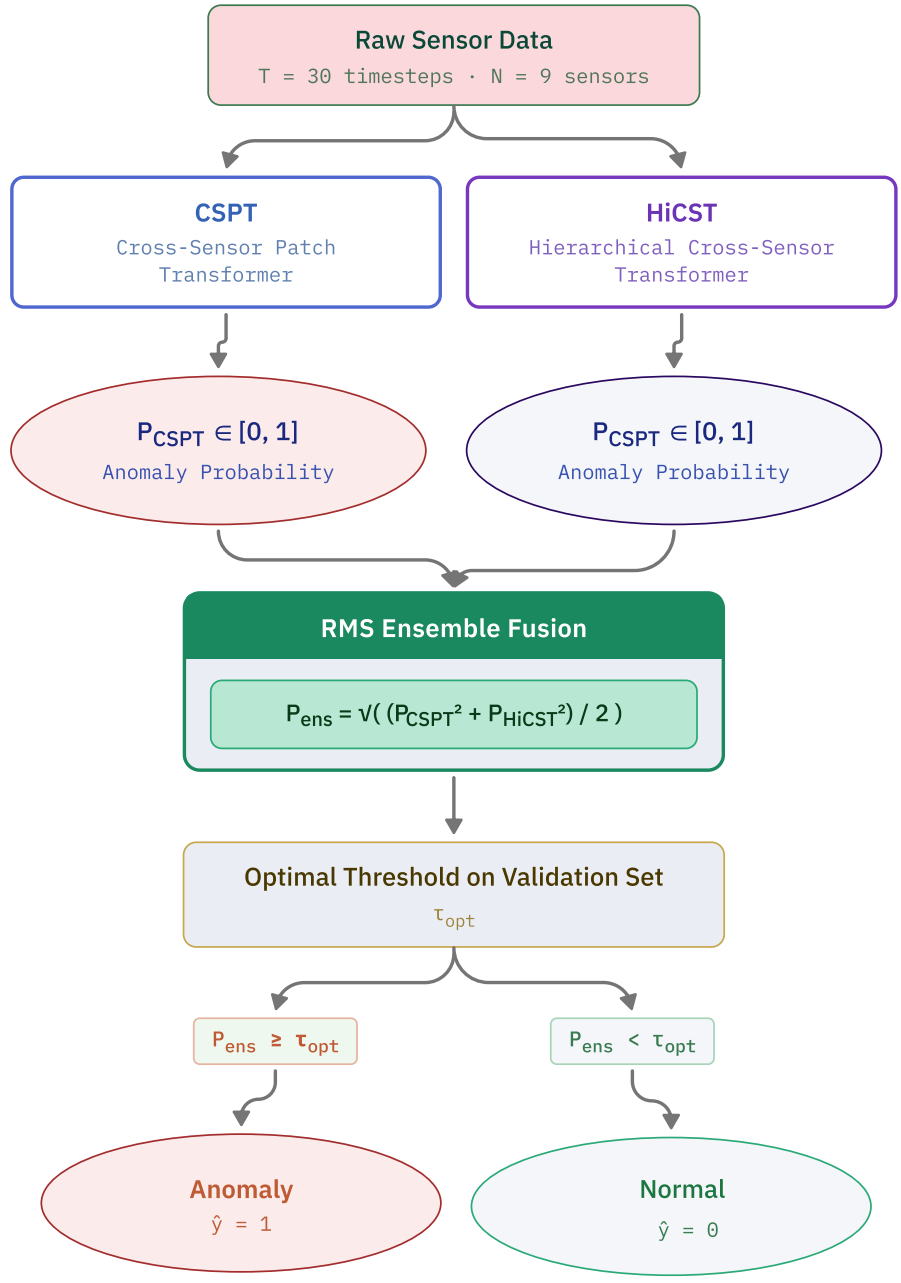

Figure 4.5: Stage 1 RMS ensemble pipeline

Independent Model Predictions:
After training, CSPT and HiCST are each evaluated independently on the test set. The softmax output of each model's classification head yields an anomaly probability for every input window:

$$p_{\text{CSPT}} \in [0,1]^N, \qquad p_{\text{HiCST}} \in [0,1]^N \tag{4.18}$$

where N is the number of test samples. Both probability arrays must have equal length, since both models are evaluated on the same test set. These two probability vectors are the sole inputs to the ensemble fusion step. Importantly, no retraining or joint optimization is performed; the ensemble combines the independently trained models at inference time only.

RMS Ensemble Fusion:
For each test sample, the anomaly probability from the two models is combined using Root Mean Square (RMS) fusion:

$$p_{\text{ens}} = \sqrt{\frac{p_{\text{CSPT}}^2 + p_{\text{HiCST}}^2}{2}} \tag{4.19}$$

This is an element-wise operation, producing one ensemble probability per sample. The RMS is a member of the generalized power mean family with exponent $p = 2$, which places it between the arithmetic mean ($p = 1$) and the maximum ($p \to \infty$). Its key property is that it gives slightly more weight to the more confident model. RMS is chosen for Stage 1 because anomaly detection benefits from amplifying strong individual signals. When one model is highly confident that a window is anomalous, that strong signal is more likely to reflect a genuine anomaly than a false positive and the RMS gives that signal slightly more influence than a simple average would. At the same time, when both models are in agreement, the RMS behaves similarly to the arithmetic mean.

Optimal Threshold on the Validation Set:
The RMS output p_{ens} is a continuous probability. To convert it into a binary prediction (Anomaly or Normal), a classification threshold is required. Rather than using a fixed threshold of 0.5, the optimal threshold τ_{opt} is determined by sweeping candidate threshold values on the validation set and selecting the one that maximizes the F1 score:

$$\tau_{\text{opt}} = \arg\max_t \text{F1}(y_{\text{val}}, p_{\text{val}}(t)) \tag{4.20}$$

A fixed threshold of 0.5 is inappropriate for this dataset. The class imbalance is extreme: approximately 99.8% of timestep windows are normal and only 0.2% are anomalous. Under this imbalance, the model's predicted probability distribution is skewed; anomalous windows may consistently receive probabilities below 0.5 because the model has seen relatively few anomalous examples during training.

Final Binary Decision:
The optimal threshold is applied to the ensemble probabilities to produce the final binary output:

$$\hat{y} = 1 \text{ (Anomaly)} \quad \text{if} \quad p_{\text{ens}} \geq T_{\text{opt}} \tag{4.21}$$

$$\hat{y} = 0 \text{ (Normal)} \quad \text{if} \quad p_{\text{ens}} < T_{\text{opt}} \tag{4.22}$$

A fixed threshold of 0.5 is inappropriate for this dataset. The class imbalance is extreme: approximately 99.8% of timestep windows are normal and only 0.2% are anomalous. Under this imbalance, the model's predicted probability distribution is skewed; anomalous windows may consistently receive probabilities below 0.5 because the model has seen relatively few anomalous examples during training.

Final Binary Decision. The optimal threshold is applied to the ensemble probabilities to produce the final binary output:

$$\hat{y} = 1 \text{ (Anomaly)} \quad \text{if} \quad p_{\text{ens}} \geq T_{\text{opt}} \tag{4.23}$$

$$\hat{y} = 0 \text{ (Normal)} \quad \text{if} \quad p_{\text{ens}} < T_{\text{opt}} \tag{4.24}$$

This binary prediction is the final output of Stage 1. Windows classified as anomalous ($\hat{y} = 1$) are passed to the Stage 2 classifier (MHANet) for event-type identification. Windows classified as normal ($\hat{y} = 0$) require no further processing. The two-stage design therefore ensures that the computationally lightweight Stage 2 model only processes the small subset of windows that have already been identified as anomalous, keeping the overall inference pipeline efficient.

4.2.1.4 Stage 2 Model MHANet (Multi-Head Attention Network)

Stage 1 models (CSPT and HiCST) operate on B × 9 × 30 tensors 30 timestep sliding windows because detecting whether a flight anomaly is occurring requires observing temporal patterns that unfold over several seconds. For example, develops gradually: airspeed falls, stall index rises, and altitude drops across multiple timesteps before the event fully manifests.

Stage 2 has a fundamentally different task. Once Stage 1 has flagged a timestep as anomalous, Stage 2 must identify which of the 10 event types are active at that specific moment. This classification is performed row by row: each timestep is classified independently. The input is therefore B × 9 a single timestep's nine sensor readings with no temporal window. The rationale is that the identity of an anomaly event depends primarily on the instantaneous configuration of sensor values: a High Altitude Stall is characterized by a high stall index at altitude, while a Roll event is characterized by excessive roll angle. These signatures are sensor-value conditions, not temporal trajectories, making per-timestep classification the appropriate formulation.

This single-timestep design is also more computationally efficient. Because no temporal window is constructed, the model processes each flagged row directly as a compact 9 dimensional input, making inference lightweight and suitable for point-in-time decisions. More importantly, the classification of aviation anomaly types is grounded in the instantaneous readings of flight sensors: the type of anomaly present at a given moment whether it is excessive roll, low fuel, high pitch, or a stall condition is determined by what the sensors are measuring right now, not by the trajectory of those readings over the past 30 seconds. Prior work in general aviation safety has established that many safety-critical conditions can be identified from the instantaneous energy and sensor state of the aircraft [4], and multi-sensor flight data has been shown to carry sufficient discriminative

information for classifying anomaly types at the individual data-point level [40]. A single timestep can simultaneously satisfy the conditions for multiple events for example, a flight can exhibit Roll and Low Pitch at the same instant which is why MHANet is designed as a multi-label classifier: each of the 10 output logits is treated independently, and any subset of events can be active at the same time. The total number of parameter is 251,338.

The MHANet architecture is illustrated in Figure 4.6. It consists of five sequential stages which are SensorNorm, Per-Sensor Independent Embedding, Positional Encoding, two Transformer Blocks, and a three-layer Classification Head.

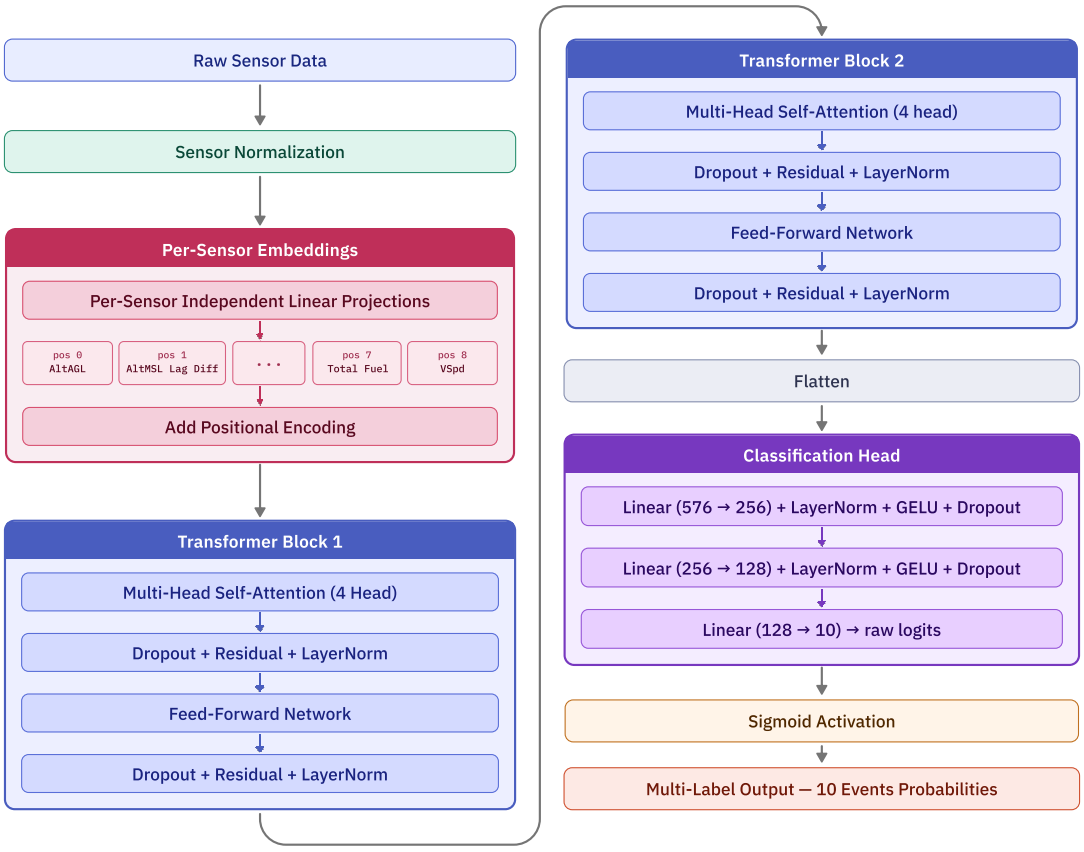

Figure 4.6: Model Architecture of MHANet

SensorNorm:
Raw unscaled sensor values are passed directly into the model, with normalization handled internally by the SensorNorm layer. This design means that during deployment, a new sensor reading can be fed to the model without any external preprocessing step. Training-set statistics (mean and standard deviation per sensor) are computed using sklearn's StandardScaler and stored as fixed non-learnable buffers:

$$\hat{x}_c = \frac{x_c - \mu_c}{\sigma_c} \tag{4.25}$$

The buffer shape is 1×9, one mean and one standard deviation per sensor. Broadcasting applies the normalization independently to every sample in the batch. If any sensor has zero variance in the training data, sklearn's StandardScaler

automatically sets its scale to 1.0, so no manual epsilon correction is needed. The output shape is $B \times 9$.

Per-Sensor Independent Embedding:
The nine normalized scalar sensor values must be lifted into a high-dimensional representation before attention can be applied. MHANet uses nine independent linear layers one per sensor each mapping a single scalar to a 64 dimensional vector:

$$\text{token}_i = W_i \cdot x_i, \qquad W_i \in \mathbb{R}^{64\times 1}, \qquad i = 0, 1, \ldots, 8 \tag{4.26}$$

A single shared linear layer would make the embedding of AltAGL = 1.5 and Roll = 1.5 identical, even though they represent physically unrelated quantities. Using separate linear layers means each sensor learns its own projection that reflects its physical meaning. The weight matrix W_i encodes both the sensor's identity and how its numerical value maps to a meaningful representation. This combines the roles that were handled separately in CSPT and HiCST (value projection plus identity embedding) into a single step. The nine $B \times 64$ vectors are stacked to form a $B \times 9 \times 64$ token sequence representing nine sensor tokens per sample.

Positional Encoding:
A learnable positional encoding of shape $1 \times 9 \times 64$ is added to the sensor token sequence:

$$\text{tokens} = \text{tokens} + \text{pos\_encoding}, \qquad \text{pos\_encoding} \in \mathbb{R}^{1\times 9\times 64} \tag{4.27}$$

Here "position" refers to the sensor index (0 through 8), not time. Self-attention is permutation-invariant, so without this encoding the model cannot distinguish sensor 0 (AltAGL) from sensor 5 (Roll) by position alone. The positional encoding provides each sensor index with a unique learnable signature that supplements the per-sensor embedding as an additional discriminating signal. Learnable encodings are preferred over sinusoidal because the number of sensors is always fixed at 9. The encoding is initialized with small random values (scaled by 0.02) to avoid adding excessive noise to the sensor embeddings at the start of training. The output shape is B $\times$ 9 $\times$ 64.

Transformer Blocks:
The B $\times$ 9 $\times$ 64 token sequence passes through two stacked Transformer blocks. Two blocks are used rather than three because the input sequence contains only 9 sensor tokens a much smaller sequence than the temporal windows processed in Stage 1. Two blocks are sufficient to capture the cross-sensor interactions needed for event classification without over-parameterizing a comparatively small input space.

Each block contains Multi-Head Self-Attention followed by a Feed-Forward Network. Unlike CSPT and HiCST, which use Pre-LayerNorm, MHANet uses Post-LayerNorm (normalization after the residual addition). With only two blocks and a 9 token sequence, the gradient stability benefit of Pre-LayerNorm is not critical, and Post-LayerNorm provides better representation quality in shallow architectures.

Multi-Head Self-Attention:
Four attention heads operate over the 9-sensor sequence, each in a 16-dimensional subspace. Every sensor attends to every other sensor, learning relationships such as altitude-airspeed (AltAGL and IAS), pitch-stall (Pitch and Stall Index), or roll-acceleration (Roll and NormAc):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{16}}\right)V \tag{4.28}$$

Dropout ($p = 0.30$) is applied to the attention output. The residual connection and Post-LayerNorm follow:

$$x = \text{LayerNorm}\Big(x + \text{Dropout}(\text{Attention}(x))\Big) \tag{4.29}$$

Feed-Forward Network. A two-layer FFN with 2× expansion is applied after attention:

$$\text{FFN}(x) = W_2 \cdot \text{GELU}(W_1 \cdot x + b_1) + b_2, \qquad W_1 \in \mathbb{R}^{128\times 64} \tag{4.30}$$

The expansion ratio is 2× (64→128→64), smaller than the 4× ratio used in CSPT and HiCST. Because the input is a single timestep rather than a temporal window, the FFN only needs to model cross-sensor relationships rather than temporal dynamics a smaller intermediate space is sufficient and reduces the risk of overfitting on a relatively compact input. GELU is used as the activation to preserve the signal of negative normalized sensor values, which carry physical meaning (e.g., negative pitch or descent rate). Dropout (p = 0.30) is applied after the FFN, followed by a residual connection and Post-LayerNorm. After both blocks, the output shape is B × 9 × 64.

Flatten and Classification Head:
After the two Transformer blocks, the B × 9 × 64 tensor is flattened to B × 576 (since 9 × 64 = 576). All nine sensor representations are concatenated into a single vector before classification. A CLS token is not used here unlike the temporal sequences in Stage 1 where some timesteps or patches carry more anomaly-relevant information than others, all nine sensors contribute to the definition of every event type. Flattening all sensor representations and passing them jointly through the classification head ensures no sensor's information is selectively de-emphasized by design.

The classification head consists of three linear layers with progressively decreasing width:

Layer 1: 576 → 256: The first linear layer maps the concatenated sensor representations to a 256-dimensional space, followed by LayerNorm, GELU, and Dropout (p = 0.30). LayerNorm is used instead of BatchNorm throughout the classification head because, in the highly imbalanced multi-label dataset, some rare event classes (e.g., Low Fuel, High Altitude Spin) may have no positive samples in a given batch. BatchNorm's running statistics become unreliable under this condition; LayerNorm normalizes each sample independently and is unaffected by batch composition.

Layer 2: 256 → 128: The second linear layer further compresses the representation to 128 dimensions, again followed by LayerNorm, GELU, and Dropout (p = 0.225). The slightly reduced dropout (0.30 × 0.75) reflects the principle that as the classification head approaches the output layer, representations become increasingly task-specific; aggressively dropping those features late in the network risks discarding the precise discriminative patterns learned for rare event classes.

Layer 3: 128 → 10: The final linear layer produces 10 raw logits, one per event class. No activation is applied at the output the Asymmetric Loss function used during training applies sigmoid internally and computes the multi-label binary cross-entropy. The 10 outputs are treated independently because MHANet is a multi-label classifier: any combination of events can be simultaneously active at a single timestep. During inference, a sigmoid is applied to each logit and a per-class threshold (optimized on the validation set) determines the final multi-hot prediction. The output shape is B × 10.

4.2.2 Model Training Strategy

4.2.2.1 Stage 1 Training Strategy: Binary Anomaly Detection (CSPT & HiCST):

Both CSPT and HiCST in Stage 1 are trained using the same training strategy. Since the dataset contains approximately 99.8% normal windows and only 0.2% anomalous windows, the training pipeline is specifically designed around this severe class imbalance. The following subsections describe each component of the training strategy in order of execution.

Reproducibility and Hardware Settings:
Before model initialization, all random sources are fixed with a seed of 42 (Python random, NumPy, PyTorch CPU, and PyTorch CUDA). This ensures that weight initialization, data shuffling, and dropout patterns are identical across runs. Two hardware-level settings are applied alongside this: cuDNN benchmark mode is enabled so that cuDNN automatically selects the fastest convolution algorithm for the specific hardware and input size, and float32 matrix multiplication is set to high-precision Tensor Core mode, which trades a marginal amount of numerical precision for a meaningful speed gain.

Data Loading and Sliding Window Construction: Raw sensor data is loaded from the pre-split CSV files. Sliding windows of length W = 30 are extracted from each flight's time series. The label assigned to each window is the label of its final timestep. Flight boundaries are strictly respected. After window construction, the class imbalance becomes explicit in the training set. Two complementary mechanisms address this imbalance in the training pipeline: a Weighted Random Sampler in the DataLoader and a Focal Loss function. These are described in the next two subsections.

Weighted Random Sampler: Instead of standard random shuffling, the training DataLoader uses a WeightedRandomSampler. Each training sample is assigned a weight inversely proportional to the size of its class:

$$w_i = \frac{N}{2 \cdot N_c} \tag{4.31}$$

where N is the total number of training samples and N_c is the number of samples in that sample's class. Anomalous windows receive a much higher weight than normal windows, causing the sampler to draw them more frequently in each epoch. Sampling is performed with replacement (replacement=True), so the same anomalous window may appear multiple times within a single epoch. This is acceptable here because no data augmentation is available for sensor time series, making oversampling the primary tool for exposing the model to anomalous patterns. The WeightedRandomSampler is applied only during training; validation and test sets retain the original class distribution to ensure evaluation reflects real-world operating conditions.

Focal Loss with Label Smoothing:

Standard cross-entropy loss is unsuitable for this dataset: a model that always predicts "normal" would achieve very low loss while detecting no anomalies. Focal Loss addresses this by down-weighting easy examples and focusing training on hard, misclassified samples:

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{4.32}$$

where p_t is the predicted probability for the ground-truth class. The modulating factor $(1-p_t)^{\gamma}$ suppresses the contribution of easy examples. For a sample the model classifies correctly with high confidence ($p_t = 0.95$), the factor is $(0.05)^{2.8} \approx 0.0003$, making its loss contribution nearly zero. For a hard example ($p_t = 0.3$), the factor is $(0.7)^{2.8} \approx 0.37$, preserving a meaningful loss signal. The focusing exponent $\gamma = 2.8$ is set slightly higher than the standard value of 2 to account for the extreme 99.8%–0.2% imbalance; a larger γ more aggressively suppresses the easy normal samples that dominate the dataset.

The class-weighting parameter α is computed dynamically from the training class distribution rather than being hard-coded:

$$\alpha = \text{clip}\left(\frac{w_{\text{anomaly}}}{w_{\text{normal}} + w_{\text{anomaly}}} \times 1.2,\ 0.80,\ 0.85 \right) \tag{4.33}$$

The ratio is boosted by 1.2 and clipped to $[0.80, 0.85]$. This means the anomaly class contributes up to 85% of the total loss signal. Dynamic computation is used because the exact class ratio can vary slightly between different data splits; a fixed α would not always be optimal. Label smoothing with $\epsilon = 0.02$ is also applied. Instead of hard targets of 0 or 1, the targets are softened to 0.01 and 0.98:

$$\tilde{y} = y \cdot (1 - \epsilon) + \epsilon / C \tag{4.34}$$

This prevents the model from becoming overconfident on ambiguous borderline samples, which exist in flight sensor data where event boundaries are not always sharply defined. The smoothing value is kept small (0.02) so it does not distort the primary supervision signal.

Optimizer:

The AdamW optimizer is used with an initial learning rate of 1.2×10^{-5} and weight decay of 1×10^{-4}. AdamW is preferred over standard Adam because it decouples

weight decay from the gradient update, applying it directly to the weights rather than incorporating it into the adaptive learning rate:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} - \lambda\theta_t \tag{4.35}$$

In standard Adam, L2 regularization is absorbed by the adaptive scaling and becomes less effective. AdamW's explicit weight penalty shrinks large weights directly, discouraging the model from over-relying on any individual feature. The weight decay value of 1×10^{-4} is conservative sufficient for regularization without interfering with the learning dynamics. The initial learning rate is set low (1.2×10^{-5}) intentionally, as it will be controlled by the OneCycleLR scheduler described next.

Learning Rate Scheduler: OneCycleLR
A OneCycleLR scheduler controls the learning rate across 150 training epochs in two phases:

Table 4.1: Configuration parameters for the OneCycleLR scheduler.

Phase	**Duration**	**Behaviour**
Warmup	First 10% of steps	LR rises from 1.2×10^{-5} to 3×10^{-4} (cosine curve)
Decay	Remaining 90% of steps	LR falls from 3×10^{-4} to $\sim 1.2 \times 10^{-8}$ (cosine curve)

The warmup phase prevents the model from making destructive weight updates in the first few epochs when the representations are not yet meaningful. After reaching the peak learning rate, the cosine decay guides the model toward a smoother minimum. Compared to a simple CosineAnnealingLR (which only decays), the warmup-then-decay cycle gives the model an opportunity to first explore the loss landscape at a moderate rate before committing to a fine-tuning regime. The scheduler steps per optimizer update, not per epoch, which is necessary because gradient accumulation means an optimizer step does not occur at every batch.

Gradient Accumulation:
Gradients are accumulated over 4 consecutive batches before performing an optimizer step, giving an effective batch size of $1024 \times 4 = 4096$. A larger effective batch size reduces gradient variance each update is based on a more representative sample of the data and supports the use of a higher peak learning rate. To ensure the accumulated loss is correctly scaled, each batch's loss is divided by 4 before the backward pass.

$$\ell_{\text{scaled}} = \ell / N_{\text{accum}} \tag{4.36}$$

Without this division, the sum of four backward passes would produce gradients four times larger than intended. After every fourth batch, the gradients are unscaled (required by AMP, described next), clipped, and the optimizer step is taken. Gradients are then zeroed by setting them to None rather than zero, which is slightly faster and uses less memory.

Automatic Mixed Precision (AMP):
The forward pass and loss computation are performed in float16, while the model weights remain in float32. On modern GPUs, float16 operations run approximately twice as fast and use half the memory of float32. A GradScaler is used to prevent float16 gradient underflow: the loss is multiplied by a large scale factor before the backward pass, and the factor is removed from the gradients before the optimizer step. Gradient clipping must occur after unscaling because clipping the still-scaled gradients would apply the norm threshold at the wrong magnitude. Validation inference also runs under autocast for speed, though GradScaler is only active during training.

Exponential Moving Average (EMA) of Weights:
Alongside the actual model weights, a shadow copy of the weights is maintained and updated after every optimizer step:

$$\theta_{\text{shadow}} = 0.9999 \cdot \theta_{\text{shadow}} + 0.0001 \cdot \theta_{\text{current}} \tag{4.37}$$

The shadow weights are a smoothed running average of the training trajectory. Because training loss landscapes involve oscillation especially early in training the instantaneous weights at any given step may be noisy. The EMA, by averaging over many recent steps, is significantly more stable. A decay of 0.9999 corresponds to an effective averaging window of approximately $1/(1 - 0.9999) = 10{,}000$ optimizer steps. All validation evaluation and model checkpointing use the EMA weights exclusively; the best checkpoint is saved from the EMA model, not the live training model.

Regularization:
Three regularization mechanisms operate throughout training. Dropout is applied at three locations in the architecture: $p = 0.10$ in the cross-sensor attention layers, $p = 0.20$ in the feed-forward networks, and $p = 0.10$ in the CrossSensor module. These values are deliberately graduated higher dropout where the representations are more complex and more prone to memorization. Dropout is automatically disabled during validation and inference when model.eval() is called. Weight decay (L_2 regularization) with $\lambda = 1 \times 10^{-4}$ is applied through AdamW, penalizing large weight magnitudes and preventing the model from over-fitting to specific sensor patterns in the training set. Gradient clipping is applied with a maximum global L_2 norm of 1.0:

$$g \leftarrow g \times \frac{1.0}{\|g\|_2} \quad \text{if } \|g\|_2 > 1.0 \tag{4.38}$$

Flight sensor data occasionally contains batches with unusually large sensor readings that can produce very large gradients, destabilizing weight updates. Clipping prevents these exploding gradients without otherwise interfering with the optimization dynamics.

Dynamic Threshold Tuning:
After collecting validation probabilities at the end of each epoch, a threshold sweep is performed over the range $[0.20, 0.99]$ in steps of 0.01. For each candidate threshold, the binary F1 score on the validation set is computed, and the threshold

yielding the highest F1 is recorded as the epoch's optimal threshold. Because the model's probability distribution shifts as training progresses, re-sweeping the threshold every epoch ensures the optimal decision boundary always reflects the model's current calibration. The threshold is never selected using the test set.

Early Stopping:
Training runs for a maximum of 150 epochs with early stopping based on validation F1 with a patience of 25 epochs. If validation F1 does not improve for 25 consecutive epochs, training is terminated. The best checkpoint the EMA weights at the epoch with the highest validation F1 is saved and used for all subsequent evaluation. Validation loss is not used as the stopping criterion because, under severe class imbalance, a model can achieve low loss by predicting normal for every sample while still failing to detect any anomaly. F1, which jointly measures precision and recall, is a more meaningful stopping metric for anomaly detection.

4.2.2.2 Stage 2 Training Strategy: Multi-Label Event Classification (MHANet)

MHANet uses a different training strategy from CSPT and HiCST. The key differences stem from two facts MHANet operates on individual timesteps (B $\times$ 9) rather than windows, and it simultaneously classifies 10 event types under an even more severe per-class imbalance some events have as few as 10–20 positive training samples. The reproducibility setup (SEED = 42) is identical to Stage 1. cuDNN benchmark mode and AMP are not used here because the input dimensionality is far smaller, the batch size is only 256, and training speed is already adequate without these optimizations.

Loss Function: Asymmetric Loss (ASL) with Class Weights:
Focal Loss, used in Stage 1, applies the same focusing exponent to both positive and negative easy examples. For multi-label classification with 10 severely imbalanced event classes, this is insufficient: the dataset is over 99% negative for every event, meaning the model is overwhelmed by easy negative samples during training. Asymmetric Loss (ASL) solves this by applying different focusing exponents to positive and negative cases:

$$L_{\text{pos}} = -(1-p)^{\gamma_{\text{pos}}} \log(p) \qquad [\gamma_{\text{pos}} = 1.0] \tag{4.39}$$

$$L_{\text{neg}} = -p_m^{\gamma_{\text{neg}}} \log(1-p_m) \qquad [\gamma_{\text{neg}} = 4.0,\ p_m = \min(p+0.05, 1)] \tag{4.40}$$

$$L_{\text{ASL}} = \frac{1}{1+\exp(-L_{\text{pos}})} \cdot \log(p) + p_m^{\gamma_{\text{neg}}} \cdot \log(1-p_m) \tag{4.41}$$

$\gamma_{\text{neg}} = 4.0$ **versus** $\gamma_{\text{pos}} = 1.0$**:** Easy negative examples (the dominant class) are suppressed far more aggressively than positive ones. With $\gamma_{\text{neg}} = 4.0$, a correctly classified negative sample with $p \approx 0.05$ contributes essentially zero gradient, so learning is driven almost entirely by the rare positive samples. This asymmetry is the core advantage over Focal Loss, which treats both directions with equal suppression.

clip $= 0.05$**:** The shifted probability $p_m = \min(p + 0.05, 1)$ applied in the negative case prevents near-zero probabilities from being entirely excluded from the gradient. Without this shift, a model that consistently assigns very small probabilities to a rare event would receive near-zero gradient for all its negative predictions, effectively stopping learning for that event class.
Class weights are computed as inverse event frequency (negative count ÷ positive count per class), then normalized so all weights sum to 10:

$$w_c = \frac{N_{\text{neg},c}}{N_{\text{pos},c}} \tag{4.42}$$

then

$$w_c \leftarrow \left(\frac{w_c}{\sum_c w_c} \right) \times 10 \tag{4.43}$$

Normalization keeps the loss magnitude stable so the learning rate requires no manual adjustment for different class distributions. The two rarest events receive an additional manual boost: Low Fuel × 2.5 and High Altitude Spin × 1.5, determined empirically during model development.

Optimizer: AdamW:
AdamW is used for the same reason as in Stage 1 decoupled weight decay gives correct L2 regularization. However, the hyperparameters differ significantly: lr $= 1 \times 10^{-3}$ (higher than Stage 1's 3×10^{-4}) and weight_decay $= 1 \times 10^{-2}$ (100× stronger than Stage 1's 1×10^{-4}). The stronger weight decay is necessary because the input space is only 9-dimensional a model with such a small input can memorize training patterns very easily, so stronger regularization is required. The higher initial learning rate is appropriate because MHANet uses CosineAnnealingWarmRestarts, which begins decaying from the first epoch without a warmup phase; a higher starting value ensures sufficient parameter exploration before the first decay cycle completes.

Learning Rate Scheduler: CosineAnnealingWarmRestarts:
Instead of OneCycleLR (used in Stage 1), MHANet uses CosineAnnealingWarmRestarts with $T_0 = 70$, $T_{\text{mult}} = 2$, and $\eta_{\min} = 1 \times 10^{-9}$. The learning rate follows:

$$\eta_t = \eta_{\min} + \frac{1}{2} (\eta_{\max} - \eta_{\min}) \left(1 + \cos \left(\pi \frac{T_{\text{cur}}}{T_i} \right) \right) \tag{4.44}$$

The LR decays from 1×10^{-3} to 1×10^{-9} over the first 70 epochs, then resets and repeats over 140 epochs, then 280, and so on. The warm restart mechanism is particularly important for multi-label training: different event classes may converge at different rates within a single LR cycle, and restarting the LR allows the model to revisit parts of the loss surface and improve on classes that did not converge in the previous cycle. OneCycleLR, by contrast, reduces the LR to near-zero permanently after one cycle, eliminating any further exploration. $T_{\text{mult}} = 2$ ensures each subsequent cycle is longer, giving the model progressively more time to fine-tune on top of what was learned in the previous cycle. The scheduler steps once per epoch (not per batch) because there is no gradient accumulation in MHANet; stepping per batch would complete the 70-epoch cycle far too quickly.

Regularization:
MHANet uses the same three regularization mechanisms as Stage 1 dropout, weight decay, and gradient clipping (max norm = 1.0) but with stronger settings throughout. Dropout is 0.30 in both the Transformer attention/FFN layers and the classification head (compared to 0.10–0.20 in Stage 1), and weight decay is 1×10^{-2} (compared to 1×10^{-4} in Stage 1). Both are higher because the small input dimensionality (9 features per timestep) makes the model considerably more susceptible to overfitting. Since AMP is not used, gradient clipping is applied directly to the gradients without a prior unscale step.

Training Loop Simplifications:
MHANet's training loop is simpler than Stage 1 in three respects. There is no gradient accumulation each batch triggers an immediate optimizer step. There is no AMP the small input size makes float16 speed gains unnecessary. There is no EMA the multi-label loss surface is smoother than the binary detection task in Stage 1, so weight oscillation is less severe and stable checkpoints can be obtained directly from the live model weights. Validation computes both macro F1 and micro F1 using a default threshold of 0.5.

Early Stopping:
Training runs for a maximum of 300 epochs with early stopping patience of 30 (compared to 25 in Stage 1). The longer patience accounts for the CosineWarmRestarts schedule: with T = 70 epochs per cycle, a patience of 25 could terminate training before the first full cycle completes, preventing the model from benefiting from the restart. Validation macro F1 is the stopping criterion it averages the F1 score separately for each of the 10 event classes, making poor performance on rare events clearly visible in the metric. The best checkpoint is saved directly from the model weights (no EMA).

Per-Class Threshold Tuning:
After training, a per-class threshold search is performed on the validation set. Unlike Stage 1 (which optimizes a single global threshold for binary detection), MHANet optimizes a separate threshold for each of the 10 event classes by sweeping t over [0.05, 0.95] in steps of 0.01 and selecting the value that maximizes the per-class F1 on the validation set. Each event class has a different number of positive training samples and a different output probability distribution from the model a single shared threshold would be appropriate for at most one class. Per-class thresholds are determined solely on the validation set; the test set is never used for threshold selection.

4.3 Data Collection

The data used in this study is taken from the the National General Aviation Flight Information Database (NGAFID) which is a Flight Data Monitoring (FDM) project funded by the Federal Aviation Administration (FAA) as a part of the Aviation Safety Information Analysis and Sharing (ASIAS) program. The purpose of NGAFID was to develop an open-access platform where flight training organizations, pilots and operators can upload and analyze multivariate sensor data collected from the onboard flight data [11]. General aviation has historically lacked centralized infrastructure unlike commercial aviation which has organized systems such as FOQA to monitor safety. NGAFID identifies and solves this limitation by offering a secure and de-identified repository, which already hosts over 900,000 hours of data on flight hours. As this it need to handle a large scale of time series data so in order to manage efficiency, they use a per-parameter compressed BLOB storage strategy [11].

The important part of NGAFID is the automated event detection system that analyses the flight recordings according to 20 safety event definitions. This system detects and indicates the possible unsafe flight conditions on a per-flight basis. But the full NGAFID database can be accessed by developers and administrators only, and the general users are only able to access their personal uploaded data. In this study, a subset of the data was curated in partnership with NGAFID stakeholders. Where all sensitive information like pilot identity, flight tail number, and location was removed before realising it so that it complies with privacy regulations including FERPA and other applicable agreements.

The dataset that is used in this study is the General Aviation Training Set (GATS), which is a publicly accessible benchmark dataset based on NGAFID and for machine learning tasks in the aviation safety field [35]. This dataset was collected from two flight training schools in united state of America for around two months. It has a total of 10,641 flight hours from 7,679 individual flights. There are 76 parameters and is sampled at 1 Hz meaning one data point is recorded per second. In these parameters, there are navigational, aerodynamic, engine-related, and environmental parameters, which give a multivariate representation of the flight conditions [35]. This is one of the biggest publicly available datasets for general aviation research which is also under the CC-BY 4.0 license. Table 4.1 gives an overview of the main features of the GATS data.

Table 4.2: Summary of high-level attributes of the GATS dataset.

Dataset Attribute	**Value**
Total Flight Hours	10,641
Total Number of Flights	7,679
Unique Aircraft Types	3
Sampling Frequency	1 Hz
Number of Sensor Parameters	76
Data Licence	CC-BY 4.0

The dataset used in this study is mainly consist of data from the Student pilot training and evaluation flights. The dataset has raw sensor reading from the flights along with different derived features like stall index and density ratio. These derived features are calculated programally from the original sensor data which can be used to identify critical flight conditions such as detection of loss of control in flight and stall related events [11]. The flight times differ greatly, with the shortest flight times of about 3 minutes and the longest flight times of up to 9 hours, which is important for the variation in training operations.

The GATS data consists of three aircraft categories commonly used in the training of general aviation Cessna 172S, Piper PA-28-181 and Piper PA-44-180. The Cessna 172S is a high wing single engine airplane, compared to the PA-28-181 and PA-44-180 which are low wing aircraft. Also, the PA-44-180 is a twin-engine aircraft, which adds an additional complexity to the dataset as it has eleven more sensor columns concerning the second engine. As there is a difference between the structural and mechanical distinctions among the different aircraft, this can affect the sensor arrangements and the flight dynamics in the data.

As illustrated in Figure 4.7, the dataset is heavily skewed toward the Cessna 172S, which accounts for 4,481 flights (58.3% of the total), compared to 2,214 flights for the PA-28-181 and 984 for the PA-44-180. Table 4.3 presents the full aircraft distribution.

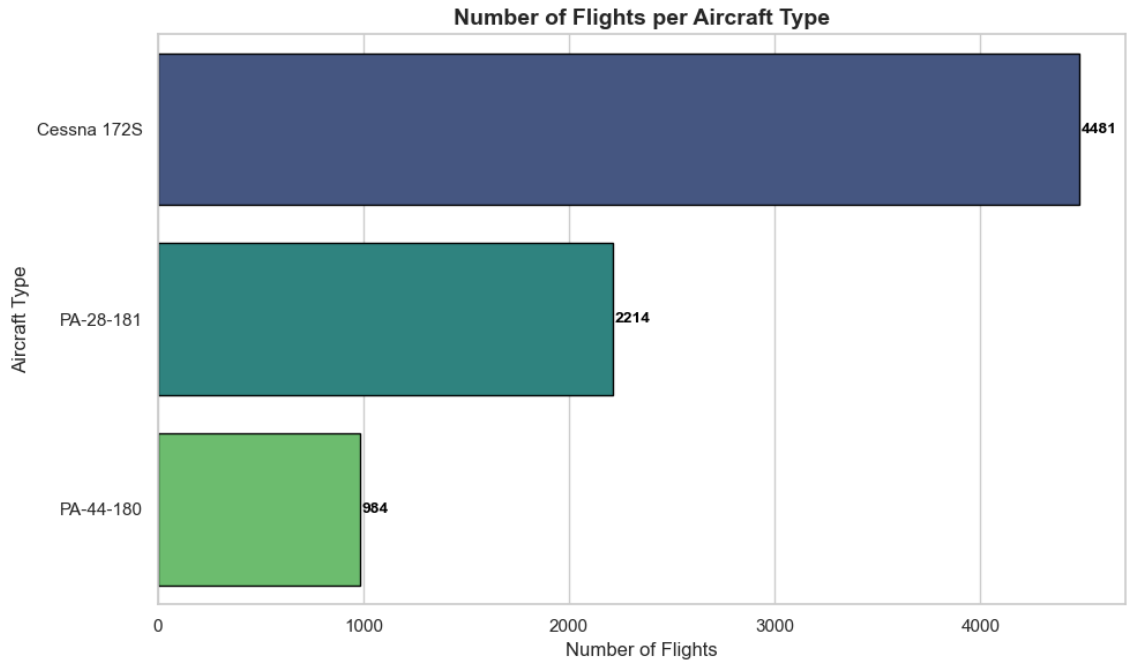

Figure 4.7: Number of Flights per Aircraft Type

Table 4.3: Distribution of flights by aircraft type in the GATS dataset.

Aircraft Type	**Flight Count**	**Percentage of Total**
Cessna 172S	4,481	58.3%
PA-28-181	2,214	28.8%
PA-44-180	984	12.8%
Total	**7,679**	**100%**

This imbalance in the dataset is even greater when the distribution of the anomaly events among the aircraft types is analyzed. As Figure 4.8 shows, most of the anomaly events around 8,383 of the 8,802 events or about 95.2% of all the anomalies detected are related to the Cessna 172S. Comparatively, the PA-28-181 adds 340 events, whereas the PA-44-180 only adds 79 events. Because of this great imbalance in the dataset the available samples of anomalies in the PA-28-181 and PA-44-180 are too small to undergo successful training and evaluation of deep learning models separately.

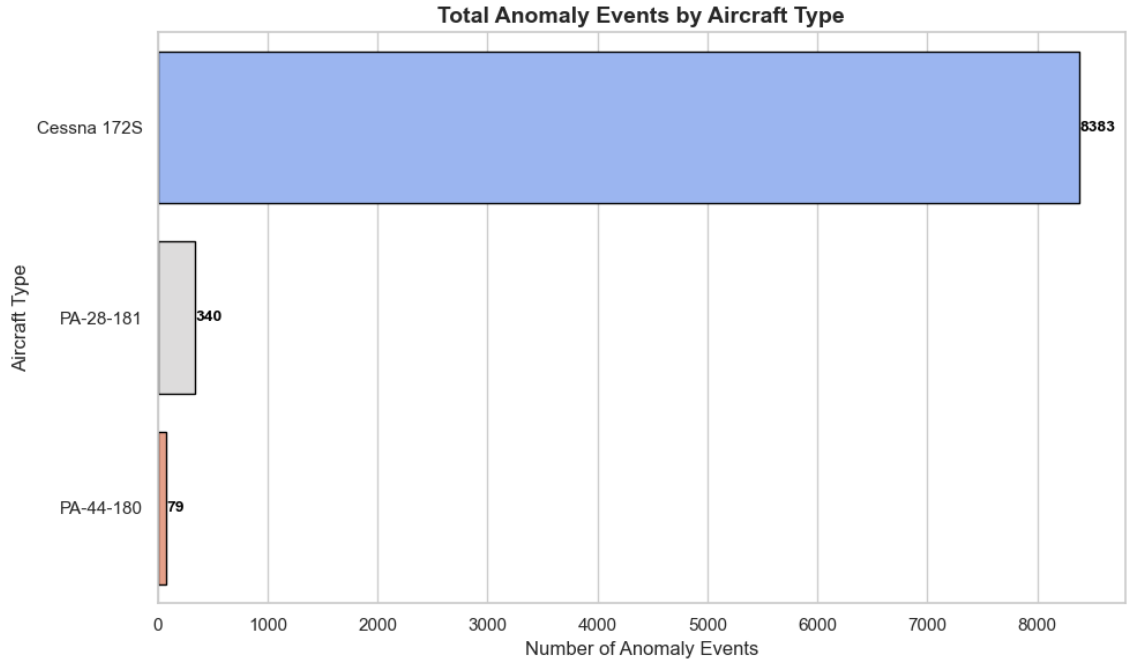

Figure 4.8: Total Anomaly Events by Aircraft Type

The dataset is highly imbalanced not only in the number of flights but also in the distribution of the anomaly events among aircraft types. As shown in Figure 4.8, the Cessna 172S contributes 8,383 of the 8,802 reported anomaly events, which is about 95.2% of all the reported anomaly events. This paper will only concentrate on the Cessna 172S subset due to its overwhelming dominance in the number of flights and the number of anomalies. This is not only because of the availability of large volume of data but also it makes the aerodynamic behavior and sensor configurations consistent across all training and evaluation examples. This is important to make models that will learn genuine flight dynamics rather than spurious inter-aircraft configuration differences. Among the Cessna 172S, there are

18 different types of anomaly events identified, out of 20 types of anomaly events defined by the NGAFID event detection system.

These anomalies are related to certain in-flight safety events like Loss of Control In-Flight (LOC-I), which is still one of the most frequent causes of fatal accidents in general aviation [24], [38]. LOC-I events are the ones that are characterized by the deviation of the intended flight path as a result of stalls, spins or unintended energy management.

From Figure 4.9 it can be seen how different types of anomalies are present in the GATS dataset. The type of anomaly events distribution is highly right-skewed, where High Altitude Stall is present 4,616 times then Low Airspeed on Climbout occurs 1,810 times, VSI on Final for 606 and Low Airspeed on Approach for 596 times. On the other hand, there are a few event types that happen very infrequently, such as High Lateral Acceleration, Cylinder Head Temperature anomalies, Prox warnings, and Engine Shutdown Below 3000 Ft, which are represented by a single to three instances in the whole data. Such a severe imbalance in the event classes poses a major challenge to multi-label classification, which escribed further in Section 4.4.

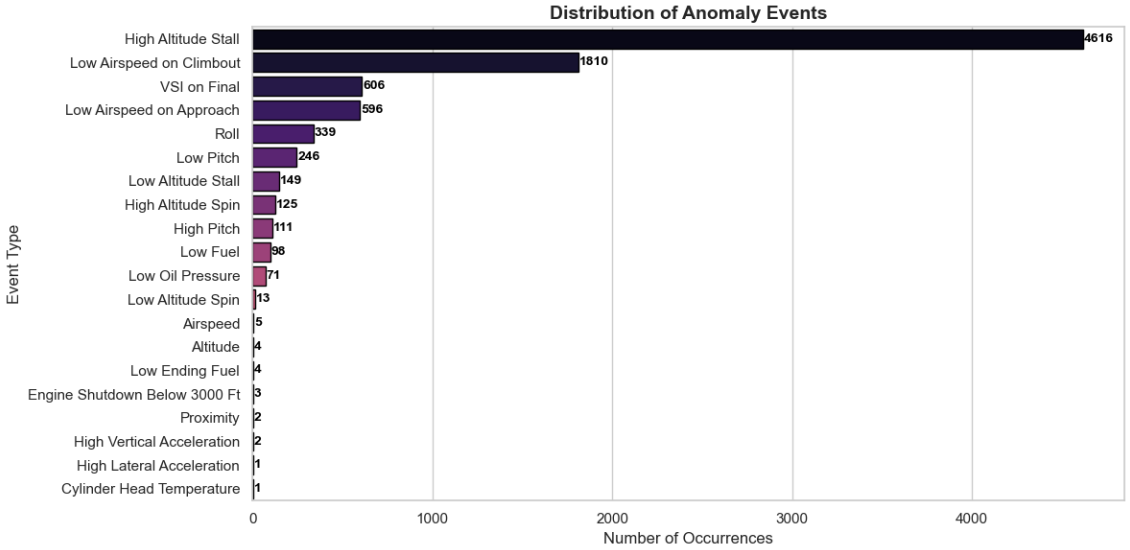

Figure 4.9: Distribution of all 20 observed anomaly event types

Table 4.4 shows the full breakdown of the anomaly events per aircraft type. It is important to note that some of these critical events are only witnessed in the Cessna 172S in events like High Altitude Stall, High Altitude Spin, Low Altitude Stall, and Low Altitude Spin. These events are closely related to the stall and spin training maneuvers, which are typically used in high-wing, single-engine training aircraft like Cessna 172S, which is another reason the Cessna 172S was chosen as the target of the current research.

Table 4.4: Anomaly event occurrence by aircraft type.

Event Name	**Cessna 172S**	**PA-28-181**	**PA-44-180**
Airspeed	5	0	0
Altitude	4	0	0
Cylinder Head Temperature	0	0	1
Engine Shutdown Below 3000 Ft	0	0	3
High Altitude Spin	125	0	0
High Altitude Stall	4,616	0	0
High Lateral Acceleration	1	0	0
High Pitch	92	18	1
High Vertical Acceleration	2	0	0
Low Airspeed on Approach	553	43	0
Low Altitude Spin	13	0	0
Low Altitude Stall	149	0	0
Low Ending Fuel	4	0	0
Low Fuel	62	36	0
Low Oil Pressure	28	7	36
Low Pitch	243	3	0
Proximity	1	1	0
Roll	315	13	11
VSI on Final	398	182	26
Total Events	**8,383**	**340**	**79**

In addition to the imbalance between the types of anomaly events across classes, the subset of Cessna 172S has an even worse imbalance at the time-series row level. The dataset has 22,106,847 sensor reading rows with only 36,543 rows (0.17%) labeled as an anomaly and 22,070,304 rows (99.83%) being completely normal flight conditions. This gives a ratio of 1:603 between anomalous and normal time steps meaning that there are 603 normal seconds to every second of sensor data that is anomalous. This is a highly imbalanced ratio that has been extensively recorded as a difficulty in time-series anomaly detection tasks, and cannot be resolved by a set of traditional training processes. To deal with this imbalance, design of loss functions, sampling strategies and evaluation metrics should be carefully considered as explained in Chapter 5.

Table 4.5: Row-level class imbalance statistics for the Cessna 172S.

Metric	**Value**
Total Rows	22,106,847
Normal Rows	22,070,304 (99.83%)
Anomaly Rows	36,543 (0.17%)
Imbalance Ratio (Anomaly : Normal)	1 : 603

This extreme imbalance ratio is a well-documented challenge in time-series anomaly detection tasks and cannot be addressed through standard training procedures alone. Deep learning models trained on such data without correction tend to converge toward predicting the majority (normal) class for every time step, achieving

high overall accuracy while failing entirely on the anomaly class. Addressing this imbalance requires careful consideration in the design of loss functions, sampling strategies, and evaluation metrics, as discussed in Chapter 5.

Another structural property of the data that has direct effects on model design is the distribution of anomaly events per flight. This distribution of the Cessna 172S is shown in figure 4.10. A big percentage of flights of the Cessna 172S around 2,220 occurred zero anomaly events which are normal flight operations. The distribution is steeply decreasing, with about 800 flights having a single event and about 450 having two. Above five events per flight, the number of events decreases exponentially, but there are a few flights with up to 60 concurrent or sequential events, probably due to special stall and spin practice, where a student performs a maneuver repeatedly.

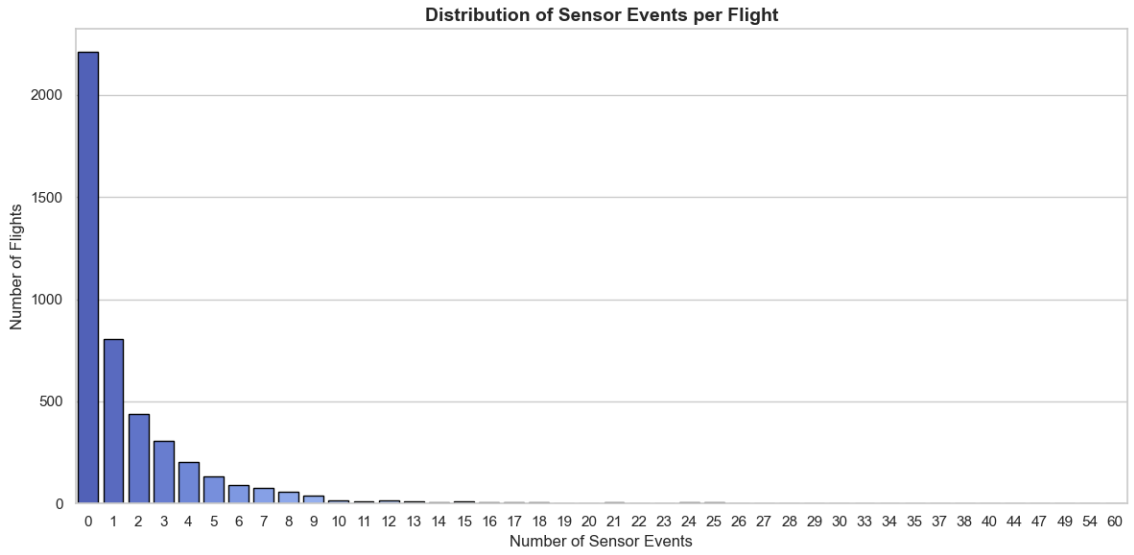

Figure 4.10: Histogram of the number of anomaly events recorded per flight in the Cessna 172S

This long-tailed, right-skewed distribution has important methodological consequences. The concentration of anomalous rows within a small number of high-event flights means that a naive random split of the dataset at the row level would risk grouping time steps from the same flight across both training and test sets introducing data leakage and producing unrealistically optimistic evaluation results. Furthermore, the co-occurrence of multiple event types within individual flights introduces the multi-label character of the problem: a single flight, and indeed a single time step within it, may simultaneously satisfy the trigger conditions for several different event types. These two considerations data leakage prevention and multi-label co-occurrence collectively motivate the stratified, flight-level partitioning strategy described in Section 4.3.3.

While the frequency and types of anomalies are important, the temporal duration of these events provides deeper insight into whether an anomaly is just a short noise or a real safety issue. Since the GATS dataset is recorded at a fixed rate of 1 Hz,

the duration of each anomaly directly corresponds to the number of consecutive anomalous rows. The Cessna 172S subset analysis demonstrates that the duration of anomalies is greatly concentrated around a small period of time and can be divided into four major categories. The distribution is highly skewed to the right with most of the anomalies being very short. The dataset is dominated by the Transient category ∼68%, with 6,109 events having a duration between 1 to 4 seconds whereas a significant number of events having a duration of 1 to 2 seconds. These are typically short-lived disturbances due to wind gusts or rapid corrections by pilots, in which the system soon gets back on track. The Intermediate (5-10 seconds) and Sustained (>10 seconds) events combined constitute approximately 26 percent of the data, and they reflect more extreme cases, when the aircraft is in an unsafe state longer, like low airspeed during approach or stall conditions. However, long duration anomalies are rare with more than 60 seconds. There is also one Extreme Outlier event in the dataset of 901 seconds (approximately 15 minutes). This time distribution has a direct effect on the model design in this study. In particular, a 30-timestep (30-second) sliding window is reasonable, which is large enough to identify over 98% of all anomaly events. This enables the model to acquire the complete time-varying dynamics of anomalies between normal and abnormal behavior, but not respond to individual time steps. Hence enhancing its capacity to identify meaningful safety risks.

Table 4.6: Distribution of Anomaly Event Durations for the Cessna 172S.

Category	**Duration**	**Event Count**	**Percentage of Dataset**
Transient	1–4 seconds	6,109	∼68%
Intermediate	5–10 seconds	1,366	∼15%
Sustained	>10 seconds	1,027	∼11%
Extreme Outlier	∼901 seconds	1	<0.1%

Along with the temporal duration, the complexity of in-flight anomalies is also described in terms of co-occurrence, i.e., the occurrence of multiple safety event definitions in the same time step (one second). The Cessna 172S subset analysis demonstrates that the majority of anomalous rows are isolated, and the single event occurrences are predominant in the dataset. In particular, 33,587 rows (around 95.2% of all labeled rows) have a single anomaly, usually a localised change in one flight parameter like a temporary altitude shift or a temporary decrease in airspeed.

Nevertheless, there is a significant number of overlapping anomalies in the dataset, 1,706 rows of which have two or more events occurring at the same time. Out of these, 1,018 rows have two concurrent anomalies, 613 rows have three, and 75 rows are four simultaneous events. These multi-event occurrences reflect more complicated or severe conditions, like cascading conditions during flight like a high-altitude stall which also causes low airspeed and high-pitch conditions simultaneously. There are no cases in the dataset of five or more concurrent events, so the complexity of the event rules of the NGAFID may have a limit. This distribution shows that the problem is inherently multi-label, with a variety of safety risks potentially happening simultaneously, and the model must predict multiple anomalies at once, instead of considering them independent or mutually

exclusive.

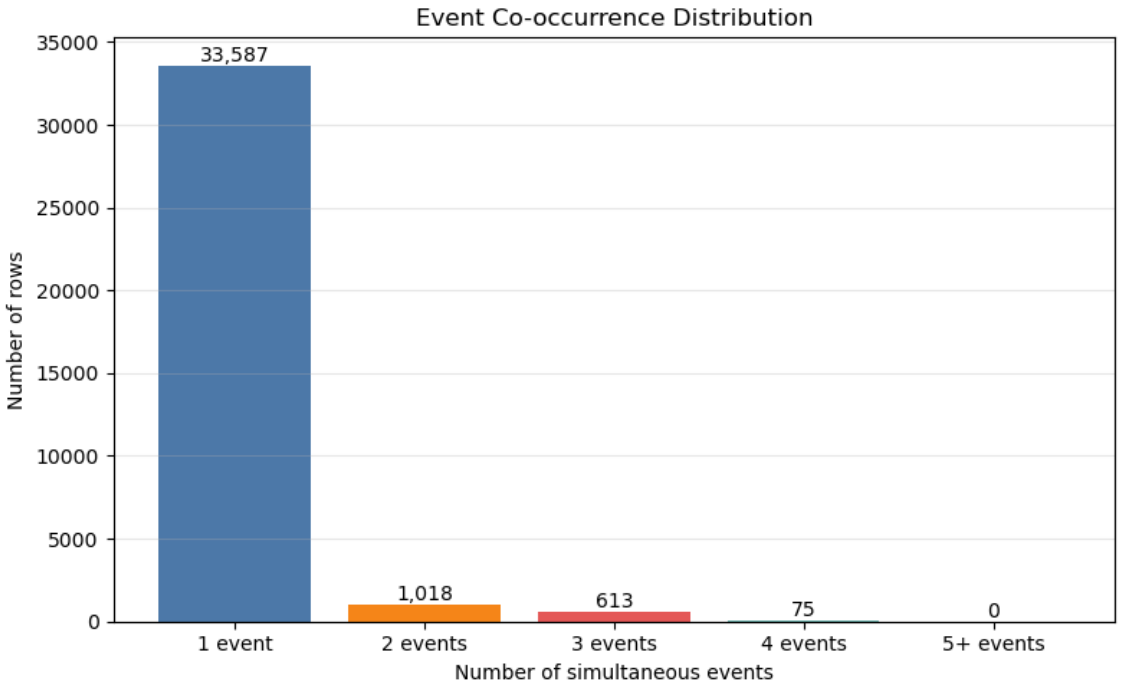

Figure 4.11: Distribution of Event Co-occurrence

4.3.1 Data Reduction

The imbalance ratio of the original GATS dataset is very large 1:603, and this is its biggest issue. This means there is an anomaly of only one second in every 603 seconds of normal flight, which is extremely difficult to be learned by a deep learning model. To fix this, a data reduction strategy is used in which only retain the flights that had one or more events of anomaly. It reduced the data to 2,258 flights. Although the data remains unbalanced, this action made the ratio 1:337, which made the process of training a lot more efficient.

Table 4.7: Distribution of normal and anomalous samples in the reduced GATS dataset.

Only Anomalous Flights Metric	**Statistical Value**
Total Rows	11,963,999
Normal Rows	11,928,706 (99.71%)
Anomaly Rows	35,293 (0.29%)
Imbalance Ratio (A:N)	1 : 337

The other data reduction process was to select which anomaly should track. As shown in Figure 4.12, some events like ”High Altitude Stall” happened 4,616 times, while others, like ”Proximity” happened only once. A model cannot learn about an event that occurs a few times since there is not enough information to divide into training and testing sets. As such, we have chosen to retain the 10 most common

events of anomalies in this study.

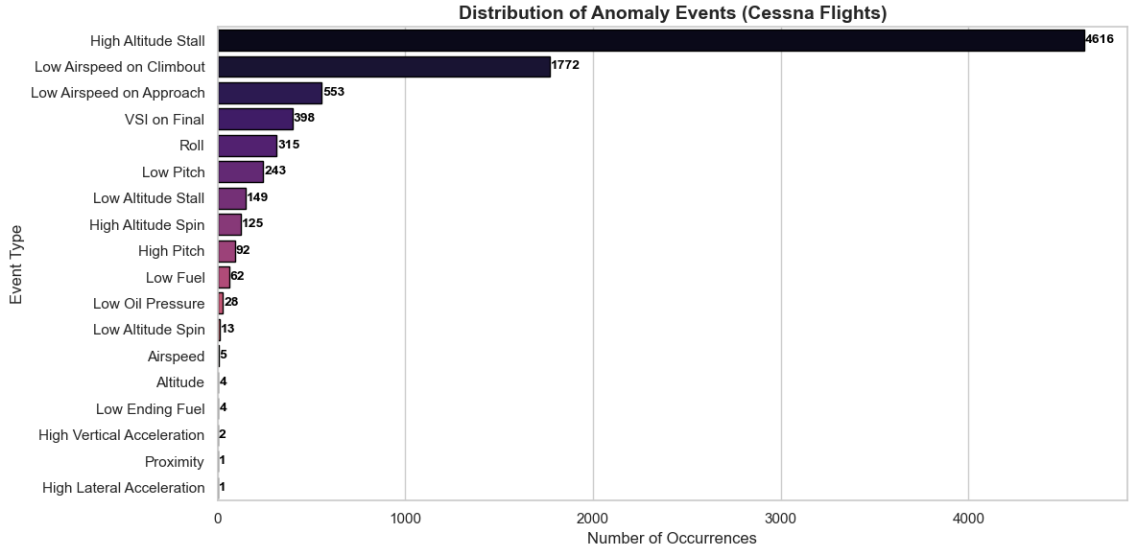

Figure 4.12: Distribution of Anomaly Events in Cessna 172S

Lastly, the sensors needed to be standardized. Figure 4.13 indicates that the various Cessna 172S flights have varying numbers of sensors, with 34 to 60 being the most common. The reason behind this is because various aircrafts are equipped differently or with more updated G1000 avionics system. But a deep learning model requires all the inputs to be of the same shape (the same number of sensors) in order to be functional. So, excess sensors that are not required might actually cause the model to become more confused and reduce its accuracy [30], [21].

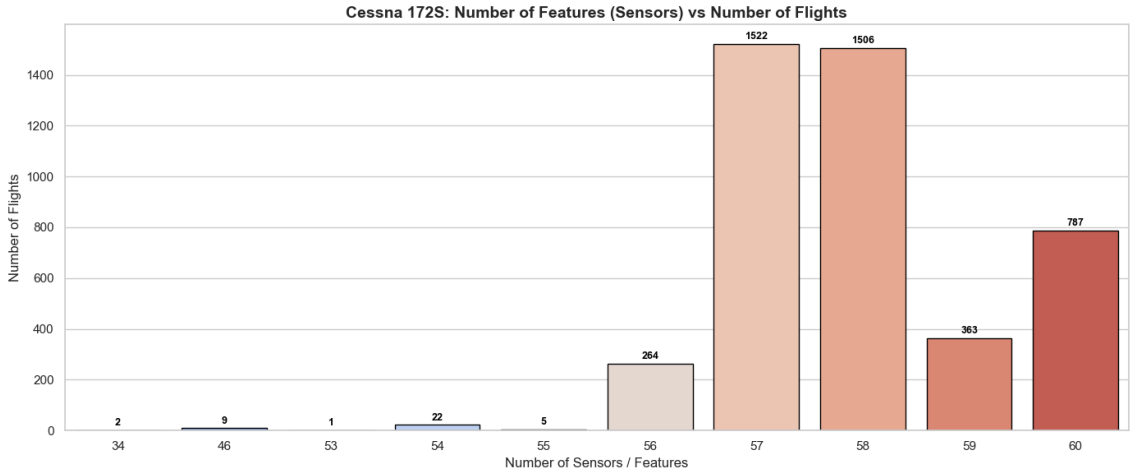

Figure 4.13: Distribution of Sensor Number in Cessna 172S Flights

To find the required sensors, the NGAFID documentation is used. Some events weren't clearly listed (like High Pitch or High Altitude Spin) in the documentation. For these events two statistical methods were used: Random Forest Feature Importance and a Correlation Matrix. The reason why Random Forest is so good is that it prioritizes sensors in order of their contribution to a correct prediction

[28], whereas a Correlation Matrix can determine and eliminate redundant sensors that are merely repeating the same information [20]. Following this extraction, the processed data is narrowed down to 9 final sensors most responsible to the top 10 events: 'AltAGL', 'AltMSL Lag Diff', 'IAS', 'NormAc', 'Pitch', 'Roll', 'Stall Index', 'Total Fuel' and 'VSpd'.

4.3.2 Data Cleaning

After the data reduction and feature extraction steps, the paper discusses the inconsistencies and gaps in the selected Cessna 172S anomalous flights. This multivariate time-series data needs to be cleaned in order to make sure that deep learning models take continuous input vectors. As mentioned above, redundant or irrelevant sensors may add noise to the model that can confuse the model and reduce predictive accuracy [31], [22]. Therefore, our cleaning process specifically focuses on maintaining the integrity of the nine final sensors identified for anomaly detection.

The first step is to identify discrepancies in the absence of columns in the 2258 flights. Although the feature extraction process aims at a standardized set of sensors, the analysis is visualized in the plot Missing Column Count by Sensor showed that the feature Total Fuel was completely missing in two flight files. To address this, the median of the remaining data on the Total Fuel and used on the missing column. Median value imputation should be used to fill gaps in multivariate data because it offers a more robust central tendency measure compared to the mean especially when there are outliers [25]. Whereas the mean is easily affected by the extreme values, like sensor spikes or abnormal flight attitudes the median is immune to such skewness [18]. This approach enables the paper to keep the original size of the sample and retain statistical power without the biases that come with the use of means to substitute [25]

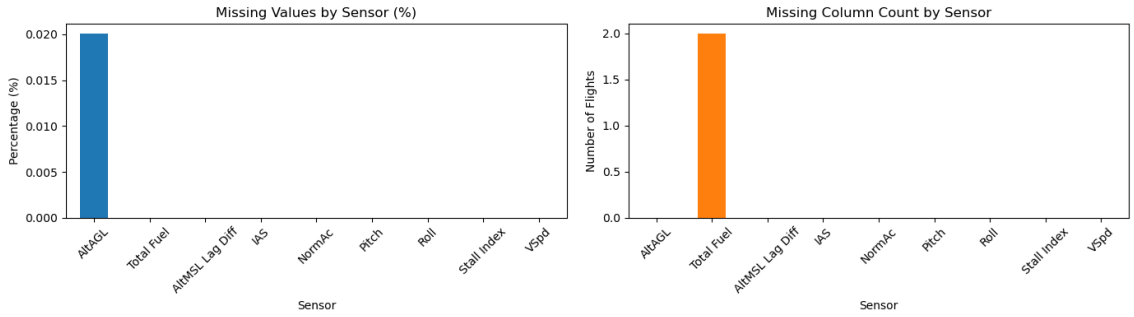

Figure 4.14: Data Inconsistency Statistics

Additionally, there are a few missing value in row level. For example, Altitude Above Ground Level (AltAGL). As illustrated in the Figure 4.14 plot, there are 2,408 missing values in 239 flights which is about 0.02 percent of the total rows. In these time lapses, linear interpolation is used instead of median imputation. Statistically filling row-level gaps in a time series with a constant median is not optimal as it overlooks [18]. Such static values could cause sudden discontinuities or "jumps" between the last recorded row and the imputed segment, which a deep

learning model might incorrectly flag as a physical anomaly.

In order to preserve the physical reality of the flight trajectory, linear interpolation is used to bridge the known values directly around the gap and approximate the intermediate values along a straight line. This guarantees a seamless movement which is a true representation of the flow of aircrafts. Linear interpolation is often regarded as a best practice in dealing with the missing values of time-series sensor data because it maintains the chronological integrity of the sequence and has been demonstrated to yield much lower error rates than the simple mean or median substitution [18].

4.3.3 Flight-Level Event-Stratified Data Partitioning

A rigorous and reproducible data partitioning strategy is fundamental to any machine learning and deep learning. But it becomes complex in the context of aviation time-series data. Random partitioning of the time-series rows of a flight into training and test would be data leakage, as the model would see subsequent time steps of the same flight in training that it would be tested on later, giving the model artificially high and false performance metrics. In order to overcome these issues, this paper uses a two-stage partitioning approach. The first stage used the multi-class anomaly detection pipeline, and the second stage used a greedy optimization algorithm designed to work with multi-label classification. Each of the two stages provides an 80/10/10 train-validation-test split at the flight level, such that all time-series rows of a flight are allocated to a single partition.

4.3.3.1 Stage 1: Event-Stratified Flight Splitting for Anomaly Detection

The preprocessed Cessna 172S data is too large to be stored in one file, so it was previously stored in 15 chunks of CSVs. The initial phase of the partitioning process was to read and concatenate all the chunk files into a single dataframe, which would give full visibility of all the flights and anomaly events prior to deciding on the process of global splitting. Such a unified perspective is necessary since the data split logic should explain the global distribution of rare events that cannot be made right when only a portion of the data is available at any one point in time.

Once the entire data was loaded, the algorithm analyzed the distribution of anomaly events at the flight level as opposed to the row level. Even though deep learning models are learned on continuous time-series windows, which consist of individual rows, the splitting has to be the flight of a single aircraft in continuous trajectory between takeoff and landing. This analysis only factored anomalous rows because normal rows do not hold any information about events. The algorithm calculated the total number of anomalous rows and the number of unique flights with a specific type of event in it, which gives a good overview of the number of independent, non-overlapping flight trajectories per type of event of interest.

A practical implication of the real-world aviation data is the presence of a multi-label complication which means one flight can be the source of multiple types of

anomaly. An example of this is that a single flight may indicate a High Altitude Stall and a Low Pitch event. When a flight such as this is counted in both classes during stratification, this would cause double-counting and make the mathematical split ratios unstable. To fix this, every flight had one main event, which was calculated by finding the type of anomaly that explained the highest number of rows in that flight. The idxmax-based assignment looks after each flight mapping to a single event class to perform stratified partitioning, but leaves the actual multi-label labels in the underlying data to be used downstream.

The essence of the Stage 1 partitioning logic works on an event-by-event basis in contrast to working on the entire dataset at any given time. On each of the 10 anomaly classes, the algorithm first determined the total number of flights allocated to that anomaly, and then used the target ratios to allocate the number of flights to be allocated to training (80%), validation (10%), and test (10%) of that event-specific number of flights. More importantly, the allocation imposes a tough minimum of one flight per split on each event class, so even the most unusual anomalies like High Pitch that occurred only two times in qualification are represented both in train and test. In the case of an event where there were only two flights, the algorithm directly skipped the normal ratio computation and hard coded one flight as training and the other one as test, favoring test coverage instead of validation where data is of vital scarcity. The list of flight IDs per event class was shuffled randomly with a permutation operation before each flight was assigned to a particular split, avoiding implicit ordering bias like earlier flights which might be part of a particular pilot cohort or training period, being overrepresented in the training set.

After assigning all flights, the algorithm retrieved the respective time-series rows of the integrated dataset and built the final train, validation, and test dataframes. A verification test was then performed to ensure that no split had zero examples of any event type, and that there were not more rows in the test set of any event type than in the training set two failure conditions that would interfere with model training and evaluation. The resulting division, which is summarized in Table 4.8, obtained the distribution of 1,800 training flights (79.7%), 224 validation flights (9.9%), and 234 test flights (10.4%) in a total of 2,258 anomalous flights.

Table 4.8: Flight-level event-stratified split for Stage 1 anomaly detection.

Event	**Total Flights**	**Train**	**Val**	**Test**
High Altitude Spin	11	8	1	2
High Altitude Stall	922	737	92	93
High Pitch	2	1	0	1
Low Airspeed on Approach	164	131	16	17
Low Airspeed on Climbout	969	775	96	98
Low Altitude Stall	32	25	3	4
Low Fuel	5	3	1	1
Low Pitch	6	4	1	1
Roll	16	12	1	3
VSI on Final	131	104	13	14
Total	**2,258**	**1,800**	**224**	**234**

4.3.3.2 Stage 2: Greedy Optimization-Based Splitting for Multi-Label Classification

The Stage 2 partitioning problem is much more complex compared to Stage 1. At Stage 1, the flight had been cut down to one main event and therefore stratification was not a problem. In the multi-label classification pipeline, the model has to learn to predict all types of anomalies that co-occur. The flight cannot be classed in a single class; it has to retain all of its labels. Due to this, a flight is transferred to a split also transfers several labels of event type simultaneously. This prevents the possibility of balancing each class separately. This leads to the problem being a combinatorial optimization problem that cannot be solved using standard stratified splitting methods.

Just like in Stage 1, the multi-label data was saved in blocks. These chunks were loaded and then combined into a single dataframe. Subsequently, the algorithm generated a mapping of each flight, in which it noted the precise number of rows in each of the 10 types of anomalies in that flight. Simultaneously, it also estimated the total number of rows per type of event in the entire dataset. These two structures combined to create a clear mathematical perspective: given any flight, the algorithm was aware of the number of rows of each event class that would be included in a split in case that flight was allocated there. It is this detailed tracking which makes the greedy optimization possible.

Then, each event class was assigned target ratios of 80 percent training, 10 percent validation and 10 percent testing. The algorithm then computed a weight of the rarity of each type of event based on the inverse of the total number of rows. This implies that rare events were allocated more weights. This will help in prioritizing rare anomalies when allocating them. The flights were then rated according to the rarest event in them. Flights with the most unusual types of anomalies were put at the top of the list. This rarity-first ordering is important. When common events such as High Altitude Stall are spread initially, then rare events such as High Pitch could be concentrated in a single split, which would not be possible to evaluate properly.

The algorithm went through the flights one by one. In every flight it computed a score in all the three splits: training, validation and test. This was scored on three items: distance each split is to its target, the number of rows that the flight adds to the event, and the weight of the event in terms of rarity. The flight was assigned to whichever split produced the highest score that is, the split that would benefit most from receiving those specific multi-label rows given its current state. Then the algorithm transferred to the next flight. The most important characteristic of the greedy method is this step-by-step updating. For instance, in a case where the test split already has excessive Low Pitch rows the scoring will have less likelihood of receiving additional Low Pitch flights and instead redirect them to training or validation.

As this multi-label problem is not solvable mathematically (flights can never be partitioned and every flight has several labels), the final ratios are approximate only to 80/10/10 target. The results show that the greedy algorithm achieved distributions very close to the target for all 10 event classes. Notably, no split lacked any classes. The largest disparities are observed in uncommon categories such as High Pitch (Train: 68.98%, Val: 19.25%, Test: 11.76%) and Low Fuel (Train: 80.97%, Val: 6.53%, Test: 12.50%). Balancing them precisely is challenging since it has very low number of flights. All classes exist in all splits, so there is sufficient model training and assessment.

Table 4.9: Row distribution per event class of Stage 2.

Event	**Total Rows**	**Train Rows**	**Val Rows**	**Test Rows**
High Altitude Stall	15,095	12,032	1,512	1,551
Low Airspeed on Climbout	12,768	10,198	1,292	1,278
Low Airspeed on Approach	2,220	1,777	223	220
VSI on Final	1,310	1,048	131	131
Roll	1,103	860	137	106
Low Pitch	1,656	1,270	212	174
Low Altitude Stall	316	250	32	34
High Altitude Spin	1,699	1,295	206	198
High Pitch	187	129	36	22
Low Fuel	1,408	1,140	92	176

The last divided datasets of the two stages were stored as fixed CSV files. Both the multi-class and multi-label pipelines are all trained, validated and tested using the same fixed data split. Therefore, the assessment is unbiased and can be replicated to all model architectures.

4.3.4 Data Transformation

The last step of preparation is to transform the partitioned data into a format that can be used during the training of the neural network. Although the specifics of implementing these changes are elaborated in Section 4.2 (Model Specification) as belonging to the model input layer, calculating their parameters is a significant process at the data level.

The 9 selected multivariate sensors are normalized using the Z-score (standardization). In the process, the features undergo centering by calculating the difference between the feature and the mean and scaling by the standard deviation. This is required since the aviation sensor data have very different ranges, such as Altitude is in thousands of feet, and Pitch is in small degree values. In order to adhere to the clean training principle and prevent statistical leakage, the mean and the standard deviation are only computed using the training set and applied to the validation and the test sets.

In the multi-label classification task, multi-hot encoding is used to convert the target anomaly labels. Here, every single timestep is represented as a 10-bit binary number, with the presence of each anomaly represented by a 1 at that spot. This enables the model to identify several overlapping events simultaneously, like a "High Altitude Stall" event that is combined with an over-turning event of excessive "Roll.

4.3.5 Temporal Segmentation and Sequential Windowing

Raw multivariate flight sensor data cannot be directly used in deep learning models in its raw form. Prior to the training, the time-series data should be transformed, a process called temporal segmentation or sliding window construction, into fixed length sequential segments. All the windows are model inputs, denoted as a tensor of size [TxF], where T is the time steps, and F is the features. During this step, the decisions taken on window size and stride. The calculation of evaluation metrics and the similarity of the testing setup to the deployment in the real world. The size of the window T is used to determine the number of consecutive time steps in a single input sequence. The value of T used in this research was 30 which is a 30-second window as the NGAFID sensor data is sampled at 1 Hz. This decision was made considering both the knowledge in the domain and the available literature.

The anomaly events that were the focus of this work like the stall onset, excessive rolling, low approach airspeed, and abrupt deviations in pitch, do not occur immediately. These occurrences are built up with time by varying parameters such as indicated airspeed, pitch angle, roll angle, and stall index. A 30-second window with a 1 Hz sampling rate gives sufficient time history of how these parameters vary before and during an anomaly to the model. Meanwhile, it is not too long to retain temporal locality in order to make correct predictions per second. When the window is excessively long, the anomaly signal may be diluted in an excess of normal data. Conversely, when the window is too short, the model might not acquire sufficient context to make the right decisions between actual anomalies and short-term sensor noise.

The selection of T=30 also reflects a deliberate balance between model complexity and computational efficiency. For Transformer-based architectures in particular, input sequence length has a quadratic effect on the computational cost of self-attention, meaning that unnecessarily long windows inflate training time and memory requirements substantially without a commensurate gain in detection performance. This trade-off has been noted explicitly in the time-series anomaly detection literature: Tuli et al. [16] observed in their analysis of TranAD that when windows are too small, they fail to capture local contextual information adequately, while excessively large windows can obscure short-duration anomalies by embedding them within a large number of surrounding normal data points, leading to a measurable drop in F1 score [16]. Their analysis highlighted the importance of selecting a window size that strikes a balance between contextual richness and anomaly resolution. Separately, in a study evaluating GAN-based anomaly detection across 75 real-world time-series datasets, [39] found that a window size of 30×2=60 steps with a stride of 1 produced near-optimal results, and consistently used a shift length of 1 across both training and testing to capture the relevant temporal dynamics of the datasets [39]. While their effective window was larger, the underlying principle that window sizes in the range of 30 to 60 time steps, combined with dense stride-1 sliding, are well-suited for capturing anomaly-relevant temporal patterns is consistent with the design adopted in this work. Taken together, domain analysis and empirical evidence from the literature converge on T=30 as an appropriate and well-grounded choice for the aviation sensor data considered here.

Training, validation and test set were not constructed similarly in the windows construction process. The splits were treated differently based on the role that they played, the primary difference factor being stride.

In the training set, a stride of 1 was uesd it is also referred to as a fully overlapping or dense sliding window. Here, each window slides forward by a time step. As an example, the rows of Window 1 are 1-30, Window 2 are 2-31, Window 3 are 3-32 and so on. The reason behind this approach is that it is essential to have varied training data in order to have good model generalization. Using stride 1, each row is displayed up to T=30 times, with each display in a new position in the sequence occasionally at the start, occasionally in the middle and occasionally at the end. This enables the model to acquire temporal transitions rather than learning fixed patterns. It also makes the training dataset size effective, which is very important as anomalies are very rare in this dataset (only 0.17% of total rows).

For the validation set, a stride of 1 was also used. The primary goal of the validation set is to find the best classification threshold. A prediction should be made to each and every single row so that the Precision-Recall curve can be computed correctly. With a bigger stride, certain rows would be skipped and would not be predicted, leading to incomplete evaluation and possible misinterpretation of a threshold. The stride 1 is used to make sure that all rows are covered so that the correct metrics such as F1-score can be computed.

In the test set, Stride 1 windowing was applied likewise. The reason here is to be in line with the deployment conditions in the real world. The model is continuously used in a real-time monitoring system. It does not work on a single 30-second window and then move forward by 30 seconds. Rather, when a new sensor value is received the window shifts one time step forward and the model generates a new prediction. This is a continuous prediction process simulated by stride 1 in the test set. The assessment is based on the real-time performance of the model as opposed to providing a highly optimistic assessment which may occur with non-overlapping windows.

In all three splits, the label of each window is the label of the final time-step of the window, i.e. the last observation on the 30-second sequence. The model takes the last 30 seconds of the information to forecast the present. This configuration is similar to a real-world system that would be used in dealing with continuous streaming sensor data.

4.3.6 Summary of Data Preprocessing

The preprocessing pipeline of the data was developed in a systematic manner such that the high-dimensional flight data is converted into the deep learning models. In the data reduction step, only the Cessna 172S aircraft was selected since it contributed more than 95% of the anomaly events, and only those flights that contained at least one anomaly were kept, which helped reduce the class imbalance from 1:603 to 1:337. Subsequently, during feature extraction, the original 76 sensor features were narrowed down to 9 significant sensors like Pitch, Roll, IAS and Stall Index with the help of Random Forest feature importance and correlation analysis to eliminate redundant and confusing data. Moreover, in data cleaning, missing values were treated with caution through median imputation of column-level missing values and linear interpolation of row-level gaps to ensure realistic time-series behavior. Then, during data transformation, Z-score normalization was performed to normalize all the features appropriately and multi-hot encoding of the target labels was performed to allow multi-label classification. In temporal segmentation, the continuous flight data was segmented into sliding windows (T = 30) of time with a stride of 1 which not only provided the model the opportunity to learn successfully on the data, but it also made it possible to predict anomalies per second.To avoid data leakage, 80/10/10 split was carried out at the flight level instead of the row level. This included event-stratified division of single-label detection and greedy optimization technique of multi-label classification.

Table 4.10: Finalized preprocessed dataset used for model training.

Metric	**Initial GATS (Cessna 172S)**	**Final Preprocessed Dataset**
Total Flights	4,481	2,258
Total Rows	22,106,847	11,963,999
Anomaly Classes	18 Observed	10 (Most Frequent)
Imbalance Ratio	1:603	1:337

Stage 1: Event-Stratified Split (Multi-Class)

This stage utilized idxmax-based assignment, where each flight was categorized by its most frequent anomaly type to ensure stable stratification.

Table 4.11: Distribution of flights across training, validation, and test sets for Stage 1.

Split	**Number of Flights**	**Percentage**
Training	1,800	79.7%
Validation	224	9.9%
Test	234	10.4%
Total	**2,258**	**100%**

Stage 2: Greedy Multi-Label Row Partitioning

Stage 2 partitioning deals with the complexity of multi-label flight data, in which one flight may have multiple anomalies and cannot be partitioned without leaking data. To address this, a greedy rarity-first algorithm was employed. The flights with rare events received priority and each assignment was provided by a dynamic scoring system that verified the distance of each split (train/validation/test) to the desired 80/10/10 ratio. Each assignment was self-corrected in real time by updating the split distributions continuously by the algorithm. All the anomaly classes were fairly represented, small imbalances were left in very rare events because of the small number of available flights.

Table 4.12: Distribution of anomaly classes across data partitions using greedy multi-label stratification.

Event Name	**Train %**	**Val %**	**Test %**
High Altitude Stall	79.71%	10.02%	10.27%
Low Airspeed on Climbout	79.87%	10.12%	10.01%
Low Airspeed on Approach	80.05%	10.05%	9.91%
VSI on Final	80.00%	10.00%	10.00%
Roll	77.97%	12.42%	9.61%
Low Pitch	76.69%	12.80%	10.51%
Low Altitude Stall	79.11%	10.13%	10.76%
High Altitude Spin	76.22%	12.12%	11.65%
High Pitch	68.98%	19.25%	11.76%
Low Fuel	80.97%	6.53%	12.50%

4.4 Implementation of Selected Design

This section describes how the two-stage anomaly detection and classification system is implemented in a real-time operational setting onboard a general aviation aircraft. The pipeline is designed to process raw sensor data continuously as it arrives, make near-instantaneous decisions at each second, and deliver actionable alerts to the cockpit without requiring any external preprocessing or cloud connectivity. The full system flow is illustrated in Figure 4.15.

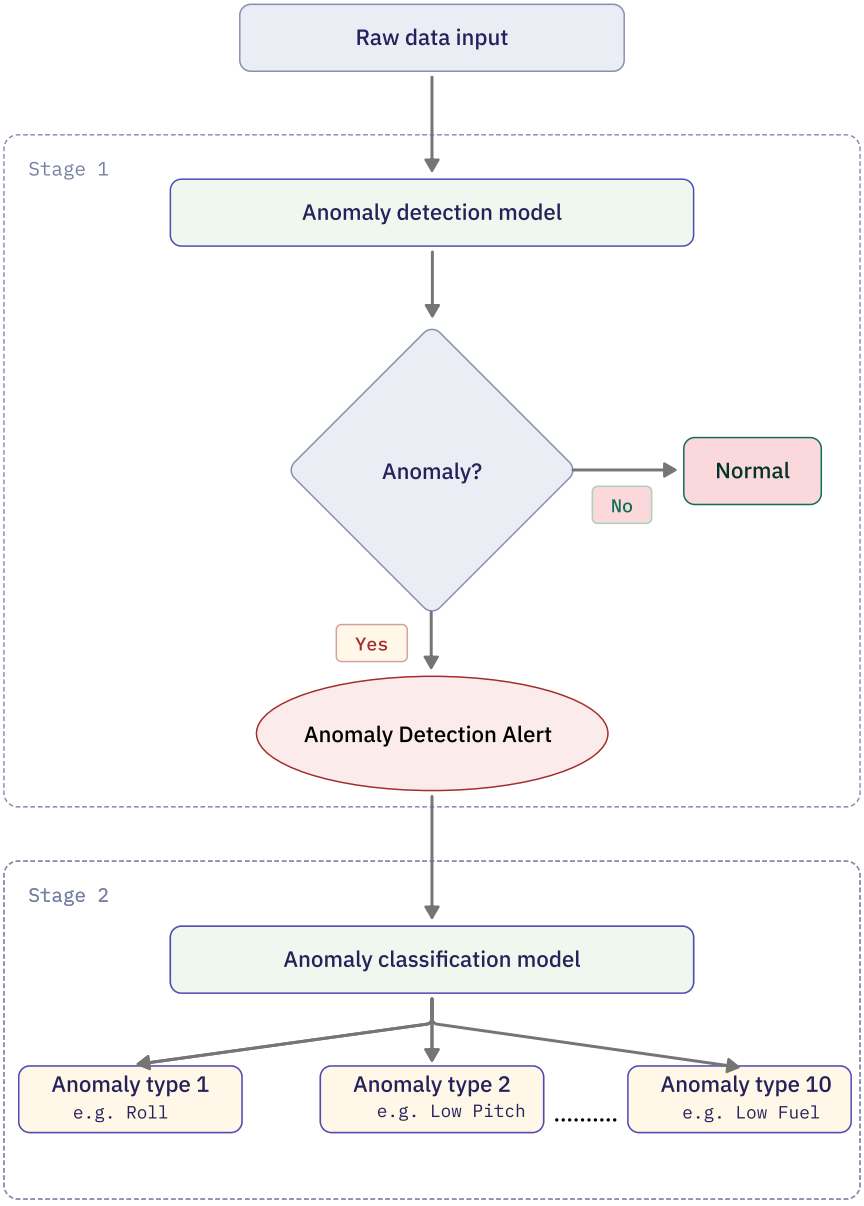

Figure 4.15: Real-time inference pipeline.

4.4.1 Data Ingestion and Real-Time Window Construction

In the offline training environment, sliding windows of 30 timesteps were extracted in advance from pre-collected flight records. In a deployed system, sensor data arrives one row at a time at 1 Hz one complete set of nine sensor readings every second. To replicate the same 30-timestep input that the Stage 1 models expect, a circular buffer (also known as a ring buffer) is maintained in memory at all times.

A circular buffer is a fixed-size memory structure that always holds the most recent 30 sensor readings. When a new sensor row arrives, it is appended to the buffer and the oldest row is automatically discarded. The buffer therefore behaves as a continuously sliding window over the live sensor stream:

```
Second 1: [s]
Second 2: [s, s]

Second 30: [s, s, ..., s] ← first complete window: run Stage 1
Second 31: [s, s, ..., s] ← s removed, s added: run Stage 1 again
Second 32: [s, s, ..., s] ← and so on each second ...
```

From the second 30 onward, a fresh 30-timestep window is available every second and is immediately forwarded to the Stage 1 ensemble. This matches the stride=1 inference mode used during offline evaluation, meaning the deployed system produces exactly one prediction per second the same cadence at which sensor data arrives.
Each complete window (shape $1 \times 9 \times 30$) is passed to both the CSPT and HiCST models simultaneously. Each model applies its internal SensorNorm layer to normalize the raw sensor values, runs the full forward pass, and outputs an anomaly probability in [0, 1]. The two probabilities are fused using the RMS ensemble formula described in Section 4.2.1.3:

$$p_{\text{ens}} = \sqrt{\frac{p_{\text{CSPT}}^2 + p_{\text{HiCST}}^2}{2}} \tag{4.45}$$

The ensemble probability p_{ens} is then compared against the pre-tuned optimal threshold τ_{opt}, which was determined offline on the validation set and is stored as a fixed configuration value in the deployed system. No real-time threshold computation is required. The decision rule is straightforward:
If $p_{\text{ens}} \geq \tau_{\text{opt}}$: the window is flagged as anomalous. A Stage 1 alert is raised and the pipeline proceeds to Stage 2.
If $p_{\text{ens}} < \tau_{\text{opt}}$: the window is classified as normal. No alert is generated, and the system waits for the next sensor reading.
When Stage 1 flags an anomalous window, the system immediately passes the current timestep's nine raw sensor values (shape 1×9) to the Stage 2 model, MHANet. This is simply the last row of the circular buffer the most recent sensor reading. No additional windowing or preprocessing is required, because MHANet is designed to classify anomaly types from a single timestep.

The output of Stage 2 is a classified anomaly report: a list of active event types at the current second, along with their probabilities. This report is immediately available for display in the cockpit and for logging.

4.4.2 Event Duration Tracking and Alert Management

A naive implementation that sends an alert every second for the duration of an anomaly would flood the cockpit with repeated notifications, distracting the pilot from addressing the actual safety condition. A simple state machine is therefore maintained alongside the inference pipeline to manage alert delivery intelligently. The system operates in one of two states at any given time: Normal or Active Event.

Event onset (Normal → Active Event). When the system transitions from Normal to Active Event for the first time i.e., Stage 1 flags an anomaly after a period of normal operation, a single alert is issued to the cockpit (visual indicator and audio notification), and a start timestamp is recorded. The classified event type(s) from Stage 2 are displayed on the pilot's screen.

Event continuation (Active Event remains). While the system remains in the Active Event state in subsequent seconds, Stage 1 and Stage 2 continue to run every second and the on-screen display is updated if the active event type changes. However, the audio alert is suppressed: the pilot has already been notified, and repeated audio alarms during an ongoing event would be disruptive and unhelpful. The visual indicator remains active to confirm the event is still in progress.

Event resolution (Active Event → Normal). When the ensemble probability drops below τ_{opt} for a sustained period, the system transitions back to Normal. An end timestamp is recorded, the visual alert is cleared, and a complete event log entry is written: event type, start time, end time, and duration. This log is stored in the flight data recorder (FDR) and is available for post-flight review by maintenance teams and safety investigators.

The event log format is structured as follows:

Table 4.13: Example event log entries showing event type, timestamps, duration, and classified sub-events.

Event Type	Start Time	End Time	Duration (s)	Detected Events
Anomaly	10:05:00	10:05:15	15	High Pitch
Anomaly	10:12:43	10:12:47	4	Roll, Low Pitch

4.4.3 Design Properties Supporting Real-World Deployment

Several design decisions made during model development directly contribute to the practical deployability of the system in real aviation hardware.

Internal normalization. Both Stage 1 models (CSPT and HiCST) and Stage 2 (MHANet) include a SensorNorm layer that applies the training-set mean and standard deviation internally. Raw sensor readings can be passed directly to the models without any external preprocessing pipeline. The normalization statistics are stored as part of the model weights and are deployed alongside them, eliminating

the risk of a mismatch between training-time and deployment-time preprocessing.

Trigger-based Stage 2 (lazy evaluation). In normal flight, approximately 99.7% of timesteps are anomaly-free. MHANet is only invoked when Stage 1 raises an alert. This means Stage 2 is idle for the vast majority of flight time, consuming no computational resources. This trigger-based design is consistent with industry practice for embedded monitoring systems, where power and processing budgets are constrained.

Lightweight architecture. Both Stage 1 models use d_model = 64, three Transformer blocks, and operate on 30-timestep windows of 9 sensors a small input by deep learning standards. Stage 2 processes a single 9-dimensional timestep. Single-sample inference for both stages requires only a few milliseconds on CPU, making the system compatible with avionics-grade embedded hardware without requiring a GPU.

No external dependencies at inference time. The system requires no network connectivity, no API calls, and no external databases during inference. All model parameters, normalization statistics, and decision thresholds are stored locally on the device. The system is fully self-contained and can operate in environments where connectivity is unavailable or unreliable.

Pre-tuned fixed thresholds. Both the Stage 1 ensemble threshold (τ_{opt}) and the 10 per-class Stage 2 thresholds were determined offline on the validation set and are deployed as fixed configuration values. No real-time threshold computation or model adaptation is required during flight, further simplifying the deployment pipeline.

Robustness to real-world imbalance. In operational conditions, the ratio of normal to anomalous flight is even more skewed than in the training data. The use of Focal Loss in Stage 1 and Asymmetric Loss in Stage 2 means both models were explicitly trained to handle severe class imbalance. The models are calibrated to minimize false positives avoiding spurious alerts that would desensitize pilots to genuine warnings while maintaining high sensitivity to true anomalies.

4.4.4 Application of Feedback Mechanisms for Continuous Enhancement

The current two-stage anomaly detection architecture operates primarily as an open loop system, where Stage 1 classifies flight windows as anomalous or normal, and Stage 2 identifies specific event types, with pilot notification and decision-making forming the final output without intrinsic model feedback correction. However, operational deployment benefits substantially from closed loop feedback mechanisms that enable continuous model improvement and adaptive decision thresholds. In a closed loop implementation, pilot confirmations of alerts (alert relevance: true positive, false positive, or missed detection) are captured and fedback to the system, enabling model retraining pipelines that adapt to aircraft-specific operational

patterns and evolving pilot behaviors. Additionally, closed loop feedback facilitates threshold adaptation—if operational data reveals systematically different optimal thresholds than laboratory validation, thresholds can be dynamically adjusted without full model retraining. Real-time model monitoring capturing inference outputs and outcomes enables detection of performance degradation (covariate shift, concept drift) triggering automatic retraining or fallback to previous validated versions. The hybrid approach maintains open loop safety guarantees (deterministic, validated outputs) while leveraging closed loop capabilities for incremental performance enhancement, providing both operational reliability and continuous improvement—critical for safety-critical aviation systems where model integrity must be maintained while operational effectiveness is continuously optimized through user feedback and outcome data.

Chapter 5

Result Analysis

5.1 Evaluation Metrics and Mathematical Formulation

5.1.1 Stage 1: Binary Anomaly Detection

For this stage, the test set contains 1,228,570 normal windows and 3,626 anomalous windows, resulting in a class imbalance ratio of approximately 339:1. When working with such a significant class imbalance, the accuracy score can become a misleading indicator of performance. Even a "dummy classifier" which identifies all windows as normal would still achieve 99.7% accuracy. That is why the performance measure for this stage only considers finding metrics that focus on finding anomalies, namely precision, recall, F1-score, and Area Under the Precision-Recall curve (AUPRC).

The four metrics mentioned above are computed using four different measures of results. Here, True Positive (TP) and True Negative (TN) occur when a window containing a real anomaly and no anomaly is identified, respectively. On the contrary, False Positive (FP) and False Negative (FN) occur when a window containing no real anomaly is identified as one that contains an anomaly and vice versa, respectively. Among these four, False Negative (FN) is most critical to safety, as an anomaly goes undetected and does not reach Stage 2.

Precision: This measures of all the "Positives" the model is flagging how many are truly Positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.1}$$

A low value for precision means that the model flags too many normal windows as anomalous. In a real-life aviation scenario, this would cause too many false alarms, which would in turn reduce the confidence flight operators place on the system. So, a high precision is required to retain the confidence of the flight operators.

Recall: Measures of all the anomalous windows, how many the model has been able to identify.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.2}$$

Recall is the most important metric for Stage 1. This is because when a False Negative (FN) happens, it means an anomalous window gets discarded and never reaches Stage 2 in the first place. In aviation, this can be the difference between life and death. Therefore, it is better to have more false alarms (low Precision) and to have no alarm for an actual anomaly (low Recall).

F1-Score: F1-score is calculated by taking the harmonic mean of Precision and Recall.

$$F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{5.3}$$

Optimizing recall alone would allow a model to achieve 100% recall by predicting every window as anomalous, but precision would collapse to near zero. The F1-score prevents this by penalizing any large imbalance between the two. The harmonic mean is used rather than the arithmetic mean because it weights the lower of the two values more heavily, making it a stricter and more meaningful combined measure.

Area Under the Precision-Recall Curve (AUPRC). Rather than evaluating performance at a single threshold, the AUPRC summarizes the Precision-Recall trade-off across all possible thresholds:

$$\text{AUPRC} = \int_0^1 \text{Precision}(r)\, dr \tag{5.4}$$

AUPRC is more appropriate compared to ROC-AUC, which can produce overly optimistic values for negative classes when working with datasets that have such massive imbalances as the one we are working on. This is because even poor-performing models can accumulate many True Negatives (TNs). AUPRC disregards TNs completely and evaluates based on the model's ability to identify the minority class (anomalous windows in this case) correctly. The higher the AUPRC score, the higher the model can maintain good quality anomaly detection, irrespective of where the decision threshold is set.

5.1.2 Stage 2: Multi-Label Event Classification

This stage identifies one or multiple (up to 10) anomalies that are passed on to it from Stage 1. One of its specialties is that it can simultaneously identify multiple active events. To achieve this, it treats each event as an independent binary output. For each sample timestep i, the true label vector is $y_i \in \{0,1\}^{10}$ and the predicted label vector is $\hat{y}_i \in \{0,1\}^{10}$. Instead of directly applying standard single-label metrics, the following multi-label metrics are used.

Macro F1-Score: The F1-score has been computed separately for each of the $L = 10$ event classes, and then the unweighted average has been taken.

$$\text{Macro F1} = \frac{1}{L} \times \sum_l F1_l = \frac{1}{L} \times \sum_l \left[\frac{2 \cdot P_l \cdot R_l}{P_l + R_l} \right] \tag{5.5}$$

For this stage, Macro F1 is the primary and most significant metric as it treats every class equally, regardless of their sample counts. As a result, rare events, such

as High Altitude Spin, get the same weight as more common events, such as High Altitude Stalls. So, if the model performs poorly on rare events, then it directly reflects in the overall Marco F1 score as it gets significantly lowered.

Micro F1-Score: This score gives proportionate weight to each class according to its sample counts.

$$\text{Micro F1} = \frac{2 \times \sum_l TP_l}{\sum_l (2 \cdot TP_l + FP_l + FN_l)} \tag{5.6}$$

Unlike the Macro F1 score, the Micro F1 score is significantly affected by the sample count of each class. Therefore, Micro F1 gives an aggregated view of the throughput of the system. In other words, it represents how many labels are being predicted correctly. While the Micro F1 score alone can give a misleading view of the correctness of the model for classifying rare events, it is really useful when measured alongside the Macro F1 score. A small delta between the two scores indicates that the model is performing almost equally and consistently well for rare events too.

Subset Accuracy (Exact Match Ratio). An inference from the model is counted by this metric as correct if and only if the inferred vector completely matches the true label vector.

$$\text{Subset Accuracy} = \frac{1}{N} \times \sum_i \mathbb{I}[\hat{y}_i = y_i] \tag{5.7}$$

Among the four metrics being used for Stage 2, subset accuracy is the strictest. For multi-label scenarios like ours, even a single incorrectly predicted label out of 10 labels can cause the entire sample to be counted as incorrect. However, in aviation, where a slight deviation can make the difference between life and death, partially correct event classification can give the aircraft operator(s) only a partial picture of the situation in a critical moment when every second counts. For example, if the system is able to detect a Roll event but not simultaneously able to detect a Low Pitch event, the flight operator(s) are not going to have the full picture. Therefore, the higher the score for this metric, the better the system would be able to provide a "full picture" of the situation.

Hamming Loss. Hamming loss measures the fraction of individual label decisions that are incorrect across all samples and all classes:

$$\text{Hamming Loss} = \frac{1}{N \cdot L} \times \sum_i \sum_l \mathbb{I}[\hat{y}_{i,l} \neq y_{i,l}] \tag{5.8}$$

While subset accuracy asks whether every label is correct, Hamming loss quantifies the average label-level error rate. A Hamming loss of 0.05 means that, on average, 5% of the 10 individual event predictions per timestep are wrong. Hamming loss and subset accuracy are complementary: subset accuracy reveals how often the full event picture is correct, while Hamming loss reveals the average magnitude of errors when the full picture is not. Lower Hamming loss is better.

5.2 Analysis of Design Solutions

5.2.1 Model Performance Evaluation

This section presents and evaluates all three models, namely two of the Stage 1 models (CSPT and HiCST), their RMS ensemble, and the Stage 2 classifier (MHANet), developed in this thesis. The evaluation focuses solely on the anomaly class metrics discussed in Section 5.1.1, as the overall accuracy is an irrelevant and uninformative result due to the massive imbalance in the test set (normal windows count = $1,228,570$ and anomalous windows count = $3,626$).

Stage 1: Binary Anomaly Detection

CSPT: The CSPT model completed its training using all 150 specified echops without any early stopping being triggered. This indicates that the model continued to have marginal gains throughout its training. The training loss and validation loss stabilized at 0.0002 and 0.0007, respectively. The highest achieved threshold is 0.88, which is slightly higher compared to HiCST. In the final epoch during training, a learning rate of 1.2×10^{-9} was observed.

Table 5.1: CSPT test set performance on the anomaly class (TN=1,227,864 — FP=706 — FN=306 — TP=3,320).

Model	Precision	Recall	F1	AUPRC
CSPT	0.8246	0.9156	0.8677	0.9419

A significantly higher recall score of 0.9156 was achieved, comparatively higher than HiCST's recall score of 0.8602. This means CSPT was able to detect 3,320 out of 3,626 anomalous windows. It is to be noted that this higher recall score comes at the expense of lower precision at 0.8246. It infers more False Positives (706) in comparison to HiCST (469). However, for usage in aviation, the behavior of CSPT is more favorable. As explained before in Section 5.1.1, it is better to be on the side of caution by having more False Positives (FPs) than to have more False Negatives (FNs). Moreover, the AUPRC score of CSPT is slightly higher than HiCST, being 0.9419 and 0.9407, respectively.

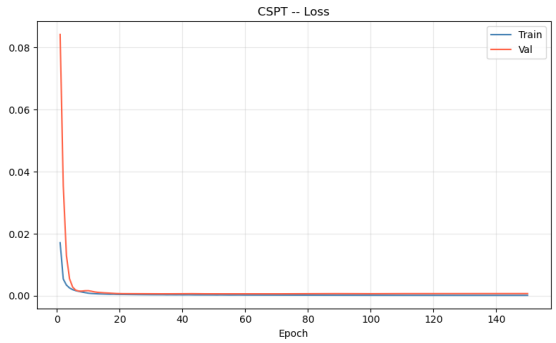

Figure 5.1: Training loss vs validation loss curves for CSPT

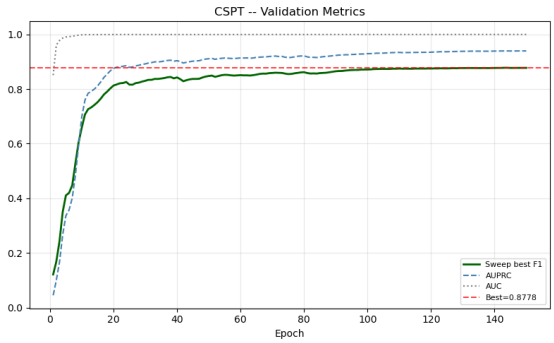

Figure 5.2: Training and validation F1 curves for CSPT

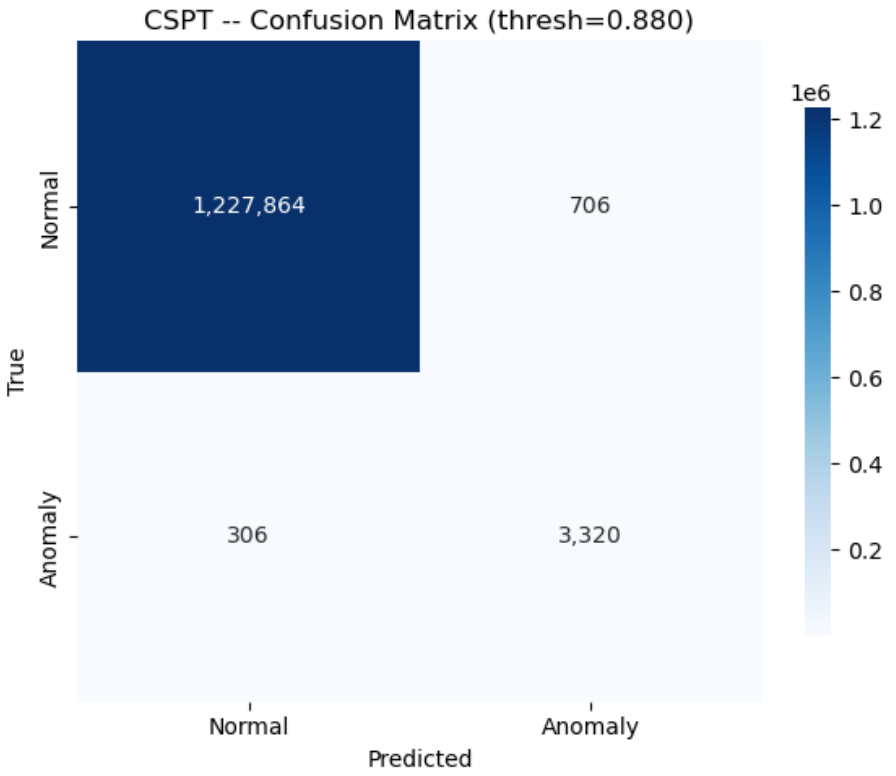

Figure 5.3: Confusion matrix for CSPT

HiCST: In contrast to CSPT, HiCST was able to reach its best validation F1 score at the 52nd epoch when early stopping was triggered, and training was terminated. The training loss and validation loss stabilized at 0.0002 and 0.0006, respectively, with a very narrow delta between the two, indicating the model generalized well without overfitting. The highest achieved threshold is 0.85. The test results are presented in Table 5.2.

Table 5.2: HiCST test set performance on the anomaly class

Model	Precision	Recall	F1	AUPRC
HiCST	0.8693	0.8602	0.8647	0.9407

HiCST achieved a precision of 0.8693 and a recall of 0.8602. The 469 false positives indicate that the model occasionally misclassifies normal windows as anomalous, while the 507 false negatives represent anomalous windows that were missed entirely. The high AUPRC of 0.9407 confirms that the model's ranking of anomalous windows is reliable across all possible thresholds, not just the operating point selected for deployment.

Close observation suggests that the two models (CSPT and HiCST) demonstrate complementary characteristics. On the one hand, CSPT has a higher Recall score. On the other hand, HiCST has a higher Precision score. This difference stems from the architectural differences between them. To elaborate, while CSPT's overlapping patch tokens can capture a much broader range of anomalies and have higher sensitivity, hierarchical compression of HiCST focuses on the temporal regions that are

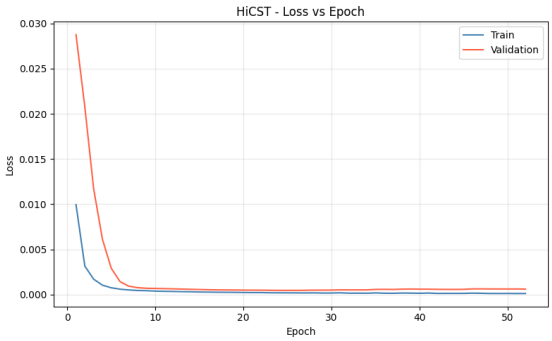

Figure 5.4: HiCST training and validation loss curves

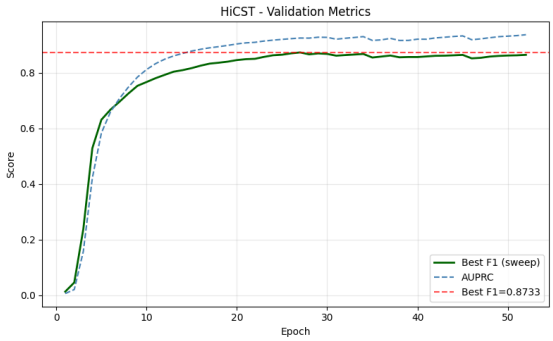

Figure 5.5: HiCST training and validation F1 curves

most activated, making its alerts comparatively more conservative. These complementary characteristics provide the perfect opportunity to combine both of them through ensemble fusion to get the so-called "best of both worlds" results.

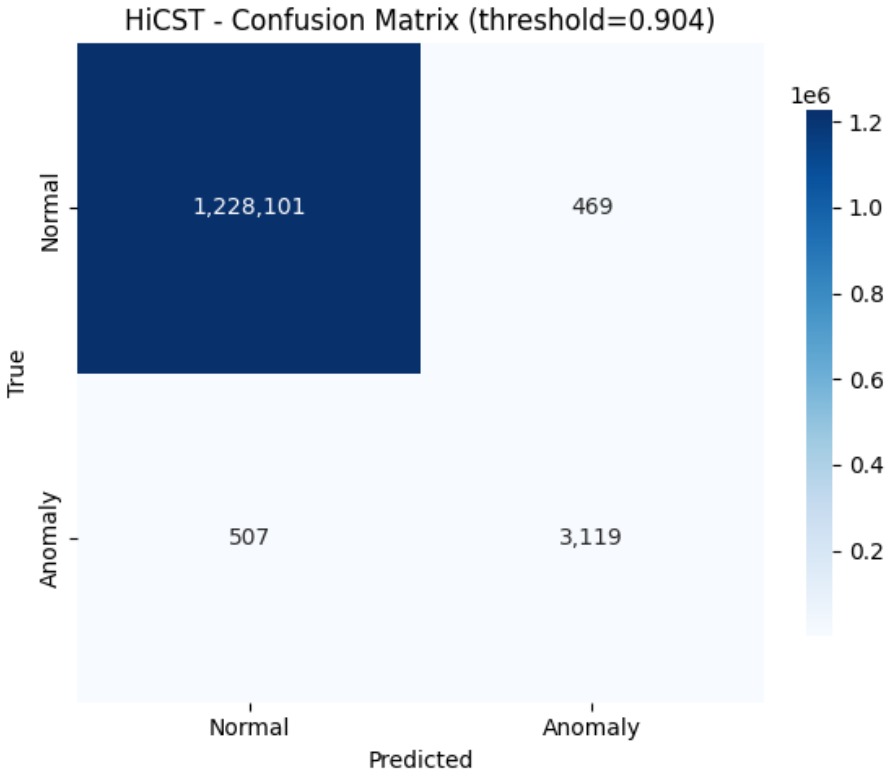

Figure 5.6: HiCST confusion matrix

RMS Ensemble: The RMS ensemble combines the output probabilities of HiCST and CSPT using the formula $p_{\text{ens}} = \sqrt{\frac{p_{\text{CSPT}}^2 + p_{\text{HiCST}}^2}{2}}$. As shown in Figure 5.7, this fusion yields a balanced performance profile that neither model achieves independently.

Table 5.3: Comparison of Stage 1 models on the anomaly class.

Model	Precision	Recall	F1	AUC	AUPRC	Note
HiCST	0.8693	0.8602	0.8647	—	0.9407	High precision
CSPT	0.8246	0.9156	0.8677	—	0.9419	High recall
RMS Ensemble	0.8568	0.9076	**0.8815**	0.9997	**0.9523**	**Best overall**

The ensemble raises precision compared to CSPT alone (0.8568 vs. 0.8246) and raises recall compared to HiCST alone (0.9076 vs. 0.8602). It achieves the highest F1-score of all three models at 0.8815, along with the best AUPRC of 0.9523 and a near-perfect ROC-AUC of 0.9997. The RMS fusion mechanism is well-suited to this task: when one model is strongly confident about an anomaly, RMS amplifies that signal slightly more than a simple average would, while remaining conservative when both models are uncertain. The result is a detector that inherits CSPT's sensitivity and HiCST's specificity, reducing both false negatives (missed anomalies) and false positives (spurious alerts) relative to either individual model.

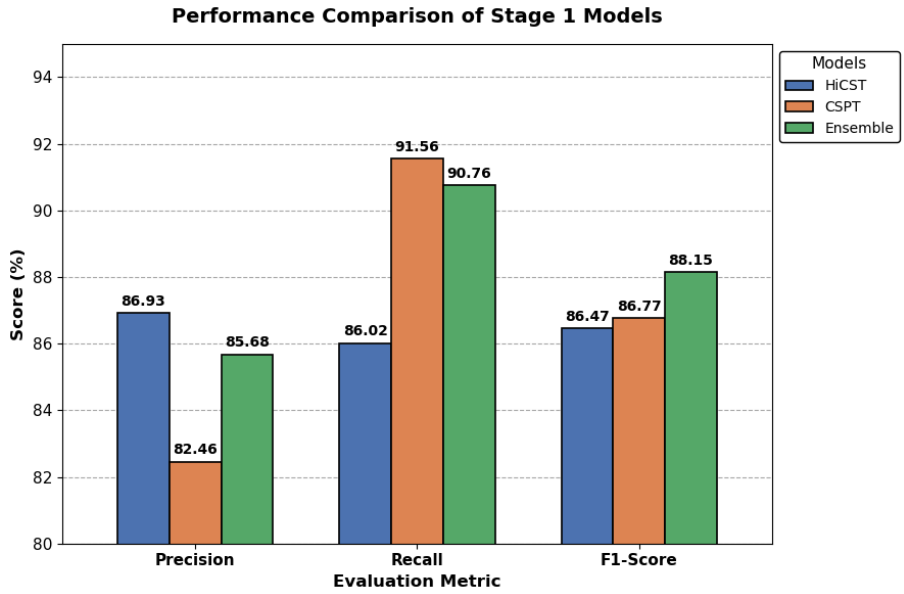

Figure 5.7: Comparison of precision, recall, and F1-score of HiCST, CSPT, and the RMS ensemble.

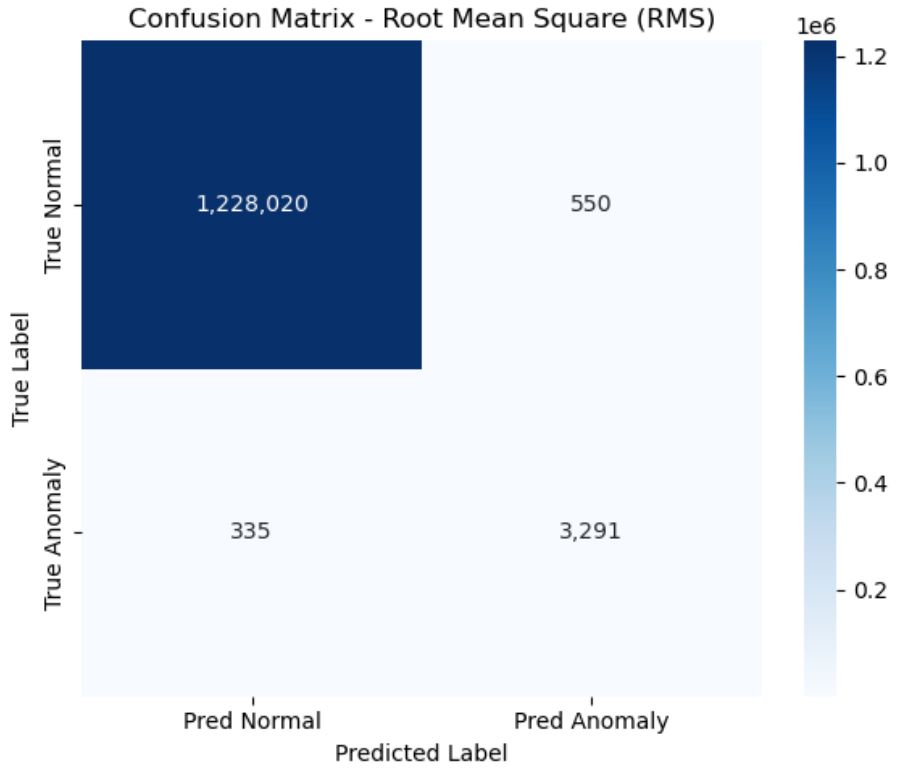

Figure 5.8: RMS ensemble confusion matrix.

Stage 2: Multi-Label Event Classification (MHANet) Stage 2 receives the anomalous timesteps flagged by Stage 1 and assigns them to one or more of the 10 target event classes. The evaluation uses the four multi-label metrics defined in Section 5.1.1: macro F1, micro F1, subset accuracy, and Hamming loss.

Table 5.4: MHANet test set performance across four multi-label evaluation metrics.

Metric	Macro F1	Micro F1	Subset Acc.	Hamming Loss
MHANet	0.9563	0.9771	0.9627	0.00497

MHANet achieves a macro F1 of 0.9563, demonstrating strong and consistent performance across all 10 event classes, including the rarest ones. The micro F1 of 0.9771 confirms that the model also performs well in terms of aggregate label-level accuracy. The macro–micro gap ($0.9771 - 0.9563 = 0.0208$) is small, indicating that the model does not perform dramatically better on frequent classes than on rare ones a particularly important result given that the event class distribution is highly skewed. The subset accuracy of 0.9627 means that the model produces a fully correct multi-hot prediction (all 10 labels simultaneously correct) for 96.27% of test timesteps, which is a demanding standard in a multi-label setting. The Hamming loss of 0.00497 indicates that on average fewer than 0.5% of individual event label predictions per timestep are incorrect.

The per-class breakdown reveals that performance varies across event types, largely in proportion to the number of positive test samples available. For example, High Altitude Stall events (F1 = 0.9894, support = 1,551) and Low Airspeed on Climbout

events (F1 = 0.9929, support = 1,278) achieve the highest scores, as it is to be expected due to their higher Support count. VSI on Final achieves a perfect precision of 1.000, reflecting zero false positives for that class. High Pitch achieves a recall of 1.000 despite having only 22 positive test samples. The lowest-performing class is High Altitude Spin (F1 = 0.8496), which is the rarest event with meaningful support (198 samples) and shows the most confusion with other events, particularly Low Pitch and Roll, due to the physical overlap of sensor signatures in spin and low-pitch conditions. Nevertheless, even for this difficult class, the model achieves a recall of 0.9697, meaning it misses only about 3% of High Altitude Spin occurrences.

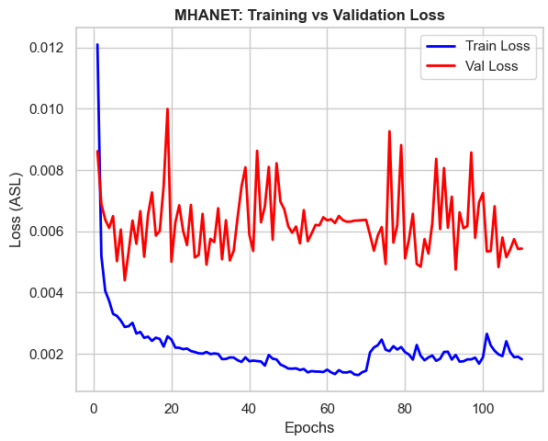

Figure 5.9: MHANet training and validation loss curves.

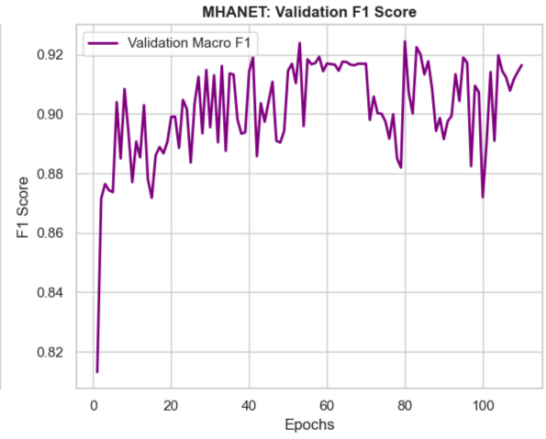

Figure 5.10: MHANet training and validation macro F1 curves.

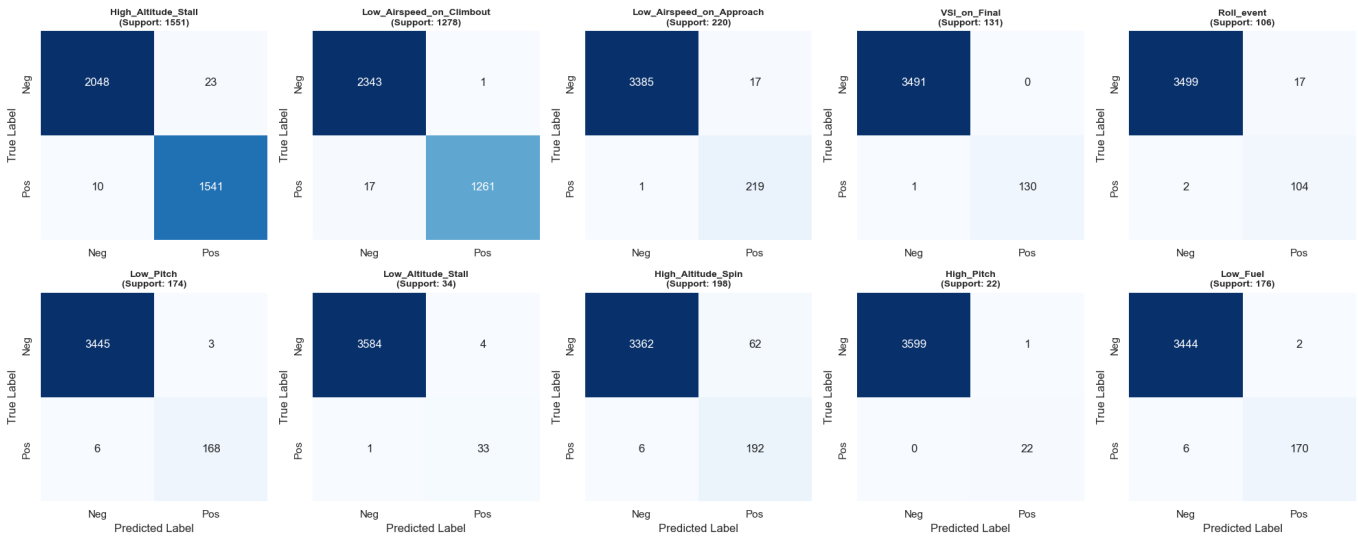

Figure 5.11: Per-Class Binary Confusion Matrices (10 Classes).

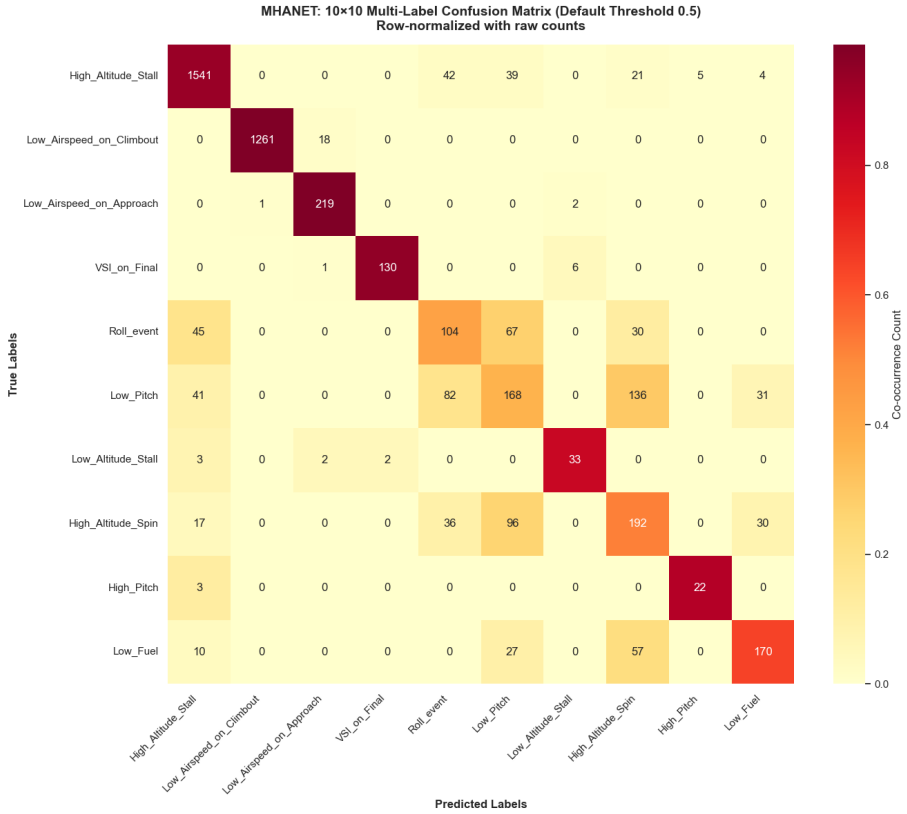

Figure 5.12: MHANet 10 × 10 Multi-Label Co-occurrence Confusion Matrix

The multi-label co-occurrence matrix (Figure 5.12) provides additional insight into the model's error structure. The strong diagonal confirms that most event predictions are correct. The most prominent off-diagonal interactions are between High Altitude Spin, Low Pitch, and Roll: when a High Altitude Spin is occurring, the aircraft typically exhibits a combination of low pitch, rolling motion, and altitude change, causing sensor signatures that partially overlap with those of Low Pitch and Roll events. These indicate that the co-occurrences actually have physical significance and are not an indication of model failure. This is because it is really difficult to separate aerodynamically related phenomena from instantaneous sensor measurement.

5.2.2 Efficiency, Feasibility, Strengths, and Weakness

Computational Efficiency: From a practical standpoint, in aviation, for a new system to be adopted and get widespread use, it must make economic sense and add significant value to safety, usability, and operational efficiency without adding unnecessary complexities. Our proposed framework is intentionally designed to be lightweight so that it can run on any low-powered and old computing systems with ease.

A count of trainable parameters in each model is presented in Table 5.5. In total, all three models of our framework have only 618,160 parameters. Compared to ResNET-50's 25 million and BERT-base's 110 million parameters, our entire framework has only 2.47% and 0.6% of the parameter count, respectively. This means the entire pipeline can run on just the CPU without requiring a separate GPU. This

makes it perfect for running on avionics-grade embedded computing systems.

Table 5.5: Trainable parameter count for each model in the framework.

Model	Trainable Parameters
CSPT (Stage 1, Model 1)	163,691
HiCST (Stage 1, Model 2)	203,131
MHANet (Stage 2)	251,338
Total	**618,160**

Table 5.6 shows the single-row inference latency for each component measured on a CPU. CSPT and HiCST run in parallel because both receive the same input window and have no dependency on each other. The effective Stage 1 latency is therefore determined by the slower of the two models (HiCST at 3.0215 ms) plus the negligible RMS fusion computation (1.419 μs), giving a Stage 1 total of 3.0229 ms. Stage 2 (MHANet) adds a further 3.0555 ms only when Stage 1 raises an anomaly alert. The complete pipeline latency is 6.0784 ms.

Table 5.6: Single-row CPU inference latency for each component.

Component	Inference Time
CSPT	2.1448 ms
HiCST	3.0215 ms
RMS Ensemble (fusion only)	1.419 μs
Stage 1 Total (parallel execution)	3.0229 ms
MHANet (Stage 2)	3.0555 ms
Full pipeline, with ensemble	**6.0784 ms**
Full pipeline, without ensemble	5.2003 ms

The NGAFID sensor system generates one new data row per second (1,000 ms). The full pipeline completes inference in 6.0784 ms, which is only 0.6% of the available sensor interval. This means the system can process every incoming sensor reading and return a result with no queuing or backlog, establishing genuine real-time capability. Furthermore, because 99.8% of all flight timesteps are normal, Stage 2 is idle for nearly all of flight time. During normal operation only Stage 1 runs, at a cost of 3.0229 ms per second. The additional 3.0555 ms of Stage 2 is incurred only on the rare occasions when Stage 1 raises an alert. This conditional computation structure keeps average-case latency and power consumption low.

Feasibility The combination of a 618,160-parameter footprint and sub-7 ms inference time makes the framework feasible across a range of deployment environments without requiring specialized hardware.

- **Ground-based monitoring.** A standard laptop or desktop PC at a flight school or fleet management center can run the full pipeline and process real-time telemetry from multiple aircraft simultaneously, given how low the per-inference cost is.

- **Onboard embedded systems.** Hardware at the level of a Raspberry Pi 4 or NVIDIA Jetson Nano is sufficient to run both stages onboard the aircraft, provided raw sensor data is available from the avionics bus. This enables autonomous in-flight monitoring without ground connectivity.
- **No augmentation.** The framework was trained entirely on real flight data without any synthetic augmentation. Augmenting time series data risks introducing artificial temporal patterns that do not reflect real flight physics, which could cause the model to learn spurious correlations. Training on real data only ensures that the model's detection behavior is grounded in genuine flight dynamics.

Strengths

- **End-to-end automation.** Traditional aviation safety monitoring relies on rule-based exceedance detection, where engineers manually define thresholds for each sensor parameter. This framework replaces that approach with a fully data-driven pipeline: anomaly detection and event classification are performed automatically from raw sensor input to classified alert, with no human intervention required at inference time.
- **Internal normalization.** All three models include a SensorNorm layer that stores training-set statistics as fixed parameters. Raw sensor readings can be passed directly to the models at inference time without any external preprocessing pipeline. This eliminates the risk of a mismatch between training-time and deployment-time data transformations, a common source of silent errors in production machine learning systems.
- **Scalability to other aircraft types.** The framework was developed for the Cessna 172S but the architecture is entirely data-driven. To adapt it to a different aircraft type, only retraining on that aircraft's sensor data is required; the model structure does not need to change. The NGAFID database contains data from other aircraft types, and extending the framework to those platforms is a natural direction for future work.

Weakness The first is its closed-world assumption regarding anomaly types. The Stage 1 models were trained to distinguish normal flight from anomalous flight based on the patterns present in the GATS dataset, and Stage 2 was trained to classify among 10 specific event types defined in the NGAFID event detection system. If a genuinely new type of anomaly occurs in flight one whose sensor signature differs from all patterns seen during training Stage 1 may assign it a low anomaly probability and fail to flag it at all, because the model has no mechanism for recognizing that a pattern is unfamiliar. There would be two likely possibilities in cases like these. One, it will flag the anomalous window as normal and would go undetected. Two, the anomalous window would get a probability score that is very close to the threshold value. This constraint is very fundamental to supervised learning approaches when applied to safety-critical applications like anomaly detection. In other words, the system can only reliably detect what it has been trained to detect.

Secondly, the trained framework cannot be generalized for other aircraft types. Although this is a data-driven architecture and therefore could in principle be used to retrain on the data from other aircraft types, the current three models were trained and tested exclusively on the flight recorder data of Cessna 172S aircrafts from two U.S. flight schools over a two-month period. Other aircrafts can significantly vary from this specific model of aircraft in terms of sensor configuration, aerodynamic characteristics, and operational profiles from this one. For that reason, it cannot be assumed that models trained on Cessna 172S can be generalized for different model of aircrafts.

5.3 Final Design Adjustments

This section describes how the final proposed framework was reached in two phases. The first subsection (Section 5.3.1) describes how a broader search was conducted in which many model families, attention configurations, loss functions, and training strategies were looked into before settling on this final architecture. Then the second subsection (Section 5.3.2) describes the process of iterative refinement that the models in the final architecture went through to reach their current best-performing configurations.

5.3.1 Architectural Exploration: Models Considered Before the Final Design

Before finally deciding upon the final architectures, we explored alternative model families. We learned from the limitations of each experiment regarding the attention configuration, data sequence modeling strategy, and parameters for loss functions, and training stability mechanisms for this specific dataset.

Stage 1: Binary Anomaly Detection

- **Point-in-time baselines:** Here, we used each one-second sensor reading separately using a 3-layer MLP with attention. With this simple setup, the models achieved poor precision due to the lack of temporal context. This resulted in the models' failure to distinguish between anomalies that last multiple seconds and also momentary spikes. Even after adding multiple attention heads and Focal loss improved recall, the False Positive (FP) count still remained high. Based on these observations, it made for sense to model the data in a way to preserve the temporal trends (ie. Sliding Windows).
- **TCN:** To address the limitations identified in the previous experiment, this time we tested Temporal Convolutional Networks (TCNs) with a short sliding window (W=4). This improved recall significantly. Moreover, using deeper and wider variants of TCN with attention, dual pooling, and Focal Loss also showed significantly improved F1-score. Seeing these results, we decided to try the experiments with a longer window size (W=30). However, we found that this actually leads to issues like training instability, specifically with CosineWarmRestarts, causing fluctuations in validation and degraded performance. Based on this, we understood that for this dataset, focusing on the optimization strategy with short sliding would be a better approach to take.

- **Precision-Focused Informer:** Finally, we developed a Precision-Focused Informer architecture with a cross-sensor multi-head attention layer to properly model inter-sensor dependencies. Then it was combined with adjusted sampling ratios and improved training strategies, namely OneCycleLR and balanced Focal Loss. It was observed that this approach reduces the False Positives (FPs) and achieves the best F1 score (F1 = 0.8624). From this experiment, we found that the cross-sensor attention mechanism worked really well on this dataset and therefore we kept using it for subsequent models.

Table 5.7: Architectural exploration summary of Stage 1

Version	Key Modification	Prec.	Recall	F1
Point-in-Time MLP v1	No temporal window; 3-layer MLP on raw 9 features; BCE loss; single attention head; no class weighting	0.5814	0.7203	0.6434
Point-in-Time MLP v2	2 attention heads; ReduceLROnPlateau scheduler; Focal Loss (gamma=1.5); model still single-timestep only	0.6112	0.7830	0.6864
TCN v1 (W=4, Shallow)	Sliding window (W=4) introduced; 2-layer causal dilated TCN; 1 attention head; LinearLR warmup; BCE loss	0.7012	0.8804	0.7806
TCN v2 (W=4, Deeper)	4 TCN layers; 2 attention heads; Dual Pooling (Max+Avg); Stochastic Depth; Focal Loss (gamma=2.0)	0.7248	0.9349	0.8166
TCN v3 (W=30, Wider)	Window extended to 30; 6-layer TCN [96,96,64,64,48,32]; CosineWarmRestarts; gamma raised to 2.8; training unstable	0.8214	0.8919	0.8552
Prec.-Focused Informer	Informer encoder; cross-sensor MHA prepended; 1:3 data sampling ratio; Focal Loss alpha=0.55; 4 encoder blocks (too deep – overfitting)	0.7831	0.8766	0.8290
Prec.-Focused Informer v2	Encoder reduced to 2 blocks; 2 attention heads; cross-sensor MHA retained; EMA decay=0.999; OneCycleLR replacing CosineWarmRestarts	0.8073	0.9255	0.8624

Stage 2: Multi-Label Event Classification

- **Multi-gate Mixture-of-Experts:** We used this architecture for our first experiment with four MLPs and 10 Softmax Gating Networks so negative transfer between conflicting event classes can be prevented. Asymmetric Loss ($\gamma_{\text{neg}} = 4, \gamma_{\text{pos}} = 1$) addressed class imbalance. With this our initial Macro F1 score stood at 0.8870. Our investigation revealed that this score was negatively

influenced by poor subset accuracy caused by the model's under-confidence level for rare classes. So, we applied per-class threshold tuning, temperature calibration, and adjusted γ_{neg} for specific classes. This bumped the Marco F1 score to 0.9279.

- **Feature-Attention Multi-Label Perceptron:** A less complex variant employed a common encoder with per-label softmax attention and separate MLP heads, allowing sensor importance per event to be interpreted. Using Focal Loss and class weighting, macro F1 was 0.8552, showing the drawbacks of a shared encoder. Additional improvements such as multi-head attention, cross-feature interaction, and stronger Asymmetric Loss ($\gamma_{\text{neg}} = 6$) led to a performance of 0.9180, but excessive suppression of gradients negatively affected frequent classes and performance on certain events.

- **Deep and Cross Networks with Sparsemax:** A DCN architecture was proposed to explicitly represent feature interactions, and Sparsemax allowed hard selection of sensors and better interpretability. Macro F1 was 0.9434 with 2 cross layers and label smoothing (0.05); increasing the number of cross layers and dimensionality to 4 and above increased it to 0.9464. But label smoothing had a detrimental effect on rare classes, adding noise. Its removal marginally decreased subset accuracy but increased per-class consistency, suggesting the significance of independent sensor representations as opposed to shared projections.

Table 5.8: Architectural exploration summary of Stage 2

Version	Key Modification	Macro F1	Subset Acc.	Hamming Loss
MMoE v1 (4 Experts)	4 expert MLPs; 10 Softmax gating networks; 1 task tower per event; ASL ($\gamma_{\text{neg}} = 4, \gamma_{\text{pos}} = 1, \text{clip} = 0.05$); AdamW; global threshold 0.5	0.8870	0.9120	0.0112
MMoE v2 (Tuning)	Per-class threshold sweep + temperature calibration; per-class γ_{neg} tuning (Roll: $\gamma = 2$, others: 4)	0.9279	0.9495	0.0068
FAMLP v1 (Per-label)	Shared encoder + per-label Softmax attention on sensors + MLP head per event; Focal Loss; ReduceLROnPlateau	0.8552	0.9102	0.0097
FAMLP v2 (Multi-head)	2 attention heads per label; Cross-feature MLP interaction layer; Gated label residuals; ASL ($\gamma_{\text{neg}} = 6$); CosineWarmRestarts	0.9180	0.9440	0.0071
DCN + Sparsemax	DCN with 2 cross layers; Sparsemax for hard selection; label smoothing=0.05; Weighted ASL	0.9434	0.9611	0.0052
DCN v2 (Expanded)	4 cross layers; SHARED_DIM = 256; HEAD_HIDDEN_DIM = 128; label smoothing removed; Weighted ASL	0.9464	0.9586	0.0055

5.3.2 Iterative Refinement of the Final Model Architectures

Based on the insights from our first phase, our final model architectures were gradually refined. This section describes our refinement processes for CSPT, HiCST, the Ensemble Fusion Strategies, and MHANet.

CSPT: Cross-Sensor Patch Transformer

- **CSPT v1 (Post-Patch Attention):** The original design used cross-sensor attention following patch embedding, that is, sensor values were combined prior to inter-sensor relationships being learned. This reduced the capability

of the model to model accurate sensor interactions leading to reduced precision (0.7640).

- **CSPT v2 (Pre-Patch Attention):** Cross-sensor attention was shifted to the pre-patch embedding stage, and the raw sensor relationships could be directly modeled. It was augmented with a sensor-identity embedding to maintain physical meaning, and attention heads (2 to 4) were added to richer representations. The model was augmented with SensorNorm and EMA (decay=0.999) enhanced training stability.

- **CSPT v3 (Final Optimization):** Patch configuration was optimized (patch_len $= 6, \text{stride} = 3 \rightarrow 9$ tokens), and the depth of Transformer was increased ($2 \rightarrow 3$ blocks). Focal Loss was enhanced ($\gamma = 2.8$, dynamic α) and better focus on hard anomalies was achieved. Replacing with OneCycleLR (with warmup) fixed training instability and enhanced convergence, with high recall (0.9156).

Table 5.9: CSPT version history.

CSPT Version	Key Change from Previous	Prec.	Recall	F1
CSPT v1	Post-patch attention; shared patch projection; no sensor identity; SensorNorm external; 2 heads in cross-sensor module	0.7640	0.8720	0.8144
CSPT v2	Pre-patch attention; per-sensor identity embedding; SensorNorm internalized; 4 attention heads; EMA decay=0.999	0.8070	0.8910	0.8469
CSPT v3	Patch tuned (9 tokens); 3 blocks; Focal Loss $\gamma = 2.8$; dynamic α clipped; OneCycleLR replacing CosineAnnealing	0.8246	0.9156	**0.8677**

HiCST: Hierarchical Cross-Sensor Transformer

- **HiCST v1 (Flat Transformer):** The original design used a flat Transformer on all 30 timesteps with no sequence compression. Attention was cross-sensor (2 heads, no FFN) and all the timesteps were equally considered, which did not allow paying attention to areas of anomaly. This resulted in reasonable accuracy (0.7980) but poor recall since subtle temporal cues were washed out.

- **HiCST v2 (Hierarchical Distillation):** A hierarchical distillation architecture (Conv1d $\rightarrow$ BatchNorm $\rightarrow$ ELU $\rightarrow$ MaxPool) was proposed, which reduced the sequence length by 30 down to 15 down to 8 tokens to focus on salient temporal patterns. CrossSensorMHA was enhanced to a complete encoder layer with FFN, and attention heads were expanded to 3. Initialization of CLS tokens was enhanced. While this increase our precision (0.8420), in decreased the recall score (0.8310), suggesting that our applied compression was overly aggressive.

- **HiCST v3 (Refined Distillation):** To enhance gradient flow, BatchNorm and ELU were retained, dropout (0.20 → 0.15) was reduced, and OneCycleLR with EMA (decay=0.9999) was used to train the model. The model had its optimal balance at a smaller decision threshold (0.85), which has more calibrated probability outputs and better overall performance.

Table 5.10: Version history of HiCST

HiCST Version	Key Change from Previous	Prec.	Recall	F1
HiCST v1	Flat Transformer; CrossSensorMHA (2 heads, no FFN); CLS zero-initialized; 2 blocks; same config as CSPT v1	0.7980	0.8540	0.8250
HiCST v2	Distillation (30→15→8 tokens); CrossSensorMHA full encoder layer; CLS truncated normal; 3 heads	0.8420	0.8310	0.8364
HiCST v3	Distillation (Conv1d-BN-ELU-MaxPool); dropout 0.15; 3 heads (16-dim); OneCycleLR; EMA 0.9999; AMP	0.8693	0.8602	**0.8647**

Ensemble Fusion Strategy Selection

- **Ensemble Fusion Strategies:** Once CSPT v3 and HiCST v3 were finalized, several fusion strategies were considered to combine their results. The most conservative was hard voting, where both models had to agree, which gave the highest precision (0.8820) but the lowest recall (0.8390). Arithmetic mean fusion increased recall (0.8980) but did not differentiate between the two models in terms of confidence, whereas AUPRC-weighted averaging did not improve significantly. Max confidence fusion maximized recall (0.9280) but minimized precision (0.8140) because of overconfident predictions by either model.

- **RMS Fusion (Final Choice):** The best balance (F1 = 0.8815) was obtained with the moderately weighting higher-confidence predictions without necessarily depending on them. When both models concurred, it acted as averaging, when one model was more confident, it boosted that signal without overfitting. This was in line with the fact that CSPT had a better recall and HiCST had a better precision, which enabled RMS to minimize the false positives without compromising the performance of the anomaly detector.

Table 5.11: Ensemble fusion strategy comparison.

Ensemble Strategy	Description	Prec.	Recall	F1
Hard Voting	Independent predictions; majority vote taken (anomaly if both models agree)	0.8820	0.8390	0.8599
Soft Voting	Average of CSPT and HiCST probabilities; single shared τ_{opt} sweep	0.8560	0.8980	0.8765
Weighted Avg.	Weighted by validation AUPRC; normalized weights applied before threshold	0.8572	0.8992	0.8778
Max Confidence	Final probability = $\max(p_{\text{CSPT}}, p_{\text{HiCST}})$; defers to most certain model	0.8140	0.9280	0.8674
Root Mean Square	$p_{\text{ens}} = \sqrt{(p_{\text{CSPT}}^2 + p_{\text{HiCST}}^2)/2}$; balances recall with precision; best overall F1	0.8568	0.9076	**0.8815**

MHANet: Multi-Head Attention Network

- **MHANet v1 (Shared Projection):** The initial design used a single shared linear projection to map all sensor values into a common embedding space. This caused ambiguity, as different sensors with identical normalized values produced indistinguishable representations. Positional encoding could not resolve this since it captures order, not identity. As a result, performance was limited (macro F1 = 0.8914) and rare classes were underpredicted.

- **MHANet v2 (Sensor-Specific Projection):** Replacing the shared projection with independent Linear(1→64) layers per sensor was the most impactful improvement. This enabled each sensor to learn distinct representations aligned with its physical meaning. Combined with per-class threshold tuning, macro F1 increased significantly to 0.9183.

- **MHANet v3 (Class Reweighting):** To deal with the performance of rare-classes, inverse-frequency class weights (normalized) were added, and the rarest class (Low Fuel) was further boosted. Although this increased per-class F1, it brought about early training instability. Learning was stabilized by increasing the CosineWarmRestarts cycle length (T 0 between 50 and 70).

- **MHANet v4 (Regularization and Stability):** Additional modifications involved tuning factors of class boosts (e.g., Low Fuel 2.5×), weight decay (10 -3 -10 -2), and dropout (0.25 -0.30) to decrease overfitting to low-dimensional inputs. The patience of early stopping was also increased to allow longer LR cycles, enhancing the stability of training.

5.4 Statistical Analysis

This section gives a more in-depth statistical analysis of the evaluation findings reported in Section 5.2. In Stage 1, the analysis will be based on the impact of threshold tuning on the detection performance, the precision-recall properties of each of the two models, per-event F1 comparisons between the two models, and the selection of the ensemble strategy. In the case of Stage 2, the analysis will look at the self-attention patterns acquired by MHANet, which will show what inter-sensor relationships the model uses to classify events.

5.4.1 Stage 1: Analysis of Binary Anomaly Detection

Effect of Threshold Tuning on Detection Performance The default binary classification systems use a fixed decision threshold of 0.5: a window where the probability of an anomaly is predicted to be above 0.5 is considered anomalous. This default is not suitable in a dataset where 99.8 percent of the windows are normal. When there is extreme class imbalance, the probability distribution learned by the model is concentrated around smaller values in the minority class: many truly anomalous windows are assigned probabilities in the range [0.30, 0.70] that would be incorrectly classified as normal by a 0.5 cutoff.

To address this, a threshold sweep was performed on the validation set at the end of each training epoch. Candidate thresholds from 0.20 to 0.99 in steps of 0.01 were evaluated, and the threshold yielding the highest binary F1 score on the validation set was recorded as the optimal threshold for that epoch. The threshold achieving the best validation F1 across all epochs was then applied to the test set. This procedure ensures that the threshold selection is entirely based on the validation set and the test set is never used for calibration.

Table 5.13 compares the test-set performance of each model and the ensemble under a fixed 0.5 threshold against their respective optimized thresholds. The improvement is consistent across all three: precision increases substantially because the higher optimized threshold (0.85 for HiCST, 0.88 for CSPT) reduces false positives, while recall is also maintained because the threshold was selected to maximize the harmonic mean of precision and recall rather than either metric alone.

Table 5.13: Effect of threshold optimization on test-set performance.

Model	Threshold	Precision	Recall	F1
CSPT	Fixed 0.50	0.7234	0.9560	0.8238
CSPT	Optimized 0.88	0.8246	0.9156	**0.8677**
HiCST	Fixed 0.50	0.6891	0.9530	0.7998
HiCST	Optimized 0.85	0.8693	0.8602	**0.8647**
RMS Ensemble	Fixed 0.50	0.7480	0.9480	0.8367
RMS Ensemble	Optimized τ_{opt}	0.8568	0.9076	**0.8815**

The values of the optimized threshold values themselves are meaningful. Both CSPT ($\tau = 0.88$) and HiCST ($\tau = 0.85$) converged to thresholds that were significantly higher than 0.5, indicating that both models place moderate-to-high probabilities

on actual anomalies instead of probabilities that are evenly distributed on [0,1]. The marginally reduced optimized threshold of HiCST (0.85 vs 0.88) is because the probability outputs of HiCST are generally lower and more conservative than those of CSPT, which is also in line with the architectural differences between the two models in terms of temporal compression.

Precision-Recall Analysis and Operating Characteristics The PR curve and threshold sensitivity plots of the CSPT model are presented in Figure 5.13. The PR curve starts with precision = 0.98 at low recall values, which means that the model is highly selective in the case of a high confidence threshold. The lower the threshold, the lower the precision, and the optimum operating point is at a threshold of 0.88 where F1 = 0.8677, indicated by the red dot on the curve. The AUPRC of 0.9419 puts the model significantly above the random baseline (which is the prevalence of the anomaly classes of about 0.003 in the test set). The right-hand subplot validates the clean, monotonic trade-off between precision and recall with changes in the threshold, with F1 maximizing evidently at 0.88.

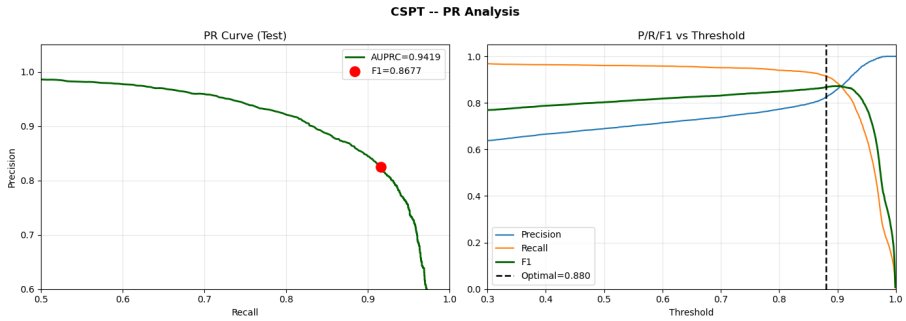

Figure 5.13: CSPT precision-recall characteristics.

Figure 5.14 shows the corresponding analysis for HiCST. The PR curve has a similar overall shape to CSPT's but starts with near-perfect precision (approaching 1.0 at low recall), reflecting HiCST's architectural preference for high-confidence predictions. The AUPRC of 0.9407 is marginally lower than CSPT's 0.9419. The optimal F1 of 0.8647 is achieved at a threshold of 0.904, which is higher than HiCST's training-time τ_{opt} of 0.85. This reflects that the PR curve analysis is evaluated at the default test-set threshold sweep whereas the reported test results used the validation-calibrated τ_{opt}. The right-hand subplot shows that HiCST's recall builds up more slowly than CSPT's as the threshold is lowered, consistent with HiCST assigning fewer high-probability scores to anomalous windows overall.

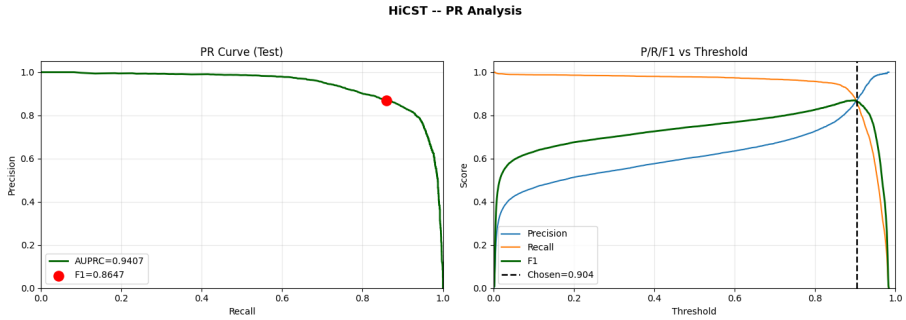

Figure 5.14: HiCST precision-recall characteristics.

Per-Event F1 Comparison: CSPT vs HiCST Although Stage 1 is a binary detection task, the per-event F1 breakdown in Figure 5.15 reveals how well each model detects different anomaly event types within its single anomaly-class output. This analysis is useful for understanding whether the two models have complementary strengths across event types, which informs the value of ensembling.

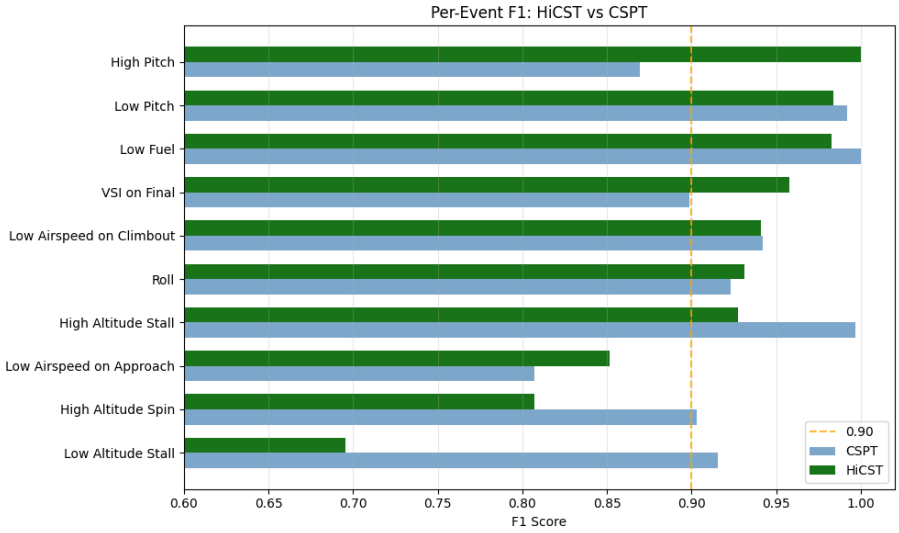

Figure 5.15: Per-event F1 scores for CSPT and HiCST.

Several patterns are visible in Figure 5.15. For High Altitude Stall, CSPT substantially outperforms HiCST (F1 near 1.00 vs approximately 0.93), likely because the stall event involves a gradual temporal build-up in stall index and IAS that CSPT's overlapping patch tokens capture more effectively than HiCST's hierarchical compression. For Low Altitude Stall, the pattern reverses: HiCST achieves a higher F1 than CSPT (approximately 0.91 vs 0.70), possibly because the rapid onset of low-altitude stall events is better captured by HiCST's attention over the full 30-timestep sequence. High Altitude Spin and Low Airspeed on Approach are the most challenging classes for both models, with F1 scores below the 0.90 reference line for at least one model. High Pitch is the most cleanly detected event for both, with both models near 1.00. The fact that no single model dominates across all event types confirms that the two architectures have genuinely complementary detection profiles, which is precisely the condition under which an ensemble is expected to outperform any individual model.

Ensemble Strategy Comparison Figure 5.16 shows the comparison of ensemble fusion strategies evaluated on the test set. Six strategies were considered: Weighted Average (W=0.40), Geometric Mean, Root Mean Square, Minimum Probability, Harmonic Mean, and Boolean AND (hard voting).

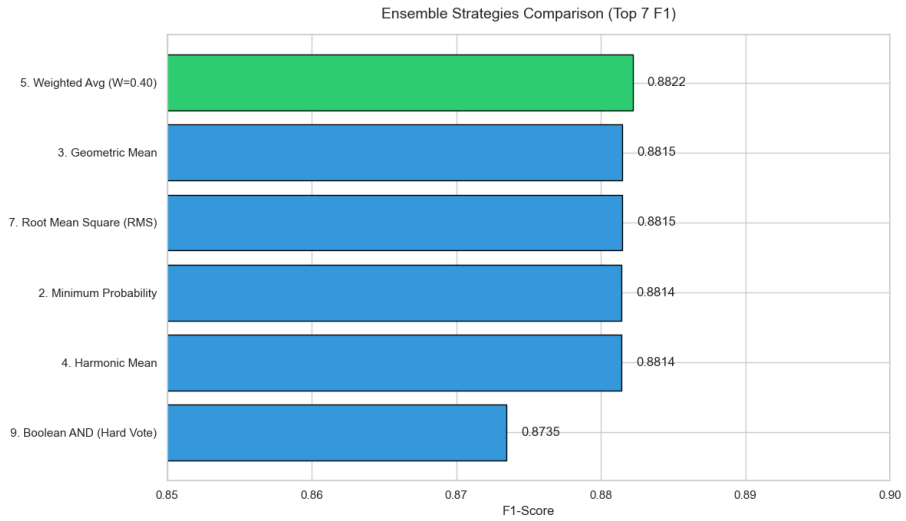

Figure 5.16: Ensemble strategy comparison by F1-score.

The Weighted Average with W=0.40 achieves the highest F1 of 0.8822, marginally above Geometric Mean and RMS at 0.8815. However, the margin between these strategies is extremely small (0.0007), and F1 alone does not capture the full picture. In aviation safety, recall is the more safety-critical metric: a missed anomaly is more costly than a false alarm. RMS fusion achieves a higher recall than Weighted Average while maintaining comparable precision, because RMS gives slightly more weight to the more confident model rather than applying a fixed weight regardless of which model is more certain in a given case. The dynamic nature of RMS weighting — amplifying whichever model scores higher on a particular window — is better suited to exploiting the complementary strengths of CSPT and HiCST than a fixed weight coefficient. Boolean AND (hard voting) produced the lowest F1 (0.8735) because it required both models to agree on an anomaly, which excessively penalized cases where only one model detected a genuine anomaly.

5.4.2 Stage 2: Analysis of MHANet Attention Patterns

Beyond the quantitative metrics reported in Section 5.2.1, an analysis of MHANet's internal self-attention weights provides insight into which inter-sensor relationships the model has learned to rely on for event classification. Because MHANet processes each anomalous timestep as a sequence of 9 sensor tokens and applies multi-head self-attention over them, the attention weights indicate which sensor pairs the model considers most informative when classifying event types.

Figure 5.17 shows the average global self-attention heatmap aggregated across all test timesteps and all attention heads. Each cell (i, j) represents the average attention weight assigned by query sensor i to key sensor j. A high value means that

when sensor i is trying to build its contextual representation, it draws heavily from sensor j's information.

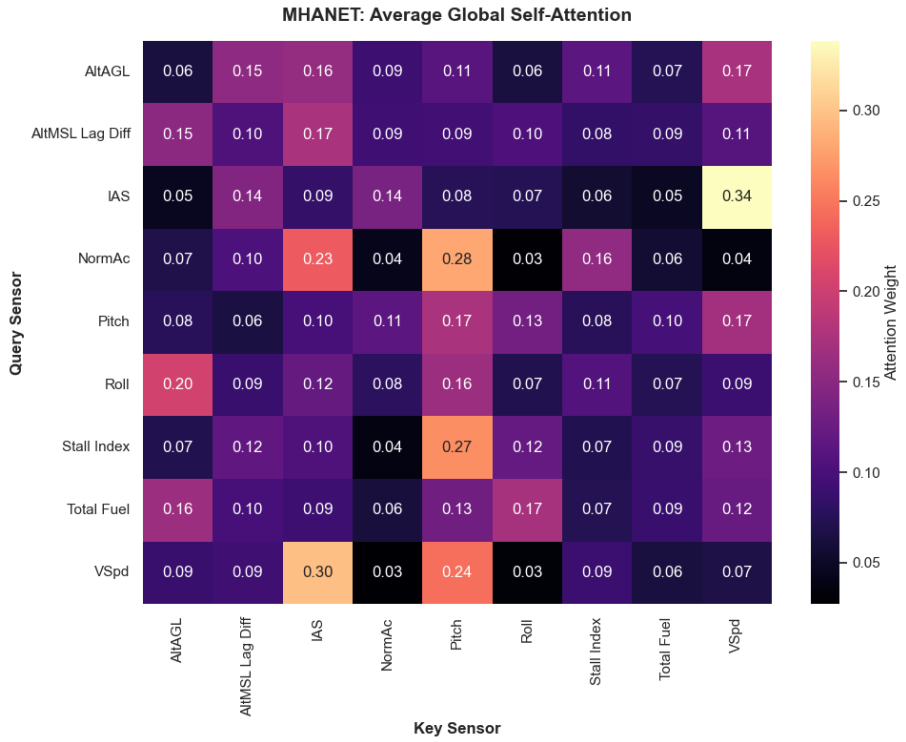

Figure 5.17: MHANet average self-attention weights.

The most prominent attention relationship is between IAS (Indicated Airspeed) and VSpd (Vertical Speed), with IAS querying VSpd at 0.34 and VSpd querying IAS at 0.30. This is physically well-grounded: the relationship between airspeed and vertical speed is a direct indicator of the aircraft's energy state and is relevant to several of the 10 target events, including Low Airspeed on Climbout, Low Airspeed on Approach, and High Altitude Stall. The model has successfully acquired the idea that the change in these two sensors should be read together.

NormAc (Normal Acceleration) is very sensitive to Pitch (0.28) and Stall Index is very sensitive to Pitch (0.27). Both of these relationships are aerodynamically significant: in level and climbing flight, normal acceleration is directly proportional to pitch angle, and the stall index (a measure of how close to aerodynamic stall one is) is strongly dependent on the angle of attack, which is directly related to pitch. The Roll sensor demonstrates moderate self-attention (0.20), which means that the Roll event class is based on the dynamics of the Roll sensor rather than on the relationships between sensors. AltAGL and AltMSL Lag Diff exhibit relatively low off-diagonal attention weights, indicating that altitude information is less useful in classifying events due to direct sensor interaction, and more useful due to its own value (which is already measured by the per-sensor embedding).

The general attention pattern validates that MHANet has acquired sensor relationships that are in line with the established aerodynamics of the Cessna 172S. No explicit encoding of domain knowledge was needed in the model; the physically

meaningful inter-sensor dependencies were learned by the data via the self-attention mechanism. In aviation, this feature of interpretability is very much useful. This is because the decisions made by models should be easily and quickly understandable to the flight operators and safety engineers.

5.5 Comparisons and Relationships

5.5.1 Comparison with Established Baseline Models

No published work has directly tested models on the GATS dataset used in this thesis, as the data is relatively new. To establish a meaningful performance reference, a set of well-established deep learning models for time series classification were evaluated on the same dataset, training split, and evaluation protocol used for the proposed models. Each baseline was trained with the same training strategy applied to CSPT and HiCST, including the Weighted Random Sampler, Focal Loss, OneCycleLR scheduler, and validation-set threshold sweep, so that differences in outcome reflect architectural choices rather than training procedure.

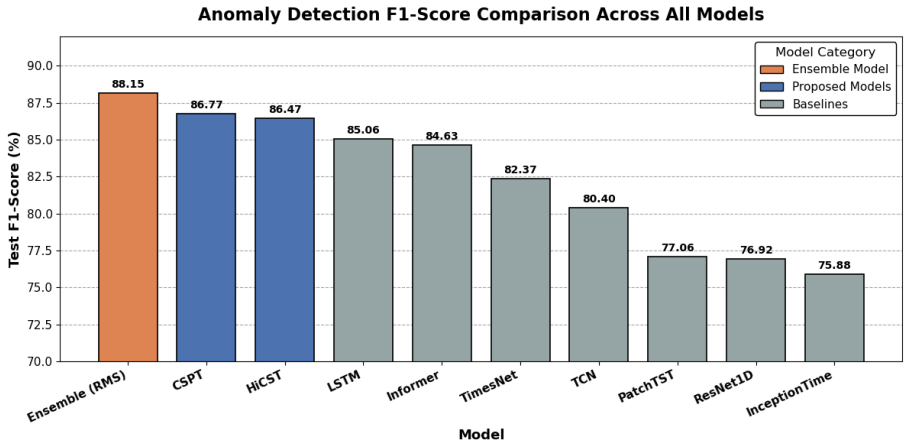

Figure 5.18: Anomaly Detection F1-Score Comparison

Stage 1: Binary Anomaly Detection As shown in Figure 5.18, the proposed models significantly outperform all baseline models in anomaly detection F1-score. The RMS Ensemble achieves the highest F1-score of **88.15%**, followed closely by the proposed CSPT (**86.77%**) and HiCST (**86.47%**). Among the baselines, LSTM records the best performance with an F1-score of **85.06%**, while the other models (Informer, TimesNet, TCN, PatchTST, ResNet1D, and InceptionTime) fall between 75.88% and 84.63%. Notably, the proposed models surpass even the strongest baseline (LSTM) by a clear margin.

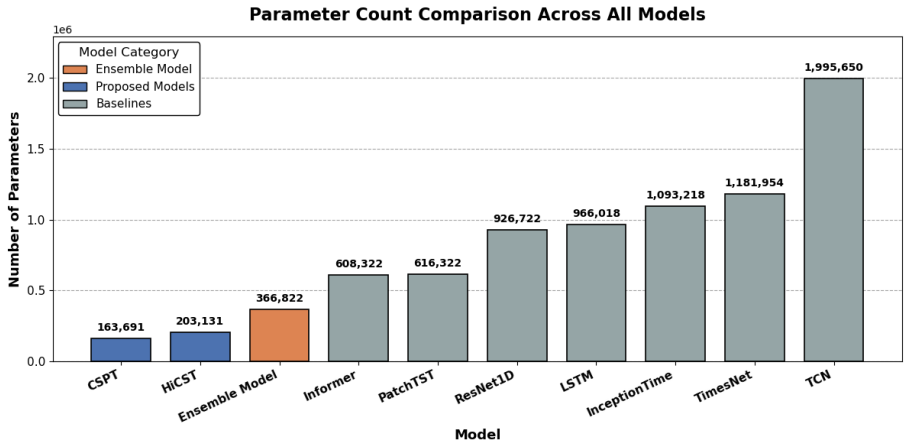

Figure 5.19: Parameter Count Comparison

Figure 5.19 compares the number of trainable parameters across all models. The proposed models demonstrate exceptional parameter efficiency. CSPT uses only **163,691** parameters, while HiCST uses **203,131** parameters substantially fewer than any baseline. In contrast, baseline models require significantly more parameters, ranging from 608,322 (Informer) to as high as **1,995,650** (TCN). The RMS Ensemble, despite combining both proposed models, still maintains a low parameter count of **366,822**, which is considerably smaller than most individual baseline models.

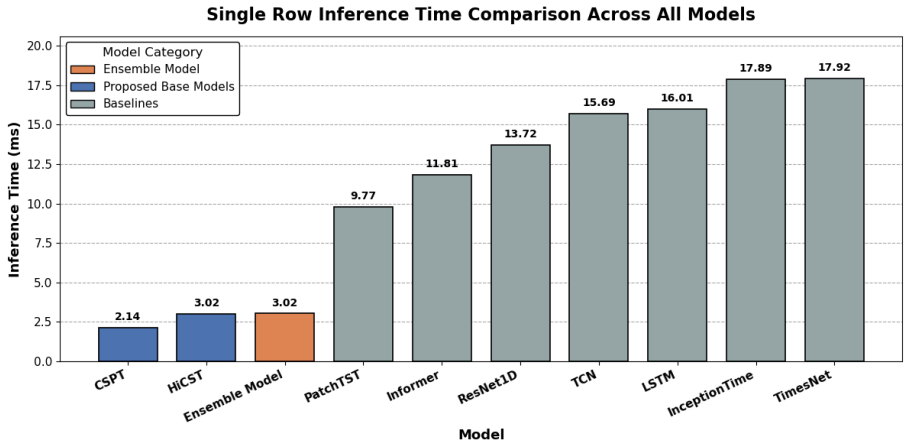

Figure 5.20: Single Row Inference Time Comparison

Figure 5.20 illustrates the single-row inference time (in milliseconds). The proposed models are dramatically faster than the baselines. CSPT achieves the lowest inference time of **2.14 ms**, followed by HiCST and the RMS Ensemble at **3.02 ms** each. In comparison, baseline models require much higher inference times, ranging from 9.77 ms (PatchTST) to 17.92 ms (TimesNet). This represents a substantial improvement in computational efficiency.

Overall, the proposed models not only deliver superior F1-scores but also achieve this with **far fewer parameters** and **significantly lower inference latency** compared to established baselines. The RMS Ensemble delivers the best F1-score (88.15%) while maintaining an extremely low inference time of just 3.02 ms, making

it highly suitable for real-time anomaly detection applications in aviation.

Figure 5.21 visualizes the efficiency trade-off as a scatter plot of inference time against parameter count. The proposed models (teal markers) cluster in the lower-left corner of the plot, occupying the high-efficiency region with both fewer parameters and faster inference than any baseline. All baseline models (orange markers) are scattered in the upper-right region, with TCN being the largest model and InceptionTime and TimesNet having the highest latency. This plot confirms that the proposed framework is not only more accurate but also more computationally efficient than the alternatives evaluated.

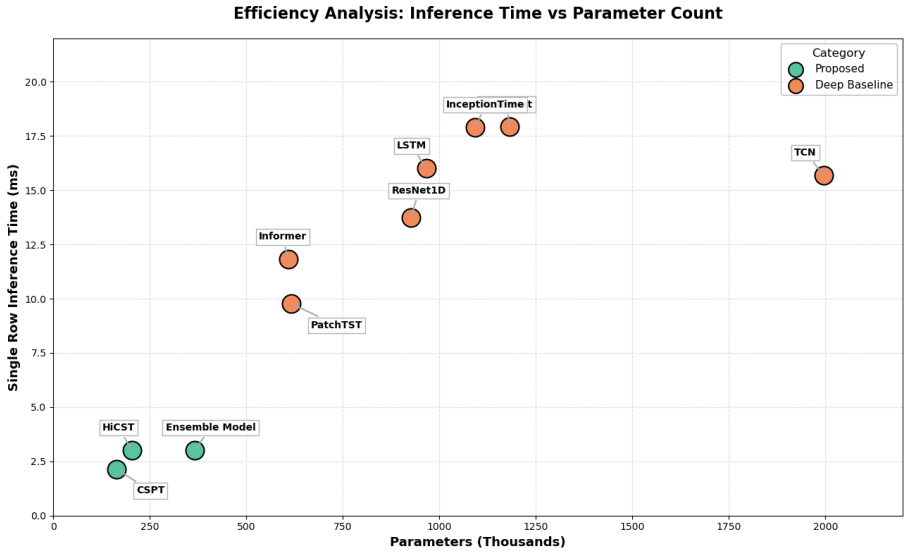

Figure 5.21: Stage 1 efficiency comparison.

Stage 2: Multi-Label Event Classification For Stage 2, MHANet is compared against three deep learning models (DCN, MMoE, and TabNet) and four classical machine learning models (Random Forest, CatBoost, LightGBM, and XGBoost). All models are evaluated on the same test set with per-class threshold tuning applied where applicable. Table 5.14 shows the deep learning comparison across macro F1, micro F1, subset accuracy, Hamming loss, parameter count, and inference time.

Table 5.14: Stage 2 deep learning model comparison. MHANet (proposed) is highlighted.

Model	Macro F1	Micro F1	Subset Acc.	Hamming Loss	Params	Inference (ms)
MHANet (Proposed)	**0.9563**	0.9771	0.9627	**0.00497**	251.3K	**3.06**
DCN	0.9337	0.9677	0.9492	0.00707	1,016K	13.04
MMoE	0.8482	0.9497	0.9215	0.01086	40.7K	1.10
TabNet	0.7194	0.9377	0.8784	0.01305	62.4K	452.63

Table 5.15 shows the comparison against classical machine learning baselines.

Table 5.15: Stage 2 classical machine learning baseline comparison.

Model	Macro F1	Micro F1	Subset Acc.	Hamming Loss	Inference (ms)
Random Forest	0.9231	0.9736	0.9559	0.00569	31.34
CatBoost	0.9207	0.9748	0.9555	0.00545	14.98
LightGBM	0.9207	0.9741	0.9531	0.00559	34.85
XGBoost	0.8534	0.9727	0.9506	0.00588	27.34

Among the deep learning models, MHANet leads on every metric. DCN achieves the next best macro F1 at 0.9337, but uses over four times as many parameters (1,016K vs 251K) and takes more than four times as long per inference (13.04 ms vs 3.06 ms). MMoE, despite having only 40.7K parameters and the shortest inference time (1.10 ms), falls significantly behind on macro F1 (0.8482), indicating that the expert routing approach without per-sensor independent projections is insufficient for reliable rare-class detection. TabNet performs the weakest (macro F1 = 0.7194) and has by far the highest inference cost (452.63 ms), making it unsuitable for real-time deployment regardless of its accuracy.

Figure 5.22 shows the grouped bar chart comparing macro F1, micro F1, and subset accuracy across all models. MHANet leads on all three metrics. The macro-micro gap for MHANet (0.9771 - 0.9563 = 0.0208) is the smallest among all deep learning models, confirming that MHANet is the most consistent across rare and frequent event classes. DCN has a larger macro-micro gap (0.9677 - 0.9337 = 0.0340), indicating that it performs better on frequent classes than on rare ones. MMoE shows the largest gap (0.9497 - 0.8482 = 0.1015), reflecting its failure to reliably classify rare events.

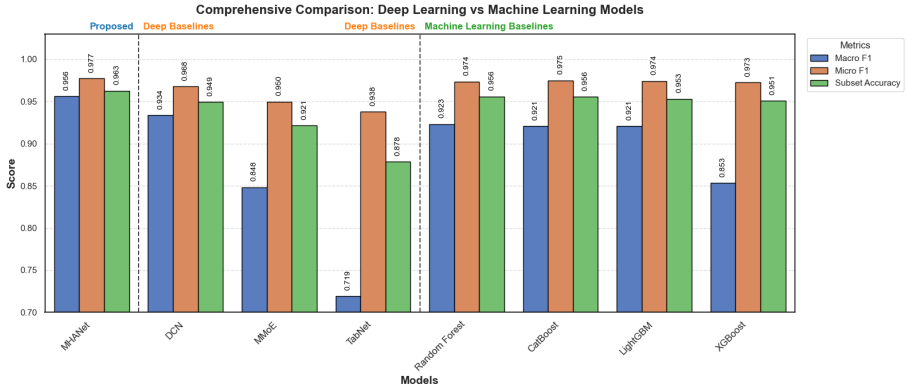

Figure 5.22: Comprehensive performance comparison of all Stage 2 models

Figure 5.23 shows Hamming loss across all models. MHANet achieves the lowest Hamming loss (0.00497), meaning that on average less than 0.5% of individual event label predictions per timestep are incorrect. DCN comes next (0.00707), followed by the classical ML models clustered around 0.0055-0.0059. MMoE (0.01086) and TabNet (0.01305) are substantially worse, with TabNet making approximately 1.3% per-label errors on average.

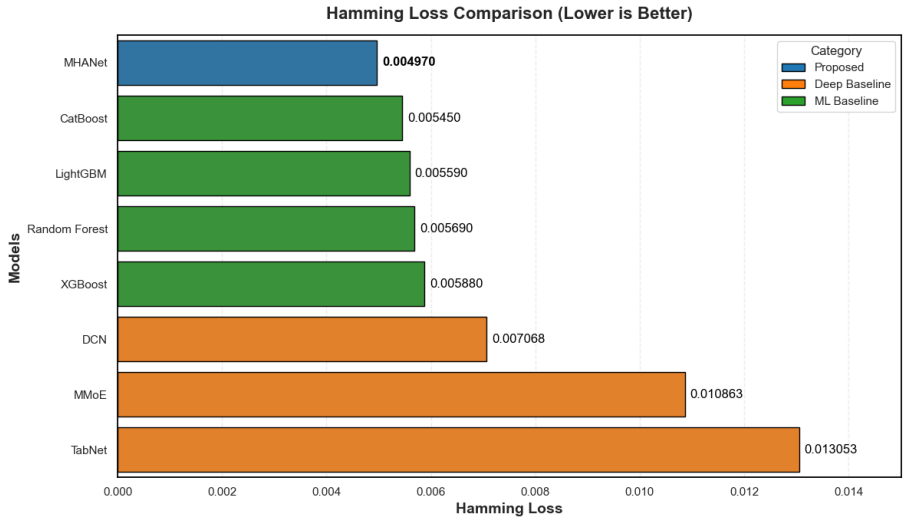

Figure 5.23: Hamming loss for all Stage 2 models

Figure 5.24 shows parameter counts for the deep learning models. MHANet uses 251K parameters, which is 75% fewer than DCN and 77% fewer than TabNet. MMoE uses the fewest parameters (40.7K) but at significantly lower accuracy. Figure 5.25 shows inference time on a log scale. TabNet's 452.63 ms makes it 148 times slower than MHANet. DCN is approximately 4.3 times slower. The classical ML models range from 15 to 35 ms, all slower than MHANet. Figure 5.26 shows the efficiency scatter plot of inference time vs macro F1. MHANet occupies the optimal corner – highest macro F1 and among the lowest inference times – while TabNet is isolated in the worst position (low F1, highest latency).

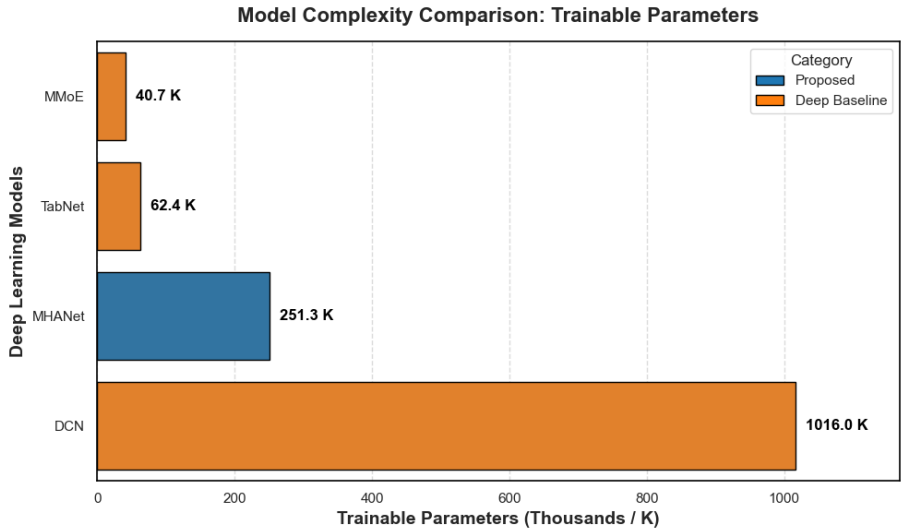

Figure 5.24: Trainable parameter counts for deep learning Stage 2 models

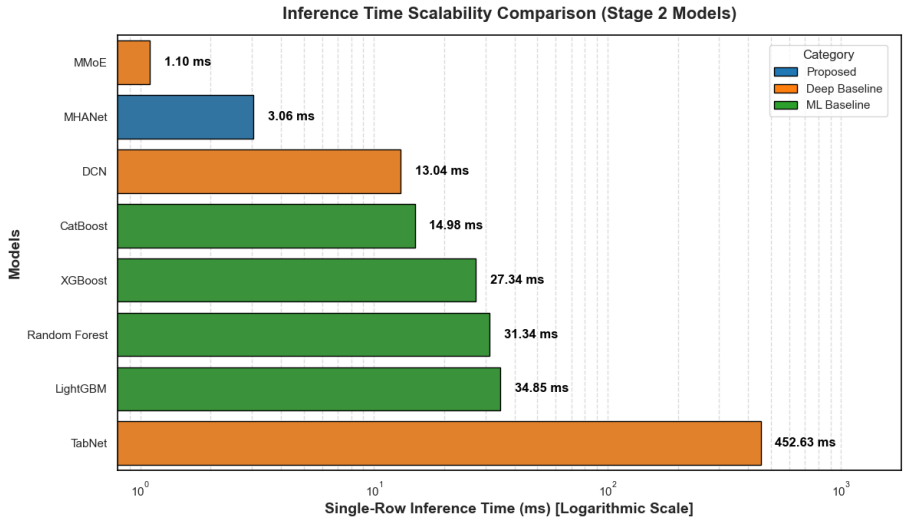

Figure 5.25: Inference time on log scale

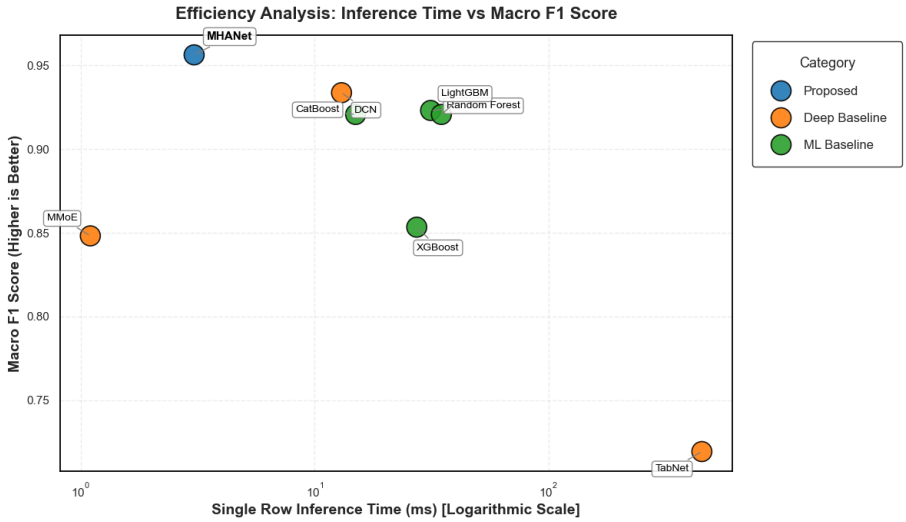

Figure 5.26: Inference time vs macro F1

Among the classical ML models, Random Forest achieves the highest macro F1 at 0.9231, which is 0.033 below MHANet's 0.9563. CatBoost and LightGBM follow closely at 0.9207. All three gradient boosting methods achieve competitive micro F1 and subset accuracy but have larger Hamming losses than MHANet (0.00545-0.00588 vs 0.00497). Their macro F1 scores are notably lower than MHANet's, confirming that classical models struggle more with rare event classes where training samples are scarce. Classical ML models also fail to generalize automatically to a deployment environment where normalization statistics are required to be stored with the model - MHANet internal SensorNorm makes it completely self-sufficient.

5.5.2 Contextual Comparison with Related Published Work

Since there is no previous research that tested models on the GATS data, a direct numerical comparison cannot be made. Rather, this section puts the proposed framework into perspective by contrasting it with two published studies that were

found during the literature review: one on flight anomaly detection (Stage 1) and one on multi-label flight event classification (Stage 2). It is aimed at pointing out the differences in the formulation of the problem, the nature of the dataset, and the methodology of evaluation, but not to draw a metric-to-metric claim.

Stage 1: Comparison with VAE-based MHSA-LSTM (Flight Anomaly Detection) The closest similar work to Stage 1 is an investigation with a VAE-based MHSA-LSTM model tested on real-world Quick Access Recorder (QAR) data of a commercial airline [15]. The paper presents precision, recall, and F1-score in three phases of flight on 100 normal flights of one aircraft type. They have achieved strong results here, most notably an F1 score of 0.9503, 0.9384, and 0.9718 for climb, cruise, and descent, respectively.

Nevertheless, the methodology of evaluation is fundamentally different to this thesis. In their work, a flight with at least one anomalous timestep is considered an anomalous flight and the task of the model is to predict normal or anomalous flights. Their training data consists of labeled normal flights only, and it is an unsupervised detection problem. Performance is evaluated at the flight level, i.e. one correctly classified anomalous flight is counted as one correct prediction no matter the number of anomalous timesteps it had.

Conversely, the framework proposed in this thesis works at the individual timestep level on the complete flight record. Secondly, an anomaly is identified on a flight-by-flight basis and over 2,258 flights with 10 different types of events and different onset durations. The test set has more than 1.2 million timestep windows, only 3,626 (0.3 percent) of which are anomalous. This per-timestep test is much more demanding than flight-level classification: a model should not just detect that a flight had an anomaly in it, but that each anomalous second of the flight was correctly identified. The imbalance ratio between per-timestep of about 339:1 is also significantly more difficult than the balanced flight-level problem encountered by the VAE-based MHSA-LSTM. In this more difficult per-timestep scenario, the proposed RMS ensemble has an F1 of 0.8815, which is indicative of the actual detection performance at every second of flight, as opposed to a more coarse flight-level label.

Stage 2: Comparison with Semi-Supervised Multi-Class Classification (FOQA Data) The most applicable published work in Stage 2 involves semi-supervised models (M1+M2 and CCLP) on Flight Operational Quality Assurance (FOQA) data of NASA DASHlink. They are tasked with multiclassifying flights (Nominal, Speed High, Path High, and Flaps Late) during the landing stage, based on 19 flight parameters measured within a 160-second window [13]. Their best model (M1+M2) with 1,000 labeled samples has around 80-83 percent accuracy on anomalous classes and about 97 percent on Flaps Late. Accuracy decreases to less than 73 percent with a small number of labeled samples (100). These findings indicate a much simpler classification environment than Stage 2 in this thesis: they categorize four mutually exclusive categories at the flight level, i.e. each flight is assigned one of four labels.

The Stage 2 task in this thesis is multi-label classification at the timestep level with

10 event types. A single timestep can simultaneously be labeled as Roll, Low Pitch, and High Altitude Spin at the same time – a co-occurrence pattern that is common in real flight (and impossible to represent in a single-label classification scheme). The training set contains approximately 35,000 labeled anomalous timestep samples compared to 250 per class in the comparable published study. MHANet achieves a macro F1 of 0.9563 and a subset accuracy of 0.9627 in this more challenging multi-label, multi-class, timestep-level setting. These results are not directly comparable to the published accuracy figures since the problem structures differ, but they confirm that the proposed approach scales to a substantially more complex and realistic event classification task.

5.6 Discussion

5.6.1 Summary of Results

The proposed two-stage framework demonstrates strong performance across both detection and classification tasks under the challenging conditions of extreme class imbalance and multi-label event co-occurrence. In Stage 1, the RMS ensemble of CSPT and HiCST achieves an anomaly-class F1-score of 0.8815, a recall of 0.9076, and an AUPRC of 0.9523 on a test set containing 1,228,570 normal windows and only 3,626 anomalous windows. Both individual models outperform all seven established deep learning baselines (LSTM, TCN, Informer, TimesNet, PatchTST, ResNet1D, and InceptionTime) on F1-score, while using between 3 and 12 times fewer parameters and achieving between 4 and 8 times lower inference latency. In Stage 2, MHANet achieves a macro F1 of 0.9563, a subset accuracy of 0.9627, and a Hamming loss of 0.00497 across 10 simultaneous event classes, outperforming all deep learning and classical machine learning baselines evaluated. The full pipeline, including both stages running sequentially, completes inference in 6.08 ms per second of sensor data, which is only 0.6% of the 1-second sensor interval and well within the real-time constraint.

One of the main results of the comparison study is that the specialization of architecture to the specifics of the aviation sensor data leads to significantly better results compared to the use of general-purpose time series models. The two design choices that most consistently led to performance improvements were the cross-sensor attention mechanism, which explicitly models inter-sensor dependencies at each timestep and then processes them with a temporal processing, and the per-sensor independent embedding in MHANet, which enables each sensor to build its own physically meaningful representation. These findings affirm that domain-sensitive architectural decisions, even comparatively simple ones, work better than scaling up the size of models or the number of parameters to this kind of problem.

5.6.2 Implications for System Deployment

The design of the framework is suitable to be deployed in real-time aviation monitoring without the need to have special hardware. The entire pipeline can handle one second of sensor data in about 6 ms, with more than 99% of the

available time wasted, so the system can easily match the rate of incoming data on a 1 Hz sensor stream on a typical CPU without queueing or backlog. All normalization statistics and decision thresholds are fixed model parameters, which means that it does not need external preprocessing pipeline or runtime calibration to deploy. The system can be operated on a regular desktop or laptop with a telemetry stream and process data on several aircraft at once due to how cheap the per-inference cost is in a ground-based monitoring scenario. To deploy onboard, the joint parameter footprint of 618,160 of all three models can be used with embedded hardware of the caliber of a Raspberry Pi 4 or NVIDIA Jetson Nano, allowing autonomous in-flight monitoring without ground connectivity.

The trigger-based two-stage design is operationally such that Stage 2 is not active when no anomaly exists, which is the majority of the flight time (99.7%), and reduces average power consumption and computational load. Stage 1 will issue an alert when an anomaly is observed and Stage 2 will instantly categorize the type of event based on the current sensor reading, and the outcome will be recorded in the flight data recorder with time and the duration of the event. To keep the system current over time, retraining would be necessary periodically as new flight data is added, especially when the operational profile of the fleet, its maintenance history or sensor configuration varies. Since each of the three models is wholly data-driven and self-contained, retraining on new data does not need any architectural modifications. Re-evaluation of the pre-tuned per-class thresholds on a new validation set should also be considered whenever retraining the model, as the best thresholds are determined by the learned probability distributions of the model.

5.6.3 Limitations

There are a number of limitations that should be mentioned along with the results. This framework was only trained and tested on Cessna 172S data of one flight school in the NGAFID GATS dataset, so its generalizability has not been tested on other aircraft types or other operational settings. The sensor envelopes, anomaly thresholds, and flight characteristics of aircraft like the Piper Archer or Cessna 172R differ, and they cannot be directly trained using the trained model without retraining. The dataset also captures the pilot behaviour, geographic location and weather conditions of a single institution, even of the same type of aircraft, which may not be representative of the diversity of general aviation operations. The Stage 2 classifier can only identify the 10 event types that it was trained on; real anomaly events that are not covered by those classes would be missed by Stage 1 or detected as anomalous without a classification, as the model has no way of identifying out-of-distribution patterns as novel. High Altitude Spin and Low Fuel in particular had an extreme scarcity of positive samples, which could not be completely overcome with Asymmetric Loss and manual class weight boosting, and model performance on these classes is not as reliable as on the more common events. There was no data augmentation in training to maintain the integrity of temporal structure, i.e. the model has not been trained on strange sensor noise patterns, on strange manoeuvres or on environments that are not represented in the training flights. Lastly, the scale of the dataset available, computing resources, and time were limiting factors to the extent of cross-validation, ablation studies, and hyperparameter search that could

be performed in this work. The conclusions that may be drawn based on these results would be significantly reinforced by a larger and more varied flight dataset that would represent a variety of aircraft types, flight schools, and geographic areas.

Chapter 6

Conclusion

6.1 Summary of Findings

The thesis introduced a two-stage deep learning model in real-time to detect anomalies in aviation and classify multiple labels of events based on multivariate flight sensor data of the NGAFID General Aviation Training Set. The framework deals with the entire pipeline of problem processing, starting at the ingestion of raw sensors, through to the production of classified events, at the level of the individual timestep, on data sampled at 1 Hz. Stage 1 employs a pair of custom-designed Transformer models, CSPT and HiCST, to do binary anomaly detection on 30-timestep sliding windows. Both models have the cross-sensor multi-head attention module that explicitly learns inter-sensor dependencies prior to processing the temporal sequences, which is inspired by the fact that most aviation anomaly events are represented by joint conditions of multiple sensors and not by anomalies in a single channel. CSPT works with the sequence of patches by processing the sequence of tokens in overlapping patches and HiCST uses hierarchical distillation to increasingly compress the sequence. Their complementary error profiles, CSPT with better recall and HiCST with better precision, make them very suitable to RMS ensemble fusion. The ensemble has an anomaly-class F1-score of 0.8815, recall of 0.9076, and AUPRC of 0.9523 on a test set with a 339:1 ratio of classes. Both models perform better than all seven established deep learning baselines tested, such as LSTM, TCN, Informer, and PatchTST, with 3 to 12 times fewer parameters and 4 to 8 times lower inference latency.

Stage 2 involves MHANet, a single-timestep multi-label classifier, to determine which of 10 event types is currently active whenever an anomaly alert is raised by Stage 1. The most notable architectural novelty of MHANet is that it uses nine independent per-sensor linear projections instead of a common projection, enabling each sensor to form a physically meaningful representation that represents its identity and not just its normalized value. Two Post-LayerNorm Transformer blocks are then used to model cross-sensor relationships on the 9-token sequence, and a three-layer classification head is used to generate 10 independent sigmoid outputs. MHANet has a macro F1 of 0.9563, subset accuracy of 0.9627 and Hamming loss of 0.00497, which is better than all deep learning and classical machine learning baselines tested. The small macro-micro gap (0.0208) is an affirmation of consistent performance between frequent and rare event classes. The

entire pipeline finishes inference at 6.08 ms/s of sensor data, or 0.6% of the sensor interval, and demonstrates real-time performance on a commodity CPU without a GPU. All the normalization statistics and decision thresholds are stored as fixed model parameters, and thus the system is self-sufficient and can be deployed locally without the need for external resource-heavy computing and networking infrastructures.

6.2 Contributions to the Field

This thesis contributes to the aviation safety monitoring and applied deep learning to time series anomaly detection in a few ways. To begin with, the thesis presents the first two-stage framework that concurrently detects binary anomalies in real time and classifies multi-label events on raw general aviation flight sensor data on an individual timestep basis. Previous aviation anomaly detection has been mainly binary flight-level classification (indicating an entire flight as anomalous), unsupervised detection using normal-only training data, or multiclass single-label detection. The joint issue of identifying anomalies on a second-by-second basis and at the same time identifying which particular events are taking place at that point in time has not been tackled previously in the multi-label context where multiple events can take place simultaneously. This problem definition is technically more challenging and more closely related to the real-world operational needs than flight-level classification.

Second, the thesis presents two custom Transformer-based architectures, CSPT and HiCST, designed to detect sensor anomalies in the aviation industry, both of which have a new cross-sensor multi-head attention module as the initial processing step. This explicitly-modeled inter-sensor-correlation module at every timestep, prior to any temporal processing, is tailored to the multivariate sensor environment in which anomaly signatures are expressed in terms of joint conditions across multiple sensors. Another architectural contribution to the hierarchical distillation mechanism in HiCST, where the 30-timestep sequence is recursively compressed by convolutional distillation after each Transformer layer, is another architectural contribution that allows multi-scale temporal abstraction with a small parameter footprint.

Third, the thesis presents MHANet, a per-sensor independent embedding Transformer to classify multi-label events, which proves that using a per-sensor learned linear projection is a significantly better design choice than a shared projection in tasks where different sensors represent physically distinct and non-interchangeable information. This design principle can be widely applied to other multi-sensor classification problems outside of aviation.

Fourth, the thesis provides the first systematic evaluation of a range of established time series deep learning models on the NGAFID GATS Cessna 172S dataset, establishing benchmark results for future work to build upon. The finding that domain-aware architectural specialization (cross-sensor attention, per-sensor projection) consistently outperforms general-purpose models at a fraction of the

parameter count and inference cost is a practically significant result for the design of real-time embedded monitoring systems.

Fifth, the framework exhibits end-to-end practical deployability without having to use GPU hardware, external preprocessing, or runtime calibration, and is one of the lightest operationally reported in the literature deep learning flight safety systems. The trigger-based two-stage design, where Stage 2 is only activated on confirmed anomalous timesteps, further reduces average computational load in deployment.

6.3 Recommendations for Future Work

Several directions are identified for extending and improving the framework based on the limitations encountered in this thesis.

Cross-aircraft validation and transfer learning: The framework was validated only on Cessna 172S data. Evaluating it on other aircraft types available in the NGAFID dataset, such as the Piper Archer and Cessna 172R, would establish whether the learned representations generalize across aircraft. A more practical direction would be transfer learning: fine-tuning the pre-trained Cessna 172S models on small amounts of labeled data from a new aircraft type rather than training from scratch. This approach could substantially reduce the data and compute requirements for adapting the framework to new platforms, and could potentially be extended to commercial aviation data such as FOQA records from Boeing 737 or Airbus A320 fleets.

Anomaly precursor detection: The existing system identifies anomalies once they have started. A more safety-important feature would be anticipating a looming anomaly a few seconds ahead of time, providing the pilot with actionable lead time. This can be investigated by continuing Stage 1 with a sequence-to-sequence prediction model in which the model predicts future sensor states and indicates windows in which the predicted trajectory is far out of the normal distribution. A conversion of the system to a proactive warning, rather than reactive alert, would be a significant improvement in its value in operation.

Adaptive threshold recalibration: Stage 1 ensemble threshold and Stage 2 per-class thresholds are tuned on the validation set, and are kept constant during deployment. Practically, sensor calibration drift, aircraft wear, and seasonal environmental change can cause a change in the optimal thresholds with time. A recalibration mechanism run online which occasionally reconfigures thresholds depending on the running statistics of incoming flight data, such as by Bayesian optimization or sliding-window F1 maximization, could keep detection reliability without necessarily retraining the model.

Semi-supervised and few-shot learning for rare events: There are very few positive training examples of High Altitude Spin and Low Fuel, which restricts the

accuracy of supervised learning on these classes. Two directions that are complementary are semi-supervised learning, where the huge amount of unlabeled normal flight data is used to enhance the representation space of rare anomaly classes, and few-shot learning with prototypical or matching networks, which can learn robust representations with only a small number of labeled examples. Both methods have the potential to enhance the detection of rare events without the need to collect more labeled data.

Fleet-level aggregated safety analysis: The present structure is tracking individual flights independently. By combining predictions over time across a fleet of aircraft, it would be possible to get a more comprehensive view of safety information: which pilots or instructors tend to produce certain types of anomalies, which routes or weather conditions are correlated with high rates of anomalies, and which aircrafts tend to generate anomaly patterns that are similar to maintenance problems. Such fleet-level insights would be directly strategic to the flight school management and aviation safety regulators.

Continual learning for model freshness: Because new flight data is constantly received during deployment, the patterns that the model has learned can become obsolete due to changes in pilot behavior, aircraft setup, or operating conditions. An elastic weight consolidation or progressive neural network-based continual or incremental learning would enable the model to learn new data whilst retaining previously acquired information, preventing the disastrous forgetting caused by standard fine-tuning on new data. This would enable the framework to be correct and up to date without the need to undergo complete retraining on a periodical basis.

Anomaly severity scoring: Practically, all anomalies are not equally dangerous: a stall occurrence at 1,200 feet will be much more dangerous than the same occurrence at 5,000 feet. To enable pilots and safety officers to prioritize alerts based on urgency, it would be possible to introduce a severity score to the binary detection and multi-label classification outputs based on sensor value magnitude, the current altitude, and the number of events happening simultaneously. This would render the system more operative in high-workload cockpit settings and bring its outputs more in line with the graduated risk assessment frameworks employed in aviation safety management.

Bibliography

[1] Y. He and J. Zhao, "Temporal convolutional networks for anomaly detection in time series," *Journal of Physics: Conference Series*, vol. 1213, no. 4, p. 042 050, 2019. DOI: 10.1088/1742-6596/1213/4/042050

[2] H. Ismail Fawaz, B. Lucas, G. Forestier, et al., "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, pp. 1936–1962, 2020. DOI: 10.1007/s10618-020-00710-y

[3] M. Memarzadeh, B. Matthews, and I. Avrekh, "Unsupervised anomaly detection in flight data using convolutional variational auto-encoder," *Aerospace*, vol. 7, no. 8, p. 115, 2020. DOI: 10.3390/aerospace7080115

[4] T. G. Puranik and D. N. Mavris, "Identification of instantaneous anomalies in general aviation operations using energy metrics," *Journal of Aerospace Information Systems*, vol. 17, no. 3, pp. 147–159, 2020. DOI: 10.2514/1.I010772

[5] B. M. de Silva et al., "Physics-informed machine learning for sensor fault detection with flight test data," *arXiv preprint arXiv:2006.13380*, 2020.

[6] S. Gopali, F. Abri, S. Siami-Namini, and A. Siami Namin, "A comparative study of detecting anomalies in time series data using lstm and tcn models," *arXiv preprint arXiv:2112.09293*, 2021. DOI: 10.48550/arXiv.2112.09293

[7] S. He et al., "Flight data anomaly detection and diagnosis with variable association change," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 346–354. DOI: 10.1145/3412841.3441916

[8] R. Mori, "Anomaly detection and cause analysis during landing approach using recurrent neural network," *Journal of Aerospace Information Systems*, vol. 18, no. 10, pp. 679–685, 2021. DOI: 10.2514/1.I010941

[9] A. A. Ajhari and I. G. P. K. Negara, "Aircraft flight movement anomaly detection using automatic dependent surveillance-broadcast," *International Journal on Informatics Visualization*, vol. 6, no. 4, pp. 821–828, 2022. [Online]. Available: http://dx.doi.org/10.30630/joiv.6.4.1166

[10] S. K. Jasra, G. Valentino, A. Muscat, and R. Camilleri, "Hybrid machine learning-statistical method for anomaly detection in flight data," *Applied Sciences*, vol. 12, no. 20, p. 10 261, 2022. DOI: 10.3390/app122010261

[11] A. P. LaBella et al., "Optimized flight safety event detection in the national general aviation flight information database," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*, Association for Computing Machinery, 2022, pp. 1570–1579. DOI: 10.1145/3477314.3507115

[12] M. Memarzadeh, A. Akbari Asanjan, and B. Matthews, "Robust and explainable semi-supervised deep learning model for anomaly detection in aviation," *Aerospace*, vol. 9, no. 8, p. 437, 2022. DOI: 10.3390/aerospace9080437

[13] M. Memarzadeh, B. Matthews, and T. Templin, "Multiclass anomaly detection in flight data using semi-supervised explainable deep learning model," *Journal of Aerospace Information Systems*, vol. 19, no. 2, pp. 83–97, 2022. DOI: 10.2514/1.1010959

[14] K. Qin, Q. Wang, B. Lu, H. Sun, and P. Shu, "Flight anomaly detection via a deep hybrid model," *Aerospace*, vol. 9, no. 6, p. 329, 2022. DOI: 10.3390/aerospace9060329

[15] C. Rong, S. OuYang, and H. Sun, "Anomaly detection in qar data using vae-lstm with multihead self-attention mechanism," *Mobile Information Systems*, vol. 2022, p. 8 378 187, 2022. DOI: 10.1155/2022/8378187

[16] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: Deep transformer networks for anomaly detection in multivariate time series data," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1201–1214, 2022. DOI: 10.14778/3514061.3514067

[17] H. Yang, A. LaBella, and T. Desell, "Predictive maintenance for general aviation using convolutional transformers," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, pp. 12 636–12 642, 2022. DOI: 10.1609/aaai.v36i11.21538

[18] M. Kazijevs and M. D. Samad, "Deep imputation of missing values in time series health data: A review with benchmarking," *Journal of Biomedical Informatics*, vol. 144, p. 104 440, 2023. DOI: 10.1016/j.jbi.2023.104440

[19] A. Nanyonga, H. Wasswa, and G. Wild, "Phase of flight classification in aviation safety using lstm, gru, and bilstm: A case study with asn dataset," in *2023 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*, Macau, China, 2023, pp. 24–28. DOI: 10.1109/HDIS60872.2023.10499521

[20] C. A. Rickert, M. Henkel, and O. Lieleg, "An efficiency-driven, correlation-based feature elimination strategy for small datasets," *APL Machine Learning*, vol. 1, no. 1, p. 016 105, 2023. DOI: 10.1063/5.0118207

[21] C. F. Worel, "The curse of dimensionality: When too many features break your machine learning model," SSRN, Tech. Rep., 2023. [Online]. Available: https://ssrn.com/abstract=5206704

[22] C. F. Worel, "The curse of dimensionality: When too many features break your machine learning model," SSRN, Tech. Rep., 2023. [Online]. Available: https://ssrn.com/abstract=5206704

[23] F. Zinnari et al., "A multivariate time-series segmentation framework for flight condition recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 3, pp. 2451–2463, 2023. DOI: 10.1109/TAES.2022.3215115

[24] AOPA Air Safety Institute, *Richard g. mcspadden report (35th ed.)* 2024. [Online]. Available: https://www.aopa.org/training-and-safety/air-safety-institute/accident-analysis/richard-g-mcspadden-report

[25] F. M. Kikhia, "A study on evaluation of mean and median imputation methods and their impact on statistical analysis of missing data," *Sirte University Scientific Journal*, vol. 14, no. 2, pp. 57–64, 2024.

[26] N. Mejri, L. Lopez-Fuentes, K. Roy, P. Chernakov, E. Ghorbel, and D. Aouada, "Unsupervised anomaly detection in time-series: An extensive evaluation and analysis of state-of-the-art methods," *Expert Systems with Applications*, vol. 256, p. 124 922, 2024. DOI: 10.1016/j.eswa.2024.124922

[27] R. Passarella, S. Nurmaini, M. N. Rachmatullah, H. Veny, and F. N. N. Hafidzoh, "Development of a machine learning model for predicting abnormalities of commercial airplanes," *Data Science and Management*, vol. 7, pp. 256–265, 2024. DOI: 10.1016/j.dsm.2024.03.002

[28] S. Sikdar, G. Hooker, and V. Kadiyali, "Variable importance measures for multivariate random forests," *Journal of Data Science*, vol. 22, no. 3, pp. 1–18, 2024. DOI: 10.6339/24-JDS1138

[29] H. Sun, F. Yang, P. Zhang, Y. Jiao, and Y. Zhao, "An innovative deep architecture for flight safety risk assessment based on time series data," *Computer Modeling in Engineering & Sciences*, vol. 138, no. 3, pp. 2549–2569, 2024. DOI: 10.32604/cmes.2023.030131

[30] X. Sun, S. Guo, S. Liu, Y. Zhang, and Z. Li, "A correlation-redundancy guided evolutionary algorithm and its application to high-dimensional feature selection in classification," *Neural Processing Letters*, vol. 56, p. 86, 2024. DOI: 10.1007/s11063-024-11440-3

[31] X. Sun, S. Guo, S. Liu, Y. Zhang, and Z. Li, "A correlation-redundancy guided evolutionary algorithm and its application to high-dimensional feature selection in classification," *Neural Processing Letters*, vol. 56, p. 86, 2024. DOI: 10.1007/s11063-024-11440-3

[32] R. Wang and J. Li, "An anomaly detection approach based on bidirectional temporal convolutional network and multi-head attention mechanism," *Information Technology and Control*, vol. 53, no. 1, pp. 37–52, 2024. DOI: 10.5755/j01.itc.53.1.34254

[33] M. Zhao, H. Peng, L. Li, and Y. Ren, "Graph attention network and informer for multivariate time series anomaly detection," *Sensors*, vol. 24, no. 5, p. 1522, 2024. DOI: 10.3390/s24051522

[34] A. LaBella et al., *Gats: A time-series dataset for addressing general aviation flight safety*, Hugging Face, Unpublished manuscript, ICML 2025 Submission, 2025. [Online]. Available: https : / / huggingface . co / datasets / NGAFID2025ICML/NGAFID-LOCI-GATS-Anonymous

[35] A. P. LaBella et al., "Gats: A time-series dataset for addressing general aviation flight safety," in *Proceedings of the 1st ICML Workshop on Time Series in the Age of Large Models*, International Conference on Machine Learning, 2025. [Online]. Available: https://icml.cc/virtual/2025/47472

[36] Z. Li, J. Ma, R. Fan, Y. Zhao, J. Ai, and Y. Dong, "Aircraft sensor fault diagnosis based on graphsage and attention mechanism," *Sensors*, vol. 25, no. 3, p. 809, 2025. DOI: 10.3390/s25030809

[37] S. Liu, W. Li, D. Yao, and J. Bi, "Effective and efficient representation learning for flight trajectories," in *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI-25)*, 2025, pp. 12 211–12 219. [Online]. Available: https://github.com/liushuoer/FLIGHT2VEC

[38] S. Martin, M. H. Laursen, and K. Calengor, "A deeper dive into general aviation loss of control: Interviews with pilots," *Journal of Air Transportation*, 2025. DOI: 10.2514/1.D0522

[39] R. Nåbo, R. Aga, and H. R. Karimi, "Algan: Time series anomaly detection with adjusted-lstm gan," *International Journal of Data Science and Analytics*, 2025. DOI: 10.1007/s41060-025-00810-2

[40] L. Ren, H. Wang, X. Pan, S. Jia, and B. Wan, "Quantitative evaluation method for anomaly levels of complex flight maneuver based on multi-sensor data," *Chinese Journal of Information Fusion*, vol. 2, no. 1, pp. 14–26, 2025. DOI: 10.62762/CJIF.2024.344084