FLIGHT REGIME CLASSIFICATION ON MILITARY STANDARD 1553B DATA BUSES

CLASSIFICATION DES RÉGIMES DE VOL POUR LE BUS DE DONNÉES MILSTAND 1553B

A Thesis Submitted to the Royal Military College of Canada
by

Dain Saulnier, BSc (Hons)
Captain

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Electrical and Computer Engineering

April 2026

Abstract

The Military Standard 1553 Revision B (MIL-STD-1553B) data bus, alternatively Standardization Agreement (STANAG) 3838, is an avionics data bus present in many western military aircraft since the mid 1970s. The bus allows communication between devices, Remote Terminals (RTs), to relay critical information which serve as essential information pipelines for modern military aircraft. This bus transmits avionics data to RTs about the status of various components as the aircraft moves through space. This dynamic movement is classified by domain experts into flight regimes. The aviation industry uses these flight regimes for many applications such as airframe structural analyses, accident investigations and fuel consumption reduction efforts. While abundant, MIL-STD-1553B is an old standard with security gaps that are challenging to remedy under the bus specification. This work bridges the gap between two fields, aircraft flight regime modeling and MIL-STD-1553B security research by permitting the segmentation of MIL-STD-1553B datasets into similar sets of flight regimes which we propose can be used to improve intrusion detection efforts on the bus. To date, no research has meaningfully attempted to combine these two domains.

The aim of this research is to verify the ability of deep learning models to perform aircraft flight regime classification using message traffic solely contained on the MIL-STD-1553B bus. Using an Extended Long Short-Term Memory (xLSTM) deep learning model, we classify real and synthetic avionics traffic extracted from MIL-STD-1553B buses in order to categorize dynamic aircraft movement into aircraft flight regimes. We evaluate the effectiveness using common statistical analysis techniques focusing on accuracy and weighted F_1 scores.

We consider the aim of this research to be met once the deep learning model classifies MIL-STD-1553B traffic into flight regimes. We evaluate the results using established statistical analysis methods. This research demonstrates accuracy scores of 0.9032 and 0.9470 on synthetic and real avionics data. Likewise, we observe respective weighted F_1 scores of 0.9019 and 0.9468.

Résumé

Le bus de données de Norme Militaire 1553 Révision B (NM1553B), aussi connu commme, l'accord de normalisation d'OTAN 3838, est un bus avionique présent dans de nombreux aéronefs militaires occidentaux depuis le milieu des années 1970. Ce bus permet la communication entre différents dispositifs, téléterminaux, afin de transmetter de l'information critique pour les aéronefs militaires modernes. Ce bus transmet des données avioniques vers les téléterminaux concernant l'état de divers composants alors que l'aéronef se déplace dans l'air. Ce mouvement dynamique est classé par des experts du domaine en régimes de vol. L'industrie aéronautique utilise ces régimes de vol pour de nombreuses applications, comme les analyses structurales de la cellule, les enquêtes d'accident et les efforts de réduction de la consommation de carburant. Bien que largement utilisé, la NM1553B est une norme ancienne présentant des lacunes de sécurité difficiles à corriger dans le cadre de sa spécification. Ce travail élimine l'écart entre deux domaines, la modélisation des régimes de vol des aéronefs et la recherche en sécurité du NM1553B, en permettant la segmentation d'ensembles de données NM1553B en ensembles similaires de régimes de vol, ce qui, selon nous, pourrait améliorer les efforts de détection d'intrusions sur le bus. À ce jour, aucune recherche n'a tenté de manière significative de combiner ces deux domaines.

Le but de cette recherche est de vérifier la capacité des modèles d'apprentissage profond à effectuer la classification des régimes de vol d'aéronefs en utilisant uniquement le trafic de messages contenu sur le bus NM1553B. En utilisant un modèle d'apprentissage profond de réseau de mémoire longue à courte terme étendu (xLSTM), nous classons du trafic avionique réel et synthétique extrait de bus NM1553B afin de catégoriser le mouvement dynamique de l'aéronef en régimes de vol. Nous évaluons l'efficacité à l'aide de techniques courantes d'analyse statistique, en mettant l'accent sur la précision et les scores F_1 pondérés. Nous suggérons que ce travail segmente le trafic NM1553B en régimes de vol distincts composés d'ensembles similaires de trafic du bus, ce qui pourrait être utilisé pour améliorer les efforts de détection d'anomalies

sur le bus.

Nous considérons que l'objectif de cette recherche est atteint une fois que le modèle d'apprentissage profond classe le trafic NM1553B en régimes de vol et que ces résultats sont évalués à l'aide de méthodes d'analyse statistique reconnues. Cette recherche démontre des résultas d'exactitude de 0.9032 et 0.9470 sur des données avioniques synthétiques et réelles. De même, nous observerons les scores F_1 pondérés respectifs des modèles 0.9019 et 0.9468.

Acknowledgements

I would like to thank my supervisors, Dr. Vincent Roberge and Brian Lachine for their support during the planning and execution of this thesis.

To my friends and family that have spent the better part of two years listening to me talking about computers and some aircraft bus, thank you. Thank you for putting up with me and for forcing me to rethink how I explain what this research actually is to others.

Contents

List of Tables

List of Figures

Acronyms

BC	Bus Controller
BM	Bus Monitor
CEL	Cross Entropy Loss
CSV	Comma-separated values
CUDA	Compute Unified Device Architecture
DTAES	Directorate of Technical Airworthiness and Engineering Support
FDR	Flight Data Recorder
FN	False Negative
FP	False Positive
HUMS	Health and Usage Monitoring System
ISR	Intelligence Surveillance Reconnaissance
LSTM	Long-Short Term Memory
MIL-STD-1553B	Military Standard 1553 Revision B
mLSTM	Matrix Long Short-Term Memory
NATO	North Atlantic Treaty Organization
PCA	Principal Component Analysis
RCAF	Royal Canadian Air Force
RNN	Recurrent Neural Network
RT	Remote Terminal
sLSTM	Scalar Long Short-Term Memory
STANAG	Standardization Agreement
TN	True Negative
TP	True Positive
WFDB	Waveform Database
xLSTM	Extended Long Short-Term Memory

1 Introduction

Aircraft are complex dynamic systems consisting of interconnected components permitting safe flight. Network buses exist on these aircraft to allow communication between interconnected systems in flight. One such bus used on many North Atlantic Treaty Organization (NATO) aircraft is the Military Standard 1553 Revision B (MIL-STD-1553B) (MIL-STD-1553B), also known as NATO Standardization Agreement (STANAG) 3838. Standardization of component specifications among alliance members increases compatibility of equipment and simplifies creation and updates to new components for both aircraft operators and manufacturers. With increasing cyber threats present in the modern world, the security of this data bus is essential to ensure the aircraft operates within an acceptable level of safety; guaranteeing its ability to safely carry out assigned missions.

Aircraft are machines that move in three-dimensional space, through various inputs given to flight surfaces, producing movement in corresponding directions. These movements can be classified into flight regimes as a set of terms for experts across the field to understand how the aircraft is moving as different regimes expose the aircraft to unique risks and dangers.

This thesis identifies a deficiency in current aircraft security research, namely that contemporary intrusion detection models on the MIL-STD-1553B data bus do not consider different operational baselines of traffic across different flight regimes. This research demonstrates an approach to classify data solely extracted from MIL-STD-1553B buses into aircraft flight regimes.

1.1 Motivation

In the modern battlefield, cyber has emerged as a new domain for military operations which requires actions to be taken by defenders in order to secure allied assets. Cyber vulnerabilities present credible threats to modern air forces [1]. A successful cyberattack could lead to degradation of military

capabilities or the loss of friendly or civilian lives.

Several NATO military aircraft use a standardized communications data bus, MIL-STD-1553B to connect various avionics and mission components on the aircraft. This bus standard, originally developed in the 1970s [2], was designed with the of objectives increasing reliability and fault tolerance [3] instead of security against unknown cyber threats present 50 years later. In the modern day, there are published attacks against the data bus in works such as Paquet [4] and many more described by Lounis et al. [3] which threaten the security of the bus. Aircraft in 2026, such as the CP-140 Aurora communicate via data links [5] to ground based stations during Intelligence Surveillance Reconnaissance (ISR) missions to provide live feed information to military commanders. Vulnerabilities in military supply chains could lead to adversaries having direct access to military aircraft without being in the vicinity of the aircraft [1, 3]. It was only in 2015 that Nguyen [6] published the first publicly available research attempting to highlight security vulnerabilities on the MIL-STD-1553B bus. However, this standard has existed for significantly longer than published security research.

The aviation industry is built on a strong foundation of safety; one built on the loss of lives through aviation accidents across the world. Programs such as the Royal Canadian Air Force's (RCAF) Flight Safety Program demonstrate that every opportunity can be taken to make aviation safer. Investigation of fatal incidents such as the crash of the CH148 Cyclone in the Ionian Sea in 2020 [7] used flight regime modelling to determine what happened in the last few moments of flight.

Current intrusion detection models [8–13] on the MIL-STD-1553B bus do not consider how the aircraft is flying while performing their analyses. However, aircraft are dynamic machines that transmit different information over the bus across various phases of flight leading to a broad definition of normal on the MIL-STD-1553B data bus. Therefore, we hypothesize that better intrusion detection models could be built with contextual awareness of what the aircraft is doing. Smaller, refined models trained on data clustered by flight regime should detect anomalous traffic faster and with higher accuracy than large generalized models across all types of aircraft flight.

1.2 Statement of Deficiency

Contemporary security research examining the MIL-STD-1553B data bus performs their analyses without consideration of the flown aircraft flight regime. The aviation industry uses flight regime information for a number of different

applications such as the development of flight envelopes for safe flight, incident investigation and risk management processes. There is only one piece of previous research publicly available that attempts flight regime modelling with MIL-STD-1553B traffic performed by Berry et al. [14]. Their analysis extracted select parameters from the bus instead of using raw message data, in addition to adding more instrumentation to the aircraft. Their aim was to model both aircraft flight regimes and weight and balance analysis of American Army helicopters. The ability to analyze MIL-STD-1553B traffic is essential as it not a given that other sources of data, such as Health and Usage Monitoring System (HUMS), is present in order to perform flight regime classification.

Contemporary security research on the MIL-STD-1553B data bus does not take into account the unique characteristics of aircraft flight. [8, 11–13]. In order to find anomalous traffic in the data, an adequate baseline defining normal states needs to be developed. The normal for an aircraft's altitude during cruise, where it is typically high and consistent, versus a landing where it is low and decreasing presents a challenge to the contemporary approach of *one model fits everything.*

Security research on the bus only dates back to Paquet's research [4] in 2014 on a bus created in the 1970s — 40 years later. Only works by Elsayed et al. and Wrana et al. [10, 15] have made distinctions between different aircraft movement in their datasets. Elsayed et al. [10], note differences in performance using the same model on the two different flight regimes each contained in their own dataset. In addition, they also revisited the models employed by Onodueze and Josyula [9] and He et al. [16] and noted differences in performance contrasting the two different datasets.

1.3 Statement of Aim

The aim of this research is to verify the ability of deep learning models to perform aircraft flight regime classification using message traffic solely contained on the MIL-STD-1553B bus. This will be accomplished through the development of an Extended Long Short-Term Memory (xLSTM) deep learning model in order to classify MIL-STD-1553B network traffic from real and simulated aircraft into flight regimes. We measure success of the aim using common statistical analysis methods used in machine learning applications. Contextual awareness of aircraft movement created by this work can be applied in future security research on the MIL-STD-1553B bus through the creation of distinct datasets by flight regime.

1.4 Research Activities

In order to satisfy the aim of this research, this work is broken down into the three principal phases described below:

1. Data Preparation: This phase consists of the collection and processing of two MIL-STD-1553B datasets for later use in the scope of this research. The first dataset, by Mukherjee [17] consists of three simulated aircraft flights recorded on the Royal Military College of Canada's (RMC) Cyber Flight Trainer and MIL-STD-1553B Simulator with accompanying video recordings of the aircraft cockpit from the pilot's perspective. The second dataset consisted of seven flights extracted from a RCAF CP-140 Aurora provided by Directorate of Technical Airworthiness and Engineering Support (DTAES) [18]. Accompanying this second dataset was a general listing of missions and tasks the aircraft conducted, annotated with timestamps. These datasets were processed separately to label the MIL-STD-1553B aircraft data.
2. Model Construction and Training: This phase consists of building an xLSTM deep learning network, feature selection and hyperparameter tuning to classify flight regimes from traffic extracted from the MIL-STD-1553B data bus. The final portion of this phase also trains models to classify aircraft flight regimes.
3. Verification and Validation: The final phase consists of two portions, verification of the model's fit for purpose and validation of the results using common statistical analysis techniques.

1.5 Results

The results of this research demonstrated that xLSTM deep learning models are capable of performing flight regime classification using only MIL-STD-1553B data bus traffic. Experimental results produced weighted F_1 scores of 0.9019 and 0.9468 on synthetic [17] and real aircraft data [18], respectively. The 7 Matrix Long Short-Term Memory (mLSTM) block to 1 Scalar Long Short-Term Memory (sLSTM) block xLSTM ([7:1]) stack configuration obtained the highest F_1 scores across both datasets. Models with fewer xLSTM components such as the [0:1] configuration trained significantly faster.

1.6 Organization

The remainder of this work is structured into four additional chapters that detail the research.

Chapter 2 includes background information relevant to this research including an overview of flight regime modelling, the MIL-STD-1553B data bus protocol and an introduction to the xLSTM deep learning model. Chapter 2 also covers the statistical analysis methods used to evaluate the model developed in this research as well as a summary of deficiencies in this research space. Chapter 3 details our methodology to classify aircraft flight regimes from the MIL-STD-1553B data bus and how the model's performance will be evaluated. Chapter 4 presents and discusses the findings of this research including verification of the presented model and validation of the aim. Chapter 5 concludes this thesis summarizing the scope of work, findings, contributions to the field and areas for potential future work.

2 Background

This chapter focuses on explaining key concepts related to this field of research to ensure common understanding of all terms and concepts explored through this work. The topics explained within this chapter are key to understanding the scope of this research. In addition, we present previous research that is fundamental to the basis of this thesis.

2.1 Military Standard 1553 Revision B

The MIL-STD-1553B data bus was developed originally for American military aircraft in 1975, superseded by revision B in 1978 [19], though its use in the modern day has expanded to other domains such as the International Space Station [20]. Published research on the MIL-STD-1553B bus in the cybersecurity domain only dates back ten years [6] with a modest selection of papers on the subject.

On the MIL-STD-1553B bus, all communication is managed by a centralized device called the Bus Controller (BC). When permitted by the bus schedule, devices will transmit data as a series of 20 bit words. This standard defines three unique words: Command, Data and Status. To ensure continuous operation in military environments, the bus contains redundant data lines, however these additional lines do not increase data transmission capacity and instead only serve as redundancy in the event of a malfunction of a primary line [19]. Communication occurs principally over the primary channel and typically only resorts to secondary channels in the event of errors.

The original design specification for the MIL-STD-1553B data bus focused on ensuring reliability and fault tolerance, creating room for cyber vulnerabilities that leaves the architecture susceptible to cyberattacks [3]. This decision resulted in only basic integrity checking of the confidentiality, integrity, and availability triad with an included odd parity integrity bit at the end of each word. Attacks exist which could “compromise the confidentiality, integrity,

and availability of systems that use the MIL-STD-1553B bus " [8].

An adversary could gain access to an aircraft's MIL-STD-1553B bus though different avenues including: a compromised supplier providing infected software or hardware, an insider performing an unauthorized modification leading to infection, or impersonation of authorized personnel to gain access to the aircraft [3]. Once access is obtained, attacks such as those detailed by Lounis et al. in [3] or demonstrated by Paquet in [4] could be executed against the aircraft potentially rendering them unsafe for flight or military operations.

A MIL-STD-1553B network is composed of three different types of network devices. A sample network layout can be seen in Fig. 2.1. According to the MIL-STD-1553B specification [19], these components are broken down as follows:

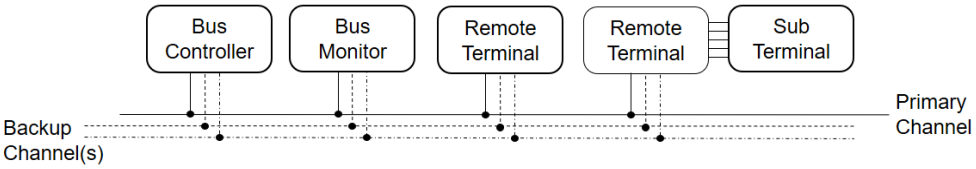

Fig. 2.1: MIL-STD-1553B Sample Network Layout - Adapted from [19].

1. *Bus Monitor (BM)*: Listens and records network traffic that is sent over the bus. There is no limit to the number of BMs on a network.
2. *Remote Terminal (RT)*: Devices which serve to communicate information over the bus such as an altimeter or the air data computer. Each RT is assigned a unique five bit identification number. The highest address is reserved for network broadcast, therefore networks are limited to 31 RTs. RTs may also have subcomponents attached to them which can be addressed using a second five bit identification field such as different sensors attached to an engine.
3. *Bus Controller (BC)*: Serves as the device that directs the network. It initiates information transfers over the network. Each network is limited to one BC. However, the specification allows for different RTs to become the BC, provided only one BC exists at a time. This transfer is communicated across the bus to ensure all devices are aware of the new BC. This feature is a redundancy protection intrinsic to the network.

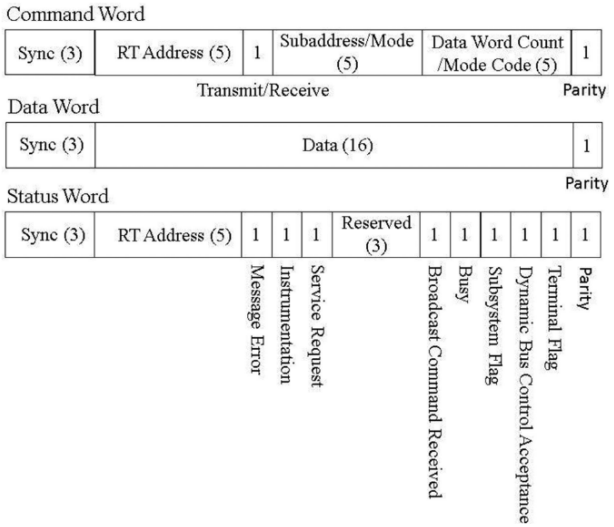

Fig. 2.2: Structure of MIL-STD-1553B Words [12]

MIL-STD-1553B messages consist of series of words of varying length transmitted between the RTs. The specification [2] defines three word types: command, data and status words. Each word is exactly 20 bits in length with three bits to synchronize the begining of a word using a valid or invalid Manchester signal, followed by 16 bits of content and terminating with an odd parity check bit. The protocol includes little room for expansion, with only one of three words, the status word, having unused fields, totalling three bits. Fig. 2.2 details the bit layout of the words which are described in additional detail below:

1. *Command Words*: Composed of four unique fields plus the synchronization and parity bits. First is the RT destination address, followed by a receive (low) or transmit (high) bit. The second field indicates either a RT subaddress or the use of mode control, `0b00000` and `0b11111` are reserved for an optional mode control. The third field indicates either the number of data words to expect or a mode code depending on the value of the second field. In the event of data word count, `0b11111` and `0b00000` represent 31 and 32 words respectively.
2. *Data Words*: Defines words sent to transmit information between RTs. 16 bits are transmitted in each word. Longer sequences can be transmitted through multiple data words being transmitted in Most-Significant-Bit order. A parity bit is included at the end of each word. Up to 32 data words, or 512 bits, may be contained in a single message.

3. *Status Words*: An information dense word that is broken down into ten unique fields. The status word starts with the RT address and a message error bit indicating that one or more of the previous data words failed the parity check. Next are two optional bits. The first, an instrumentation bit which is always set to differentiate a command and status word. The second indicates that the BC must take predefined actions relative to the sending RT which is sent in subsequent data words. The following three bits are reserved for future use. Next, the broadcast command received flag is set if the previous message was a command which was broadcasted. Following are four optional fields, the busy, subsystem flag, dynamic bus control acceptance and terminal flag bits. The first stating that the RT is unable to move data in response to the BC's request to transmit data. The second warns the BC to potentially invalid data. The third informs whether the RT accepts the offer to act as BC for the network. The fourth, indicates a RT fault condition, if set.

Sequences of MIL-STD-1553B words define messages. The flow of messages is controlled by the BC, however messages are not required to be sent to the BC and can be sent between two different RTs upon direction from the BC. There are ten different valid message sequences to facilitate communications between the BC and the RTs as defined by the specification. Fig. 2.3 samples three different communications between the various devices on the network. In the first case, the BC sends a command word followed by up to 32 data words to a RT. The RT then responds to the BC with a status word. In the second instance, the BC sends a command word to the RT, and the RT responds with a status word followed by up to 32 data words. In the last case, the BC sends a command word to two different RTs to communicate with each other. The sending RT transmits a status word to the BC and data words to the other RT which is acknowledged by the receiving RT with a status word to the BC.

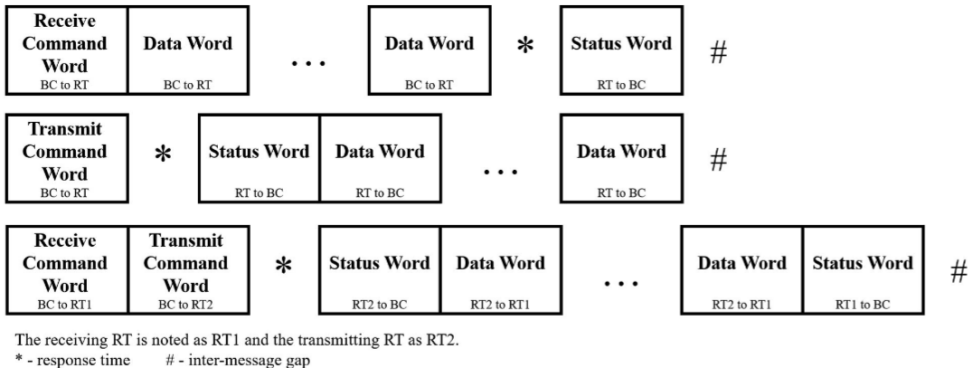

Fig. 2.3: Sample of MIL-STD-1553B Messages - Adapted from [13]

The MIL-STD-1553B bus communications are typically composed of periodic messages dictated by the bus schedule. However, aperiodic messages are sent when required. Aperiodic messages are typically event driven instead of operating on a defined schedule like periodic ones [21]. Due to the typically predictable nature of the network, aperiodic messages often look anomalous. The specified format of what information is transmitted over the bus is specified by the bus schedule or the interface control design implementation.

A simplified example of a MIL-STD-1553B bus is seen in Fig. 2.4 which consists of a network of three RTs. The pilot display found in the cockpit (RT1), an engine (RT2), and the air data computer which is doubled hatted as the BC. In this example of network transmission, the end state is achieved by the pilot display being informed of the engine temperature. This sequence consists of five distinct messages as follows:

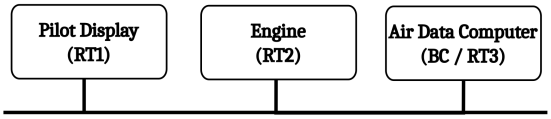

Fig. 2.4: Simplified MIL-STD-1553B Network

1. The BC sends a command word to the engine, RT2, that it will be transmitting its temperature to the pilot display, RT1.
2. The BC sends a command word to the pilot display, RT1, that it will be receiving the engine temperature from the engine, RT2.

3. The engine, RT2, transmits a status word to the BC confirming that the previous message was properly received.
4. The engine, RT2, transmits through up to 32 data words its temperature to the pilot display, RT1.
5. The pilot display, RT1, sends a status word to the BC confirming the communication was properly recieved without error.

There has been one previous attempt to classify MIL-STD-1553B data into flight regimes by Berry et al. [14]. Their data collection read data off of the MIL-STD-1553B bus, and incorporated information from additional accelerometers and tachometers into their dataset. This data was reduced to 20 features instead of processing raw message traffic through their model. These features included radar altitude and calibrated airspeed among others. Subsequently, they attempted to perform flight regime classification into 141 classes grouped into 11 flight regime super classes with a moderate level of success using elliptic base functions. Section 2.6 provides further details on their research.

2.2 Aircraft Flight Regimes

Civilian and military aircraft collect vast amounts of information for many purposes; the reasons are diverse including safety, maintainability and financial considerations. Data collection devices are installed on aircraft to record various aircraft states in order to serve as an authoritative record of what the aircraft did. One way of communicating this movement in concise terms is describing an aircraft flight regime. To put it simply, it is the way in which the aerospace vehicle is travelling through three-dimensional space. Examples could include taxiing, take off, turns, hovering or supersonic flight. Modern day academic literature and governmental aviation authorities have not published accepted lists of defined flight regimes [22]. Typically, each aircraft has a defined set of flight regimes described in its individual aircraft operating instructions.

Modern aircraft are complex machines, legislated by national governments to have redundant safety systems including recording capabilities for a variety of safety reasons. One such reason, is to figure out what happened in the event of an aircraft crash to prevent the incident from occurring again. In accordance with Canadian Aviation Regulation 625, Flight Data Recorders (FDRs) are required to log 22 different parameters including altitude and indicated airspeed every second [23]. FDRs pull avionics data from a variety of sources, including MIL-STD-1553B data avionics buses. Another common

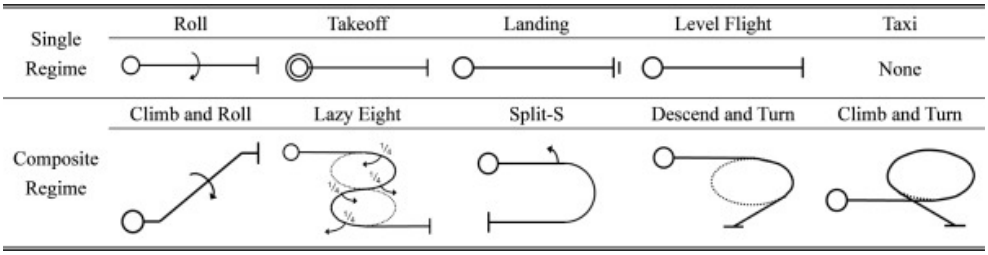

Fig. 2.5: Sample Flight Regimes [25]

data recording device, the HUMS typically serves to inform maintainers about the performance of the aircraft. Aircraft flight regime modelling has historically been performed using data extracted from HUMS as seen in works by Leoni et al. [22] and Wu et al. [24], among others.

Classification categories for aircraft flight regimes range from simple to complex. Approaches such as Wu et al. [25] pictured in Fig. 2.5, consist of simple and composite regimes. In the top row, the simple regimes such as roll and take off consist of a change in movement along one axis. The complex regimes on the second row include a change in movement along more than one axis such as lazy eight and split-s.

While Wu et al. used a simple classification set, other research such as Berry et al. [14] split their classification efforts across 141 different flight regimes grouped into ten super regime groups plus another class for anything that was not classified into one of the 141 defined regimes.

However, a given aircraft cannot necessarily perform every possible flight regime applicable to another aircraft. A fixed wing aircraft typically cannot hover and helicopters cannot go faster than the speed of sound. Therefore, each aircraft typically has a defined set of regimes for its own aircraft type and experts in the field of that aircraft understand the accepted flight regimes for the aircraft.

Leoni et al. [22], claim that single class classification of flight regimes is inferior to multi-class classification methods. They noted that contemporary flight regime classification approaches had difficulty effectively categorizing flight regimes when the aircraft was in transition from one to another. For example the transition state between straight and level flight into a turn. This problem is not new, Berry et al., [14] observed that their model also struggled to classify the transition states between flight regimes.

Modelling of vehicular dynamic movement is not unique to the MIL-STD-

1553B data bus or to aircraft in general. Research performed by Jensen et al. [26] investigated mass estimation based on ground vehicle movement. The data used for their analysis was pulled from two vehicles' Inertial Measurement Unit and the Controller Area Network Bus.

An aircraft flying in different regimes results in different information being transmitted over the MIL-STD-1553B bus. For example, an altimeter while flying in straight and level flight would be expected to maintain roughly the same altitude over a period of time. A rapid decrease in the altimeter reading could signal a potential intrusion into the bus. However, if the aircraft is landing, a decreasing altimeter reading is expected. Therefore, we theorize that flight regime aware intrusion detection models would be best suited to determine anomalous network traffic instead of generalizing across all flight regimes.

Table 2.1: Previous Experimental Results — Adapted from [22]

Research	Data Source	Accuracy	F_1
Berry et al. [14]	Real	0.762	-
Villa et al. [27]	Real	-	0.94
Leoni et al. [28]	Real	0.9732	0.97
Musso and Rogers [29]	Simulated	0.841	-
Şenipek and Kalkan [30]	Simulated	0.982	-
Wu et al. [24]	Simulated	0.999	0.999

Table 2.1 details previous experimental results on aircraft flight regime classification with methods used consisting of a wide range of techniques including Convolutional Neural Networks in Wu et al. [25], fuzzy c-means clustering in Leoni et al. [22] and elliptic basis functions in Berry et al. [14]. Leoni et al. [28] reported an F_1 score of 0.97 with their approach with others reporting accuracy scores ranging from 0.762 [14] to 0.97 [28] on real aircraft data and 0.841 [29] to 0.999 [24] with simulated data.

2.3 Long-Short Term Memory (LSTM)

LSTMs are a form of Recurrent Neural Network (RNN) first published by Hochreiter and Schmidhuber in the 1990s [31]. LSTMs solved the gradient descent problem that had plagued leading RNNs of the time. LSTMs excel at applications that process large amounts of sequential data. A well known

application of LSTMs is AlphaStar [32], an LSTM trained to play and win against top ranking human StarCraft II players.

LSTMs are composed of three main gates: the input gate, the output gate and the forget gate. These gates are modified by activation functions that binds the values of the gates. Additionally, each gate and hidden state have weights attributed to them and a bias term to aid in the model's prediction capability [33]. A diagram of a LSTM can be seen in Fig. 2.6.

LSTMs have been used in previous research with MIL-STD-1553B data by Soucy and Harlow [11,34], further evaluated by Wrana et al. [15]. Additionally, they are extensively used in classification problems stretching across multiple research fields.

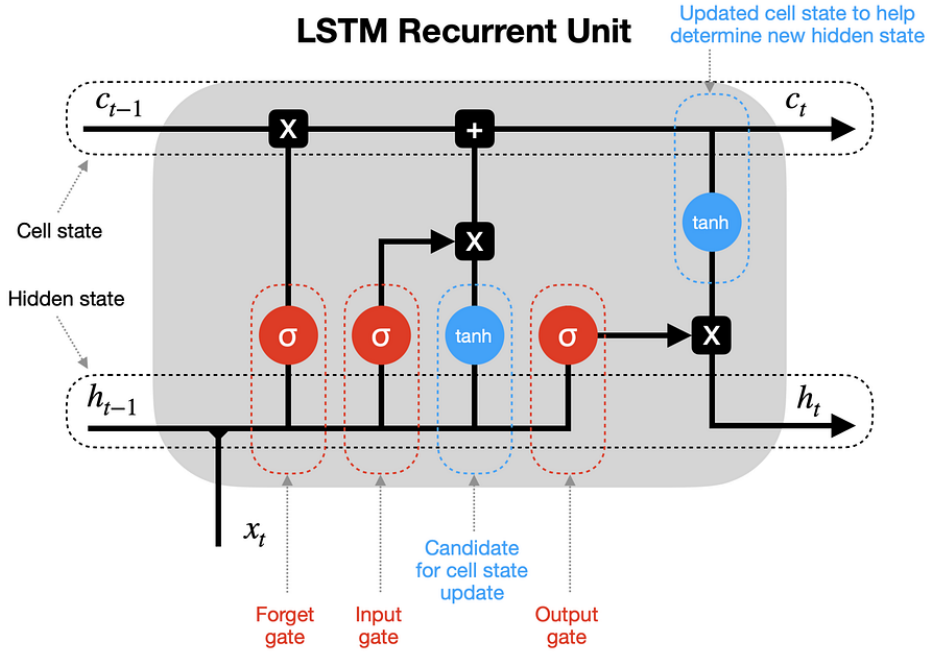

Fig. 2.6: LSTM Architectural Diagram [35]

In 2024, Beck et al. [33] published an updated LSTM model called xLSTM which addressed several technical limitations with the original model. These limitations include the inability to parallelize the computation of the model due to memory mixing, difficulty revising storage decisions when presented with similar data to the information already stored and limited storage capacities due to use of scalar vectors [33].

2.3.1 Extended Long Short-Term Memory (xLSTM)

The xLSTM model is a recent technique published by Beck et al. [33], as an improvement to traditional LSTMs. Through two key modifications to the original architecture. The LSTM block is replaced with two similar component blocks known as Scalar Long Short-Term Memory (sLSTM) and Matrix Long Short-Term Memory (mLSTM). These blocks are both modifications of the original LSTM architecture.

The first modification block, sLSTM, implements a new memory mixing model using an exponential function on the input and forget gates. Due to the tendencies to overflow, a new gate was added, called the stabilizer (or candidate) gate to combat this side effect. In addition, a normalizer state is added. Fig. 2.7 shows the architecture of the sLSTM block, highlighting the differences with traditional LSTMs. sLSTMs retain the drawback from LSTMs in that they cannot be parallelized. Since the output of the cell state is a scalar value with a single dimension, this component is called the sLSTM block.

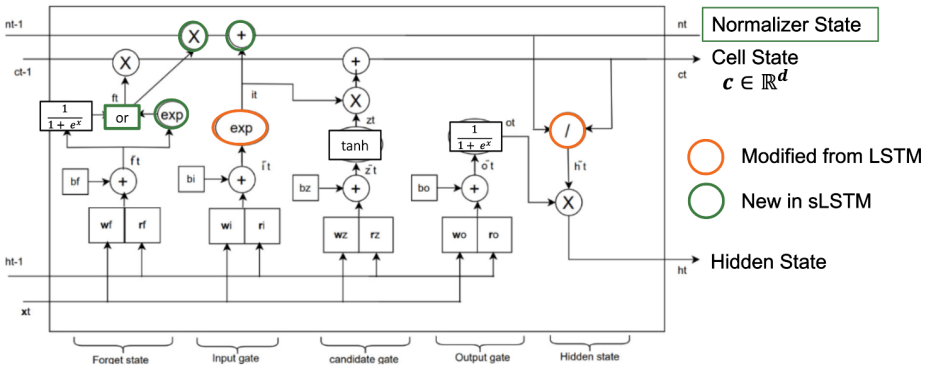

Fig. 2.7: sLSTM Architectural Diagram - Adapted from [36]

The second block, mLSTM is a change from the scalars used in traditional LSTMs, to a matrix and the incorporation of the covariance update rule. These changes introduce query, key, and value inputs. mLSTMs do set aside the ability to perform memory mixing, however gain the ability to parallelize the computation of the model, vastly improving computation time for this portion of the algorithm. Fig. 2.8 shows the architecture of the mLSTM. Since the output of the cell state is a *dxd* matrix, this component is called the sLSTM block.

The xLSTM architecture is flexible by design. Different configurations

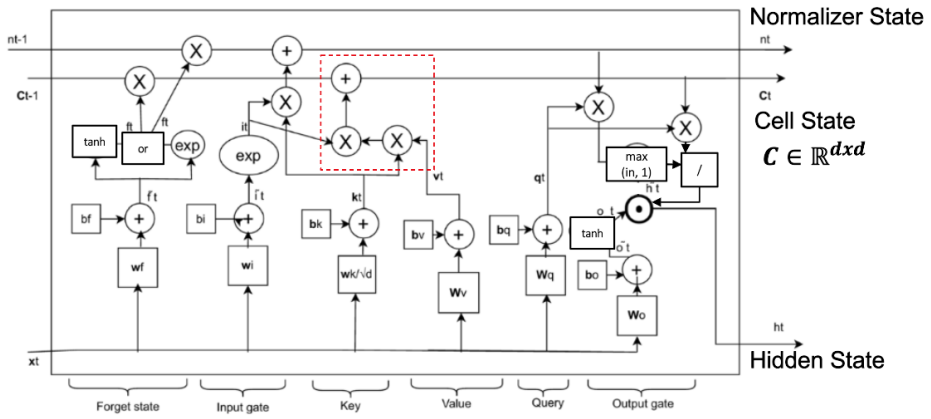

Fig. 2.8: mLSTM Architectural Diagram - Adapted from [36]

of sLSTM and mLSTM blocks can be organized into blocks, called stacks. Stacks can be set up to leverage the benefits of increased memory capacity and paralleization through the addition of mLSTM blocks and performance using sLSTM blocks to gain the benefits of the updated covariance rule and new memory mixing model. Beck et al. describes these configurations using the following notation:

$$[Num\ mLSTM\ blocks\ :\ Num\ sLSTM\ blocks]$$

where the number of each block can be zero or greater. When the number is zero, there are no instances of the zeroed block type within the xLSTM stack. The placement of the blocks can also be configured easily within the xLSTM block stack configuration.

Although the original paper by Beck et al., focuses on Large Language Models, research using xLSTMs into multi-class classification has already been performed by Kang et al. [37] and Sükei et al. [38] for classification of medical conditions. Kang et al. [37], classified 12 lead electrocardiograph readings into five classes and Sükei et al. [38], classified retinal disease into five different classes.

The xLSTM model possesses many hyperparameters which can be tuned in order to modify the performance and results of the overall model. These hyperparameters will be explored in detail in Sub-Section 2.4.

2.4 Hyperparameters

In this section we present a detailed break down of hyperparameters that we vary through the course of this work to find the set of hyperparameters leading to the best resulting weighted F_1 and accuracy scores. These values of the hyperparameters impact performance significantly with some configurations leading to unacceptable performance and others producing excellent results. In this research we consider the following hyperparameters:

1. Number sLSTM Blocks: Determines the number of sLSTM block components within the xLSTM stack. These blocks benefit from the new covariance update rule and memory mixing as described in Subsection 2.3.1.
2. Number mLSTM Blocks: Determines the number of mLSTM block components within the xLSTM stack. These blocks benefit from the increased memory capacity highlighted in Subsection 2.3.1.
3. Sequence Length: Determines the number of sequential MIL-STD-1553B messages to process at a time. A longer sequence length improves the contextual awareness of the model but also increases its training time due to the increased quantity of data.
4. Batch Size: Determines the number of groupings of MIL-STD-1553B message sequences to process per batch of the training cycles.
5. Number of Features: Determines the number of features, taken as MIL-STD-1553B words that are used to train the machine learning model. We display a listing of all words in Table A.1. Examples of these words include Data 32, Cmd 1. We expand upon selection of features in Subsection 3.2.2. An increased number of features typically improves classification performance, but similarly increases model training time.
6. Initial Learning Rate: The maximum rate that the model updates weights and biases of internal model components at the start of training. A value that is too large will increase training time and often lead to configurations that do not classify. An initial value too small frequently get stuck in local minimum instead of converging on the global minimum. The learning rate is decreased through model training to

In addition, we note the presence of several other hyperparameters which we elected to leave as static values. These include the number of nodes within the input linear layer which we set to the number of input features. The number of attention heads, and the maximum number of iterations before we stopped training the models.

Beck et al. [33] do provide some descriptions of xLSTM hyperparameters in their paper, but lack a complete summary of each hyperparameter. In order

to fully understand the impact of all applicable hyperparameters, we carefully examined the published xLSTM source code on their GitHub page [33]. We provide a listing of these hyperparameters in Table C.1.

2.5 Statistical Analysis Techniques

In order to measure the success, a metric that analyzes performance must be used to make that determination. It is common practice in machine learning implementations and research on the MIL-STD-1553B bus to use statistical analysis to measure performance of the model in supervised machine learning applications. Analyses such as [10,15,25] used metrics such as the ones detailed in this section, specifically the accuracy, macro F_1 and weighted F_1 scores. Fundamental to evaluation are four concepts that relate to the model's ability to correctly or incorrectly classify data samples. These four concepts are shown in Table 2.2, also known as a confusion matrix.

Table 2.2: Binary Confusion Matrix

	Predicted Value	
Actual Value	**True**	**False**
True	True Positive (TP)	False Negative (FN)
False	False Positive (FP)	True Negative (TN)

From the parameters in Table 2.2, metrics for statistical evaluation of performance can be derived. Four important performance metrics are listed in Equations 2.1, 2.2, 2.3, and 2.4 as described in Géron [39].

While frequently used in binary classification problems, equations for precision, recall, accuracy and F_1 scores can be expanded to multi-class settings.

Precision informs on the success rate of the model calculating a sequence is of class c really belongs to class c. Equation 2.1 details the formula for precision.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall on the other hand, speaks to the model's ability to properly classify all sequences of class c in the c class. Equation 2.2 details the formula for recall.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Accuracy is a metric that the model created is providing the correct output for a given input, expressed in Equation 2.3.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.3}$$

F_1 Score, otherwise known as the harmonic mean is a balanced score with focus on the model's precision and recall scores. This metric emphasizes that the correct class is given and that all data described as belonging to the class do in reality.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

Slightly modified equations are required for multi-class settings versus binary classification problems. These equations described and compared by Lee et al. [40] are the macro and weighted F_1 scores.

The macro F_1 score, as seen in equation 2.5, first calculates the F_1 score of each class before performing a simple average of the class-wise F_1 scores.

$$F_1^{Macro} = \frac{1}{\#Classes} \cdot \sum_{i=1}^{\#Classes} F_1^{(i)} \tag{2.5}$$

The weighted F_1 score, described in equation 2.6 calculates a class-wise F_1 score for each class before multiplying the resulting F_1 score by the weight (proportion of the nth class to the whole dataset) of the class. The macro F_1 score minimizes the impact of heavily imbalanced datasets with a few classes with few samples that are poorly classified by the model. This weight function is detailed in equation 2.7.

$$F_1^{Weighted} = \sum_{i=1} w_i \cdot F_1^{(i)} \tag{2.6}$$

where w_i, the weight of the given class is defined in equation 2.7.

$$w_i = \frac{Samples\ in\ Class}{Total\ Samples} \tag{2.7}$$

The confusion matrix as presented in Table 2.2 can be expanded to multiple classes as seen in Fig. 2.9. The real classification resting along the y-axis with the predicted class resting along the x-axis. A sample is considered to be classified properly if it lies on the primary diagonal. In this setting, false positives are all values within the column which are not on the main diagonal, since they are all samples of different classes improperly classified to the actual

Regime	Group
1	Level Flight
2	Auto
3	Climb
4	Dive
5	Partial Power Decent
6	Pullout
7	Pushover
8	Side Flight
9	Side Slip
10	Turns
11	Other

Input Class	Output Classification (%)										
	1	2	3	4	5	6	7	8	9	10	Unk
1	97.85	0.00	0.00	0.02	0.02	0.59	0.37	0.16	0.08	0.00	1.37
2	1.05	49.18	0.00	0.00	1.11	1.87	2.77	0.00	0.00	0.00	44.96
3	0.00	0.00	97.09	0.00	0.00	0.38	0.00	0.00	0.00	0.33	2.59
4	0.00	0.00	0.00	94.34	0.00	70.47	75.28	0.00	0.00	0.00	0.46
5	3.36	1.58	0.00	2.87	94.95	17.60	21.76	0.00	0.00	0.00	1.63
6	9.40	0.00	0.00	9.24	0.01	73.33	18.31	0.00	2.69	1.02	14.82
7	3.74	0.00	0.00	31.35	0.05	64.20	74.24	0.00	2.68	0.00	5.34
8	19.94	0.00	0.00	0.00	0.00	0.00	0.00	72.17	0.00	0.00	7.89
9	4.40	0.00	0.00	1.05	0.00	58.37	13.85	0.00	95.23	0.00	0.16
10	4.59	0.00	0.04	2.33	0.03	37.68	4.27	0.00	0.40	33.12	34.97
11	18.36	0.52	0.33	5.58	0.99	15.44	9.56	4.94	0.32	0.00	56.85

Fig. 2.9: Sample Multi-Class Confusion Matrix [14]

class. False negatives are all values within a given row which do not lay on the primary diagonal as these class samples were all classified incorrectly by the model.

In order to compare performance of the machine learning model produced in this research to previous research, both the accuracy, weighted F_1 and macro F_1 scores will be calculated and compared to previous work. This allows for meaningful comparisons to be made.

2.6 Deficiencies in Research Space

Previous research on the MIL-STD-1553B data bus can be broken down into four categories which are listed in Table 2.3.

The first category covers summaries of attack vectors that are present on the MIL-STD-1553B bus such as Lounis et al. [3] who highlight 20 different attack vectors the MIL-STD-1553B is vulnerable to, and demonstration of specific attacks against the bus such as Paquet [4]. These vulnerabilities include man in the middle attacks, eavesdropping and data corruption attacks [3].

Table 2.3: Categories of Previous Research on MIL-STD-1553B Data Buses

Category	Description	Previous Research
1	Demonstrating Attack Vectors on MIL-STD-1553B	[3, 4, 6]
2	Flight Regime Agnostic MIL-STD-1553B Defence	[8, 9, 11–13, 16, 34]
3	Partially Aware Flight Regime MIL-STD-1553B Defence	[10, 15]
4	Partial Use of MIL-STD-1553B Traffic for Flight Regime Classification	[14]

The second category encompasses the largest body of research and includes intrusion detection research with the objective of anomalous traffic or finding signs of attacks against the MIL-STD-1553B bus. Multiple models have been proposed to include simpler techniques such as Naive Bayes in He et al. [16] to various forms of LSTMs in Soucy [11], Levy et al. [8] and others.

The third, which consists of two pieces of research by Wrana et al. [15] and Elsayed et al. [10], which have the same objective as the second category, but where datasets used are structured such that each dataset contains a single similar flight regime. Namely, cruise and takeoff. These two papers used the same datasets.

There is only one research effort in the final category, by Berry et al. [14]. They installed additional hardware, accelerometers and tachiometers, onto the aircraft that extracted data from the MIL-STD-1553B bus and captured input from accelerometers and tachometers integrated into this installed device. Instead of using full MIL-STD-1553B messages, they collected a set of 20 continuous features which was processed into flight regimes. This approach is similar to extracting features from HUMS as there is a large overlap between the features that could be extracted from the HUMS and MIL-STD-1553B. These extracted features included pitch, roll, and main rotor speed.

The research contained in categories one and two consisted entirely of work that does not consider potential differences in MIL-STD-1553B traffic over different phases of flight. While MIL-STD-1553B buses can have other uses, in aircraft they are typically used to transmit avionic or mission related information. Therefore, reading from the avionics bus would lead to varying traffic based on changes in what the aircraft is doing. An aircraft in straight and level flight will be subjected to minimal changes in altitude, whereas in a landing sequence, a steady decrease in the altitude would be expected.

Category three consists of research which incorporates some awareness of different flight regimes into their work with MIL-STD-1553B traffic. Wrana et al. [15], sought to conduct intrusion detection on the bus with two dif-

ferent datasets, one in a cruise state, the second in a takeoff. The datasets were processed separately but demonstrated differences in intrusion detection performance with the models evaluated.

Although Berry et al. [14] classify aircraft flight regimes, they did not process full MIL-STD-1553B message traffic.

A gap in research exists in this space; Wrana et al. [15] discovered that different flight regimes could impact intrusion detection results with MIL-STD-1553B traffic. With modern performing models for intrusion detection the difference was variable, an F_1 score difference of 0.02 with their model and 0.03 with the model proposed by Génereux et al. and 0.04 in comparison to traditional LSTMs. However, for He's [16] framework the difference is 0.07. In order to be able to continue research in this domain, a technique must be developed in order to classify data stemming from an aircraft's MIL-STD-1553B data bus into a flight regime. With the advent of fly-by-wire aircraft, the transmission of flight control inputs over electronic buses such as MIL-STD-1553B make the task of securing the bus of critical importance.

2.7 Summary

To summarize, this chapter covered the concepts required to delve deeper into this work. In addition, this chapter presented foundational research upon which this work builds upon as well as key concepts such as the MIL-STD-1553B protocol, aircraft flight regimes, the model to accomplish the aim, xLSTMs, and the statistical analysis methods to evaluate success. Finally, a gap in contemporary research was presented which this work covers in detail over the following chapters.

3 Methodology and Design

Aircraft flight regime modelling for a human is a relatively straight forward task. Humans can intuitively describe how the aircraft is moving through our senses. For those with experience in the field, they may apply a set of experience based rules to make a refined definition. However, looking at an aircraft data bus, the problem no longer becomes one's intuition, but instead an attempt to decipher a network's rapid communication in order to learn what the aircraft is doing.

This chapter will resolve the deficiencies in current research identified in Section 2.6 by detailing a deep learning model that classifies MIL-STD-1553B message traffic into distinct flight regimes. We divide the remainder of this chapter into three sections, the data preparation phase, the model construction and training phase, and finally, the verification and validation phase.

3.1 Phase 1: Data Preparation

Phase 1 consists of the steps that we take to prepare the datasets for classification by the deep learning model that we develop in Phase 2. We assess this phase as complete once we label and format the data such that it can be processed by the deep learning model.

This research uses two distinct datasets. The first, a simulated dataset by Mukherjee [17]. The second, a dataset recorded on a RCAF CP-140 Aurora aircraft [18], provided by DTAES.

We do not believe models trained on one dataset can classify data from the other dataset due to the difference in labelling and composition of the underlying RTs on the distinct MIL-STD-1553B networks. Components of different versions installed on each network possess their own instrumentation control design documents which dictate the specific format, frequency, and content of messages that each RT expects to send and receive.

As described in Chapter 2, MIL-STD-1553B messages consist of no more

than two command words, two status words and 32 data words per message. All words are 20 bits long, however in both datasets, the sync and parity bits were dropped leaving 16 bits of information per word.

The two datasets, provided as Comma-separated values (CSV) files, contained MIL-STD-1553B messages by row and MIL-STD-1553B words displayed by column. The datasets contain some words spread across multiple columns which we reconstructed into full words. This phase reduces the datasets to the 36 + 2 features listed in Appendix A. The last two features are only used in Phase 1 for dataset labelling. These features contain the different words transmitted over the bus. The reduction preserves the timestamps in order to label the data, but we discard them prior to training as these timestamps are not messages transmitted over the bus. We processed the two datasets separately.

While individual message sequences as defined by the MIL-STD-1553B standard [2] can be of variable word length, both datasets provided values of `0x0000` for empty values. This occurred frequently since few messages sent all 32 data words possible by the specification. We kept the zero values so that all messages were of consistent length to facilitate processing by the deep learning model.

The final step involves splitting the datasets into fixed length message sequences and dividing the data into training, testing, and validation sets. Due to class imbalances, we first separate the datasets by flight regime before splitting into the training, testing, and validation sets. This step proves essential to ensure that all flight regimes are present across all sets. Pure random selection frequently left small classes out of the validation sets.

3.1.1 Mukherjee Dataset

The dataset by Mukherjee [17], consists of three simulated flights recorded on the RMC Cyber Trainer, a flight simulator connected to MIL-STD-1553B bus simulator. The dataset contains both recorded MIL-STD-1553B message traffic and a screen recording of the entire flight in the aircraft cockpit with the heads-up display and various instrumentation visible. The dataset contains a further three flights which execute attacks against the simulated aircraft bus. We elected to exclude these flights since they are out of scope for aircraft flight regime classification.

Each flight started with an aircraft start up sequence on the ground and ended with a shutdown sequence on the ground either on the runway or taxied off. The first flight consisted of an aircraft intercept in the Canadian arctic with an air-to-air refuelling cycle. The second, contains a fighter response

and dog fight with enemy aircraft over an airfield. The last flight was a bombing mission of an airfield from a carrier off the coast of British Columbia, weaving down mountainous valleys flying very close to the surface of rivers. The missions lasted 135, 12, and 30 minutes respectively.

This dataset consists of approximately two million MIL-STD-1553B messages that we classify into seven self-annotated flight regimes. We perform this classification by observation of instrumentation visible on the screen recording and annotating video timestamps. We add the labels to the dataset by cross-referencing the video recording timestamps and the timestamps annotated with the MIL-STD-1553B messages. Due to the resolution of timestamps on the video recording and timestamps of data, we note a slight discrepancy in the transition state labelling. We attempted to minimize the error introduced by observing the video and labelling transitions within a second of change perceptible to the observer.

These seven selected flight regimes are as follows:

1. Start or Shutdown: aircraft is not moving, and the operator is turning on or off various aircraft systems at the beginning or end of flight.
2. Take Off or Landing: aircraft is increasing or decreasing speed with weight on wheels on the runway.
3. Ascend: aircraft increasing altitude at a rate greater than 1000 feet per minute.
4. Descend: aircraft decreasing altitude at a rate greater than 1000 feet per minute.
5. Roll or Turn: observed change in yaw or roll, excluding minor corrections due to aircraft drift.
6. Level: aircraft is not on the ground and not in another state, or for the case of ascend or descend, exceeding the threshold for a short period of time (i.e. flight controls bumped.)
7. Taxi: aircraft is moving on the ground, excluding time during take off or landing regime.

We selected these flight regimes through the following mix of factors: First, they were easy to segment by viewing a recording of the cockpit instrumentation (air speed indicator, roll indicator, altimeter, navigation instrumentation, etc.) Second, they composed a critical portion of flight (take off and landing). Third, we identified a selection of flight regimes that we suspect through domain knowledge, would be more likely to produce identifyable flight regimes.

The classes of the dataset are imbalanced. Samples skewed heavily to having more samples in the Roll/Turn, Level, Descend and Taxi classes, and least in the Takeoff/Landing, Ascend, Start/Shutdown classes. Fig 3.1 provides an overview of the class imbalance within the Mukherjee dataset in a logarithmic scale.

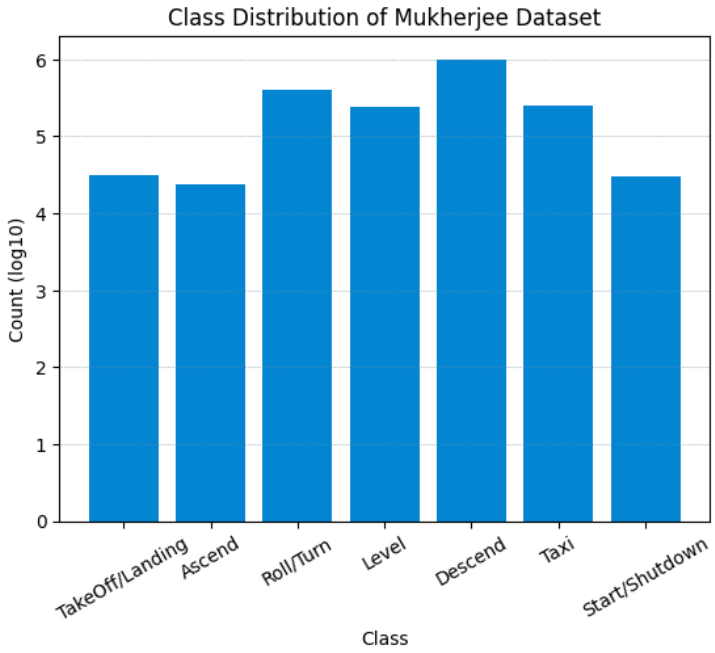

Fig. 3.1: Distribution of Flight Regimes in Mukherjee Dataset

3.1.2 CP-140 Aurora Dataset

The CP-140 Aurora dataset [18] consists of seven flights over the span of several weeks across different hours of the day. Each flight used the same aircraft and lasted several hours in duration consisting of an aircraft start, takeoff, one or more missions, landing, and shutdown. During one flight, the recording terminated in flight during the mission phase.

A general time annotated list of activities and/or missions the aircraft performed accompanied each flight with a resolution to the nearest minute. The label descriptors are either the mission the aircraft was fulfilling or a description of aircraft movement. We combine several annotations into larger clusters, for example, certain sub-missions into larger mission groups or similar types of aircraft movement. This is in contrast to the self-annotated flight regimes performed in the analysis of video recordings of the Mukherjee flights.

Due to releasability restrictions on this dataset, certain information including the names of the various flight regimes and other details on the dataset cannot be published.

In total, we reduce the number of aircraft flight regimes to 15. Fig. 3.2

displays the class flight regime distribution with a logarithmic scale. It is important to note that, this dataset is heavily imbalanced with certain classes being present over 100 times more than other classes.

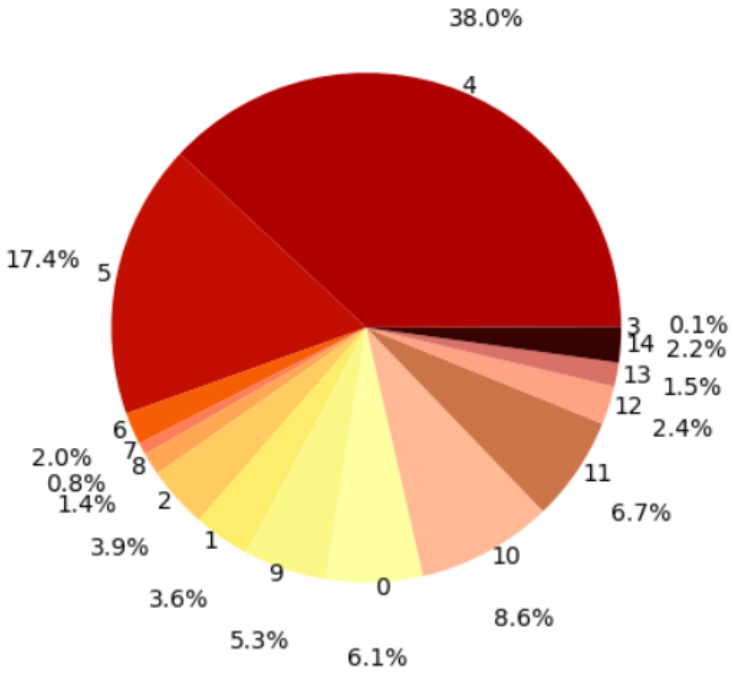

Fig. 3.2: Distribution of Flight Regimes in CP-140 Aurora Dataset

In addition, following a recommendation by G. Merkoske, the individual who processed the raw CP-140 Aurora data recorded from the flights (G. Merkoske, private communication, Nov. 19, 2025), led us to discard any data recorded on the MIL-STD-1553B bus more than 30 minutes before takeoff or 30 minutes after landing.

The CP-140 Aurora dataset contained roughly twenty-two times more samples than the Mukherjee dataset due to the length of time captured across the seven versus three flights. In addition, the CP-140 Aurora dataset contains more RTs than present in the Mukherjee dataset.

3.2 Phase 2: Model Construction and Training

Phase 2 focuses on four main aspects in order to construct and train the deep learning model in support of the aim. These areas include the construction of

a deep learning pipeline, feature reduction, hyperparameter tuning, and model training. These steps work together to improve the classification performance which we validate in Phase 3. We assess this phase as complete once we construct the xLSTM machine learning pipeline, and it outputs flight regime classification labels for a given input sequence of MIL-STD-1553B message traffic.

3.2.1 Deep Learning Pipeline Construction

The machine learning pipeline consists of three layers, with an xLSTM layer sandwiched between two linear layers. We break down the three principal layers of the model in Fig. 3.3.

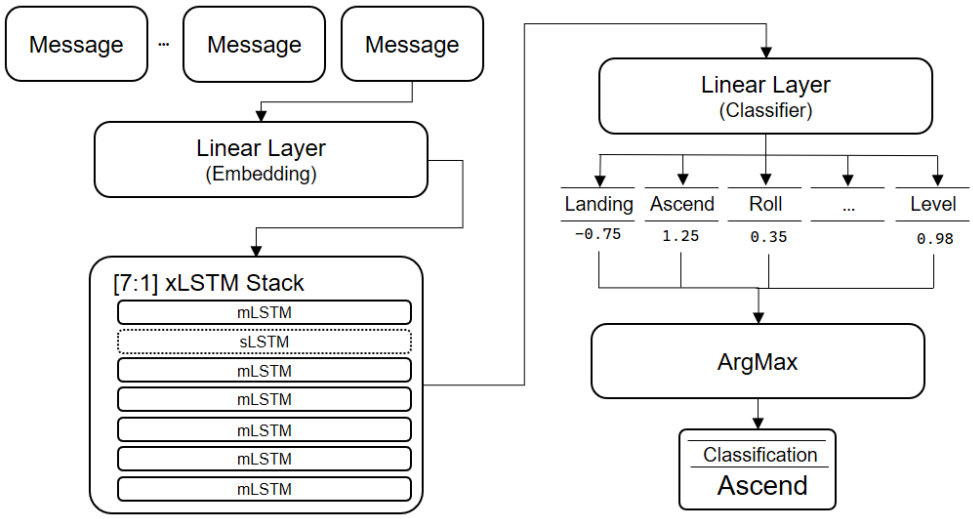

Fig. 3.3: Machine Learning Pipeline

The first layer, a linear layer, mirrors a technique from Alharthi et al. [36]. This layer ensures the input data respects the strict dimensionality requirement of the xLSTM stack. This layer also serves as an embedding layer for the data.

The second layer consists of an xLSTM stack with varying internal configurations described in Table 3.1. Current literature presents no established consensus of optimal ratios of sLSTM and mLSTM blocks [33,36,37]. Different authors use diverse configurations in their research. Table 3.1 duplicates the

xLSTM stack configurations utilized by Beck et al. [33] providing four xLSTM block stacks. We evaluate each configuration to determine which architecture yields optimal results in the output accuracy, macro F_1, and weighted F_1 score balanced against training time. According to Beck et al. [33], the sLSTM blocks' "memory mixing model enables to solve state tracking problems." The increased long term memory capabilities of the mLSTM block implies that a balance must be struck between the two to leverage the benefits of each xLSTM stack component.

The two first configurations listed in Table 3.1 use only one xLSTM block component to compare training time versus output classification performance. The third leverages one sLSTM and one mLSTM component to strike a balance between the benefits of both components. The fourth aims to increase classification performance by the presence of seven mLSTM blocks with their increased memory capacity to better learn long term dependencies. This final configuration quadruples the number of xLSTM components within the stack.

Fig. 3.3 depicts the [7:1] xLSTM stack configuration. For iterations that included a sLSTM block, it was always placed in the second position within the xLSTM stack.

Table 3.1: xLSTM Stack Configurations

Configuration	*mLSTM Blocks*	*sLSTM Blocks*
1	0	1
2	1	0
3	1	1
4	7	1

The final linear layer processes the output from the xLSTM layer into a single dimension vector with *num classes* elements called the class output vector. Subsequently, the model applies the *argmax* function to the class output vector in order to determine the output class.

The pipeline fed batches of data into the model. Each batch consisted of continuous sequences of *sequence length* MIL-STD-1553B messages. Messages consisted of words transmitted over the MIL-STD-1553B bus and padding as required. We label each sequence according to the label value of the last message in the sequence. The model outputs one prediction per input sequence which we compare against the true class to evaluate classification metrics.

We use the Cross Entropy Loss (CEL) function in order to update intrinsic parameters of the model after each batch using default parameters. The model uses an Adam optimizer to optimize the model using an *initial learning rate*

of *0.001*. After fifteen subsequent iterations without an improvement to the best loss calculated in the evaluation of the test set, the optimizer reduces the *learning rate* by a factor of ten. We define the *stopping criterion* such that the *learning rate* reaches 10^{-8} or exceeds 250 iterations. Few experimental cycles surpassed 200 iterations.

Once the model reaches the stopping criterion, we evaluate its performance by running the validation set through the model and applying statistical analysis techniques described in Section 2.5 for comparison. In order to account for lower performing initial seed values present within the deep learning model, we run each experimental configuration five times to determine the results.

3.2.2 Feature Reduction

Since this research is the first to classify aircraft flight regime traffic based solely on MIL-STD-1553B traffic, there are no known sets of features that produces optimal results. Geron [39] suggests reducing the number of features in machine learning applications to cover 95% of the covariance contained within the dataset. We define features within the context of this research as singular MIL-STD-1553B words. A reduction in the number of features used to train the model should reduce the training time. However, we must strike a balance between the amount of time it takes to train the models and the resulting classification metrics.

While Geron suggests the 95% covariance threshold, we elected to evaluate alternate thresholds to determine the impact of the number of features on model training time. We selected 6, 12, and 18 features to evaluate which represent 17%, 33% and 50% of the total 36 features present in the datasets. We considered other feature reduction techniques including Principal Component Analysis (PCA) as described by Geron [39] to reduce the number of features, but none of them produced successful results.

We provide a detailed overview of feature reduction results in Chapter 4.

3.2.3 Hyperparameter Tuning

The third portion of the machine learning pipeline creation was finding a suitable set of hyperparameters that balanced both the time that it took to train the model and classification accuracy and F_1 scores. In all experiments, the training of the xLSTM models is orders of magnitude slower than label prediction rate of the model once training is complete.

The deep learning pipeline we use in the course of this research contains several hyperparameters that we dial in order to increase the classification

outcome of the model. We perform the experimental evaluation of hyperparameters using the Mukherjee dataset due to its smaller size allowing for increased experimentation within given timeframes.

Table 3.2: Deep Learning Hyperparameters

Hyperparameter	Experimental Values
Num sLSTM Blocks	0, 1
Num mLSTM Blocks	0, 1, 7
Batch Size	256, 512, 1024, 2048, 4096
Initial Learning Rate	$10^{-3}, 10^{-4}$
Sequence Length	8, 16, 32, 64
Number of Features	6, 12, 18, 22, 36

Certain values of hyperparameters were limited to respect matrix multiplication rules within mLSTM cells. For example, the number of attention heads present within the xLSTM layer is required to be a multiple of the number of features given to the model. Default values for the number of attention heads is 2, which implies only even number of features could be selected for the model. For any other hyperparameter not listed in Table 3.2 or otherwise described elsewhere, we use default values for each component.

3.2.4 Model Training

In order to train the models, we broke the data down into blocks as described previously in this thesis and fed the data into the machine learning model. After each run, we logged relevant outputs such as saving the model state, listing execution times for various portions of the learning pipeline, and the resulting confusion matrix from the validation set alongside the output accuracy, weighted, macro, and micro F_1 scores. We set these results aside in order to evaluate performance of the model in Phase 3.

Due to data security constraints regarding the CP-140 Aurora dataset, we trained the models on different machines.

In Phase 2, we created a machine learning pipeline consisting of an input and output linear layer sandwiching an xLSTM layer of various stack configurations. We adjusted hyperparameters in order to improve model classification results and decrease model training time. The next phase evaluates model performance.

3.3 Phase 3: Verification and Validation

Phase 3 consists of two main efforts. The first, we verify that we properly constructed the xLSTM machine learning pipeline that we outlined in Subsection 3.2.1. The second, validating the experimental values we obtain in Phase 2 and comparing the results against previous bodies of research using common statistical analysis techniques highlighted in Section 2.5. The final portion consists of confirming or rejecting our aim statement.

3.3.1 Verification

With the recent publication of xLSTM in late 2024 [33] and the lack of inclusion of xLSTM in common Python machine learning packages such as PyTorch or Tensorflow, proper verification of the machine learning pipeline is of utmost importance. Beck et al. [33] published xLSTM as a pip library and on GitHub, though documentation is limited in the repositories.

To ensure proper construction, we verify the model's performance against previous research. We elected to re-implement the research by Kang et al. [37] for model verification. They use xLSTM to process the PTB-XL dataset [41] into five different diagnoses of heart conditions. These categories classify one class of normal readings and four classes of abormal readings, myocardial infraction, conduction disturbance, hypertrophy, and ST-T abnormalities. The PTB-XL dataset consists of 12 lead electrocardiograph measurements from roughly 22,000 patients. Once our machine learning pipeline demonstrates its ability to properly classify the results of the PTB-XL dataset into classes as described by Kang et al., the model will be considered fit for purpose. Since the model will be able to classify samples into different classes, the model should be able to classify MIL-STD-1553B traffic into different classes.

3.3.2 Validation

Validation of this work will be conducted through two primary statistical analysis techniques described in Section 2.5. These are accuracy and weighted F_1 score. We also provide macro F_1 scores for an alternate view of classification performance. We opt to use the weighted score as primary due to its superior ability to provide meaningful metrics on unbalanced datasets. Table 2.1 presents a summary of contemporary results in the domain of aircraft flight regime classification.

We assed Berry et al. [14] being the closest in scope to our research. However, major differences still exist due to the method of data collection sur-

rounding the MIL-STD-1553B bus by Berry et al. They obtained an accuracy value of 0.762 [14]. As described in Table 2.1, Villa et al. [27] obtained an F_1 score of 0.97 using real HUMS data. We elect to take Berry et al's. [14] accuracy result as a baseline for flight regime classification performance due to the highest similarity between their work and ours. Examining the synthetic data, Wu et al. [24] note accuracy and F_1 scores exceeding 0.999 but older approaches such as Musso and Rogers [29] provide an accuracy score of 0.841.

In order to successfully defend the aim, accuracy scores no lower than 0.762 obtained by Berry et al. [14] needs to be attained by the machine learning model proposed by this research since we assess their research to be the most similar to our research.

3.4 Summary

In this Chapter, we cover the design of our experimentation in order to achieve the aim statement outlined in Chapter 1. We break the work into three distinct phases. The first, preparing the two datasets for a machine learning application through processing of CSV files, screen recordings to ensure all MIL-STD-1553B messages are labelled. The second, focuses on construction of the xLSTM machine learning pipeline described in Fig. 3.3 and training the models to classify MIL-STD-1553B traffic into aircraft flight regimes. The third stage, verifies the construction of the xLSTM model by processing a different dataset through the model and comparison to previous research by Kang et al. [37] and validates the aim statement through comparison to aircraft flight regime classification research as detailed in Table 2.1.

4 Results

This chapter evaluates the xLSTM deep learning pipeline's ability to classify aircraft flight regimes solely from MIL-STD-1553B message traffic. We outline each phase of the pipeline in Chapter 2. The statistical analysis techniques we use to evaluate performance are detailed in Section 2.5, namely accuracy, macro F_1 score, and weighted F_1 score. We divide this chapter into five sections, the hardware and software design of the experimentation, verification, hyperparameter tuning, validation, and a final portion to discuss and summarize all findings of this chapter.

4.1 Hardware and Design

In this section, we explain the configurations of hardware and software used in the execution of this research.

Through the course of this research, we use two different machines due to departmental security policies limiting processing of the CP-140 Aurora dataset to certain devices. We list a selection of hardware and software composition in Table 4.1. The exact software version numbers we use varies based on the Compute Unified Device Architecture (CUDA) version of the underlying graphics card. We require no changes to the code base with the different systems due to the backwards compatibility of the libraries we use. We use System 1 to train the models for synthetic data in the Mukherjee [17] dataset, and we use System 2 to train the models for the real avionics data in the CP-140 Aurora dataset [18].

Table 4.1: Selected Experimental Hardware Configuration by System

Specification	**System 1**	**System 2**
Processor	AMD Ryzen 7 9800X3D	AMD EYPC 3954
Cores	8	Up to 100
Memory	64 GB DDR5	Up to 512 GB DDR5
GPU	NVIDIA RTX 5080	Up to 8x NVIDIA RTX A4000
Operating System	Ubuntu 24.04	Ubuntu 24.04
CUDA Version	12.0	8.6

We elect to use Python 3 as the programming language for this thesis since it maintains well-supported and documented machine and deep learning libraries. Additionally, Beck et al. [33] released xLSTM as a Python 3 pip package which we elected to use. The following libraries were used in the model training pipeline: NumPy (data processing), Pandas (data processing), PyTorch (deep learning backend), sklearn (evaluation techniques), Matplotlib (display graphics), xLSTM, mLSTM-kernels (GPU acceleration for xLSTM). We performed all experimentation within dedicated jupyter lab environments.

In addition, for the validation step, we used the Waveform Database (WFDB) library in order to perform pre-processing of electrocardiograph (ECG) readings contained in the PTB-XL dataset.

The differences in computational ability between System 1 and System 2 is significant. This difference makes meaningful comparisons of computational times between the two systems difficult. However, as noted in Section 3.1, the prevalance of both different labels and different numbers of labels makes comparison between the two datasets not equivalent.

4.2 Verification

This section provides details on the verification efforts we embody to ensure the deep learning framework detailed in Subsection 3.2.1 successfully produces labels for given input data. As noted in Subsection 3.3.1, we re-implement the research performed by Kang et al. [37]. This requires careful application of their pre-processing steps to the PTB-XL dataset [41] using only the 100 hertz samples, replicating their research. Once we completed the training of our implementation of their work, we compared accuracy provided in their paper and the experimental results we obtained and noted no major discrepancies

between the results. Therefore, this successful re-implementation of previous work confirms that the deep learning pipeline we constructed in Phase 2 is fit for purpose within the aim of this research.

4.3 Hyperparameter Tuning

This section provides details on our evaluation of hyperparameters in order to find optimal values for the training of our models. The hyperparameters we consider are: the number of mLSTM and sLSTM blocks present with an xLSTM stack, the number of sequential MIL-STD-1553B messages used to train the model, the size of the training block and the number of features (MIL-STD-1553B words) we use in model training. As we describe previously, we perform all hyperparameter experimentation on the Mukherjee dataset.

4.3.1 xLSTM Stack Configuration

We begin by exploring the xLSTM stack configuration as our first evaluated hyperparameter. This consists of the number of mLSTM and sLSTM blocks contained within the xLSTM stack. We list the four configurations we evaluated in Table 3.1 which Beck et al. [33] use in their research. Table 4.2 displays the results of this hyperparameter evaluation displaying the xLSTM stack configuration alongside model training time and resulting F_1 score.

Table 4.2: Hyperparameter Evaluation: xLSTM Stack Configuration

Configuration	Model Train Time (min)	Weighted F_1 Score
0:1	37.78	0.8392
1:0	60.66	0.7887
1:1	107.70	0.8385
7:1	479.88	0.8943

Table 4.2 demonstrates the [7:1] xLSTM stack configuration produced the highest weighted F_1 score at 0.8943. The [0:1] and [1:1] xLSTM stack configurations perform 6.2% worse than the optimal selection. The [1:0] xLSTM stack performs 11.8% worse than the [7:1] stack configuration.

We observe linear increase corresponding to the number of xLSTM stack components to the training time. In comparison to the [1:0] configuration, the [1:1] layout with twice the number of xLSTM components takes twice as long and the [7:1] configuration takes approximately eight times as long with eight

individual stack elements. Moreover, additional xLSTM components tends to increase the resulting weighted F_1 score produced by the given configuration. While the time increase is significant, we assess the improvement to the final F_1 score to be sufficiently significant to include all configurations in the experimental evaluation.

4.3.2 Sequence Length

The sequence length defines the number of subsequent MIL-STD-1553B messages that are processed by the model in order to learn short and long term dependencies contained within the message traffic an effort to classify aircraft flight regimes. We select four successive powers of two, ranging from 8 to 64 for potential sequence lengths. We make this selection on the basis of matrix multiplication requirements of mLSTM blocks, only powers of two were considered for the sequence length. Table 4.3 details results of the sequence length experimentation.

Table 4.3: Hyperparameter Evaluation: Sequence Length

Sequence Length	Model Train Time (min)	Weighted F_1 Score
8	30.47	0.4984
16	39.01	0.7463
32	40.59	0.8621
64	77.09	0.8532

We assessed the optimal message sequence length to be 32 messages long. Additional messages past 32 failed to improve the weighted F_1 score and increases training time drastically. Values lower than 32 decreased the overall training time, but came with significant performance hits to the calculated F_1 score. We assessed the experimental results with eight messages as not being able to properly classify the aircraft flight regimes.

4.3.3 Block Size

The block size speaks to the number sequences that the xLSTM pipeline processes at a time. We evaluated five different values for block size ranging from 256 to 4096. In a similar fashion to sequence length, due to the internal configuration of the xLSTM stack, the values are limited to powers of two. We present the findings of block size experimentation in Table 4.4.

Table 4.4: Hyperparameter Evaluation: Block Size

Block Size	Model Train Time (min)	Weighted F_1 Score
256	100.08	0.8394
512	53.65	0.8338
1024	45.47	0.8392
2048	49.20	0.8385
4196	74.75	0.8327

Table 4.4 demonstrates the optimal value for block size to be 1024 sequences of messages as it results in the highest F_1 score. Overall, the block size did not impact the performance of the model regarding the resulting F_1 score, but did impact the length of time that the models took to train.

4.3.4 Feature Selection

The number of features selected was the final hyperparameter evaluated. Geron [39] suggests the optimal number of features is one that captures 95% of the covariance intrinsic to the data in the dataset. We calculated the covariance within the two datasets and found that we required 19 features in the CP-140 Aurora dataset and 22 features in the Mukherjee dataset to attain this threshold. Noting this difference, we considered a fixed value of 22 features across both datasets to reach the recommended threshold by Geron in order to permit equivalent comparisons between the two sources of data. It is important to note, the 22 features are not identical across both datasets. The list of which features that we selected in each dataset are found in Table A.1 for the Mukherjee dataset.

Table 4.5 details the results of our experimentation with different configurations of features.

Table 4.5: Hyperparameter Evaluation: Feature Selection

Num Features	Model Train Time (min)	Weighted F_1 Score
6	27.90	0.4915
12	25.76	0.5539
18	34.37	0.7347
22	44.09	0.7797
36	36.37	0.8327

Table 4.5 demonstrates a gradual increase of final weighted F_1 score as the number of features increases. While the 95% covariance threshold suggested by Geron does lead to suitable results in the final F_1 score, the increase in this research to all 36 features squeezes out an additional 6.8% to the F_1 score. These runs also demonstrated an increase to training time with the increase of features except with 36 features. We attribute this to rapid model convergence which decreases the number of iterations required to reach the stopping criteria.

This section covered the hyperparameters that we use in the scope of this research to classify aircraft flight regimes from MIL-STD-1553B message traffic. While the configuration of the xLSTM stack and the number of features used will vary, we use a consistent block size of 1024 and a sequence length of 32 messages.

4.4 Validation

This section presents experimental results classifying MIL-STD-1553B message traffic into aircraft flight regimes. We evaluate the ability of various configurations of xLSTM stacks and number of features input into successive model training attempts using the hyperparameters determined to produce the best classification results. The selected hyperparameters are a block size of 1024, a sequence length of 32 messages, and an initial learning rate of 10^{-3}. This evaluation is compared to results of other bodies of work attempting to perform aircraft flight regime classification presented in Table 2.1. It is important to note that other bodies of research do not identify which type of F_1 score that they use.

We provide a table for each dataset detailing training time, testing time, accuracy, macro F_1 score, and weighted F_1 score at Figs. 4.7 and 4.8 respectively. We present a graphical multi-class confusion matrix for the best classification performance of each dataset with the confusion matricies for the other configurations presented in Appendix B. For each of the confusion matricies, the real classes are presented along the y-axis and the predicted classes are listed along the x-axis. A successful classification occurs when the model labels given input along the primary diagonal.

We define an experimental run as being one full training cycle of the xLSTM stack with a selected number of features. Each experimental run is conducted five times. We determined an experimental run to be successful if both of the following criteria are met:

1. The training loss rate converges towards zero. Experimental runs with

lower training loss rates report higher scores across each evaluated metric.

2. The macro F_1 score achieves at least 0.45. In unbalanced datasets, weighted F_1 score can report inflated scores while classifying most elements in the largest class. We note that 0.45 was the observed threshold were classification shifted from classification of everything in one class to a distributed classification.

Table 4.6 provides a summary of previous aircraft flight regime classification research that we will compare our results against.

Table 4.6: Summary of Previous Experimental Results — Adapted from [22]

Research	Data Source	Accuracy	F_1
Berry et al. [14]	Aircraft MIL-STD-1553B+	0.762	-
Villa et al. [27]	Aircraft HUMS	-	0.94
Leoni et al. [28]	Aircraft HUMS	0.9732	0.97
Musso and Rogers [29]	Simulated	0.841	-
Şenipek and Kalkan [30]	Simulated	0.982	-
Wu et al. [24]	Simulated	0.999	0.999

4.4.1 Mukherjee Dataset

The Mukherjee dataset [17], consists of roughly two million unique messages extracted from a simulated MIL-STD-1553B data bus across seven different flight regimes as detailed in Section 3.1.1. Table 4.7 provides an overview of experimental results obtained for this dataset. Experimental runs marked with a "*" failed one or both successful experimental run criteria.

Table 4.7 notes the [7:1] xLSTM stack with 12 features yields the highest weighted F_1 score at 0.9019. This set up also produces the highest accuracy and macro F_1 scores on this dataset. The [7:1] xLSTM configuration performs better in all cases to any of the other three xLSTM block stacks. However, this comes at a fairly high training cost of 372 minutes to train the [7:1] models with 36 features. In comparison, the [0:1] xLSTM stack with 36 features trained in 38 minutes on average. This implies that the [7:1] xLSTM stack takes almost ten times the time to train compared to the simpler [0:1] xLSTM stack.

We note that additional features typically imply an increase to the length of model training, but the xLSTM block stack has a significantly larger impact on model training times. The addition of each sLSTM or mLSTM stack element increases training time roughly linearly. While testing times, or label

Table 4.7: Mukherjee Dataset Results

xLSTM Config	Features	Training Time (min)	Test Time (s)	Accuracy	F_1 Macro	F_1 Weighted
0:1	6*	47.53	1.74	0.6341	0.3348	0.4915
	12*	47.48	1.60	0.6153	0.4117	0.5539
	18	50.87	1.95	0.7424	0.6329	0.7347
	22	34.36	1.70	0.7860	0.6894	0.7797
	36	37.78	2.10	0.8408	0.7654	0.8392
1:0	6	45.34	1.84	0.7288	0.5540	0.7105
	12	45.21	1.70	0.7823	0.6789	0.7739
	18	49.60	1.95	0.7929	0.7244	0.7902
	22	54.53	2.05	0.8133	0.7458	0.8124
	36	60.66	3.08	0.7928	0.7112	0.7887
1:1	6	86.30	2.25	0.7311	0.5380	0.7124
	12	123.57	1.93	0.7893	0.6897	0.7893
	18	114.92	2.35	0.7912	0.7145	0.7864
	22	108.76	2.62	0.8233	0.7559	0.8218
	36	107.70	2.96	0.8403	0.7695	0.8385
7:1	6	377.87	5.54	0.8598	0.8124	0.8564
	12	322.90	4.79	0.9032	0.8798	0.9019
	18	316.54	5.87	0.8784	0.8375	0.8778
	22	362.55	5.93	0.8633	0.8209	0.8621
	36	479.88	8.10	0.8948	0.7928	0.8943

* denotes failed experimental run

prediction time, increase with additional xLSTM stack components, the rate where increase is not both the time the model takes to predict labels in the testing phase increasing. Rather, it increases by a factor of four from the [0:1] to the [7:1] xLSTM stack configurations.

Fig. 4.1 displays the confusion matrix demonstrating aircraft flight regime classification performance with 12 features and the [7:1] xLSTM configuration. The figure displays nearly all elements correctly classified along the primary diagonal. The class with the highest incidence of false positives, the Level class, can be partially attributed to the transition state labelling between regimes since the aircraft would typically transition from the level state to non-level states and vice versa and not between non-level states. This discrepancy could be minimized finer timing resolution on transition states or adoption of multiple class classification instead of single class classification as suggested by Leoni et al. [28].

In iterations with low number of features, the presence of a mLSTM led

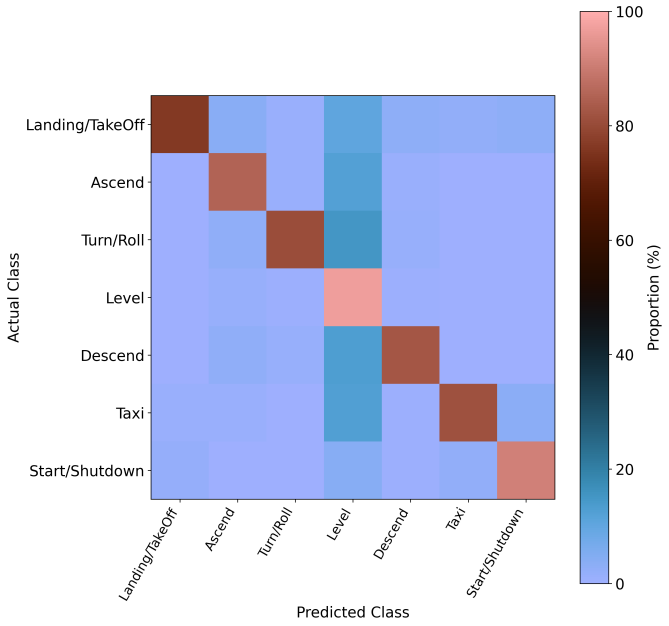

Fig. 4.1: [7:1] xLSTM, 12 Features Confusion Matrix

to better classification performance with the [1:0] stack and [1:1] stack using 6 features having similar metrics across accuracy, macro and weighted F_1 scores. The low macro F_1 with a significantly higher weighted F_1 score demonstrates that there are several small flight regime classes which are not classified properly. In contrast, with all features, the presence of a sLSTM stack element leads to higher performance with the [0:1] and [1:1] stack with 36 features proving almost identical across the three metrics evaluated.

This framework demonstrates an accuracy value of 0.9032, which does improve upon the work by Musso and Rogers [29], but does not achieve classification results obtained by other bodies of work such as Senipek and Kalkan [30] or Wu et al. [24]. As noted previously, these datasets are pulled from HUMS vice consisting of MIL-STD-1553B messages. The differences between the data sources prevents direct equal comparisons since HUMS provides clean and well-segmented data, while the data from the MIL-STD-1553B can mean different things depending on predicated messages.

The accuracy scores for all but four iterations described in Table 4.7 surpass the accuracy result of 0.764 obtained by Berry et al. [14] which we assess

as the closest body of work as we both pull at least some traffic from the MIL-STD-1553B bus. The four iterations which do not pass are the [0:1] xLSTM configuration with 6, 12 and 18 features and the [1:0] xLSTM configuration with 6 features.

Based on the results presented in this subsection, we conclude that this deep learning model can classify aircraft flight regimes using solely MIL-STD-1553B message traffic.

4.4.2 CP-140 Aurora Dataset

The CP-140 Aurora dataset [18], consists of roughly 44.3 million unique messages extracted from a flying RCAF CP-140 aircraft across 15 different flight classifications. Due to data releasability restrictions imposed upon the dataset, we limit the publication of certain details surrounding this dataset. We explain the composition of this dataset in Section 3.1.2. This classification achieved great success; we display a confusion matrix of the best results in Fig. 4.2. We provide the rest of the confusion matricies in Appendix B.

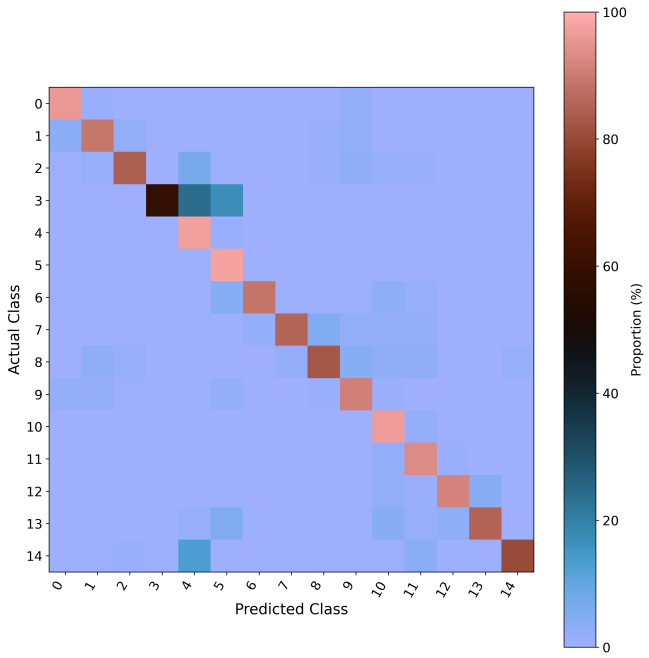

Fig. 4.2: [7:1] xLSTM, 12 Features CP-140 Aurora Confusion Matrix

The confusion matrix displays the vast majority of elements along the

Table 4.8: CP-140 Aurora Dataset Results

xLSTM Config	Features	Training Time / iter (min)	Test Time / iter (s)	Accuracy	F_1 Macro	F_1 Weighted
0:1	6*	11.77	78.62	0.5135	0.1740	0.4476
	12*	11.11	68.17	0.4481	0.1745	0.3490
	18*	12.55	84.62	0.7283	0.3391	0.6874
	22	12.74	86.18	0.8201	0.5128	0.8039
	36	12.34	82.68	0.9237	0.7540	0.9225
1:0	6	14.75	99.16	0.7540	0.5336	0.7403
	12	14.13	92.53	0.9162	0.7959	0.9153
	18*	14.89	104.16	0.3910	0.0616	0.2444
	22*	14.98	106.14	0.3956	0.0687	0.2514
	36*	17.81	121.64	0.3999	0.0847	0.2727
1:1	6	24.60	119.35	0.7710	0.5880	0.7626
	12	23.58	104.31	0.8858	0.7346	0.8844
	18	25.45	128.56	0.7676	0.4463	0.7450
	22	25.55	131.19	0.8023	0.4939	0.7898
	36	28.22	139.17	0.9205	0.7515	0.9196
7:1	6	102.37	385.81	0.8404	0.6789	0.8385
	12	99.08	340.13	0.9470	0.8887	0.9468
	18*	103.28	398.51	0.5832	0.1617	0.4918
	22*	103.31	400.45	0.4149	0.0939	0.2864
	36	124.07	502.13	0.9210	0.7599	0.9205

* denotes failed experimental run

primary diagonal which indicates a high degree of success. We note the model had some difficulty in classifying the flight regime labelled three. Class 3 is significantly smaller than all other classes, 10 times smaller than the next; well over 100 times smaller than the largest class. This distribution is displayed in Fig. 3.2.

Table 4.8 provides a summary of experimental results we observe classifying aircraft flight regimes contained within the CP-140 Aurora dataset. In a similar fashion to the Mukherjee dataset, the highest performing xLSTM setup is [7:1] xLSTM block stack with 12 features. We record a weighted F_1 score of 0.9468 and an accuracy score of 0.9470. The [7:1] xLSTM stack demonstrates the highest weighted F_1 scores across all number of features, displaying the same increasing trend with more components typically increasing the resulting performance metrics. However, the magnitude of the difference is much smaller in comparison to performance observed on Mukherjee's synthetic dataset.

Experimental results prove that the highest weighted F_1 scores are ob-

served with 12 or 36 features. The [0:1] and [1:1] xLSTM stacks having the highest F_1 score with 36 features and the [1:0] and [7:1] xLSTM stacks noting highest values at 12 features.

The loss rates while training on the CP-140 Aurora dataset typically converged within 50 iterations, with very minor improvement to the loss upon subsequent iterations and with a large portion converging within ten iterations. Since we set the stopping criteria based on performance of the Mukherjee dataset, this model spent a lot of time not impactfully training a stricter stopping criteria than could have been set. In order to offset this difference, we provide the average training and testing times per iteration in Table 4.8 which we recommend multiplying by 50 iterations for total training time of the given xLSTM configuration.

Several configurations failed to yield successful training results as defined by the two successful experimental run criteria previously defined. When the model fails, as denoted by a "*" in Table 4.8, typically all values are classified into Flight Class 4. This behaviour can be visualized in Fig. B.22. This class is by far the largest class in the dataset and becomes the default when the model does not demonstrate the ability to classify the flight regimes. This classification failure occurs when the loss does not converge to zero. Geron [39] references Bengio [42] who states that "the optimal learning rate is usually close to the largest learning rate that does not cause divergence of the training criterion, [...]" Our decision to perform hyperparameter selection on the Mukherjee dataset led to a selection of a suboptimal initial learning rate for the CP-140 Aurora dataset. The effect implies that several iterations failed to have their loss rates converge prior to hitting the stopping criterion.

We do note that while the macro F_1 score of the [0:1] xLSTM stack with 18 features is below the threshold, the resulting accuracy, weighted F_1 score and confusion matrix appear to indicate a moderately succesful experimental run.

Experimental timing results show that the training time for the model is significant. At 50 iterations, it takes 4954 minutes (83 hours) to train the [7:1] xLSTM stack with 12 features. In comparison, the [0:1] xLSTM stack with 36 features trains in 707 minutes (12 hours) for a similar weighted F_1 score and accuracy score. The results are better with the [7:1] stack in comparison to [0:1] stack and the [0:1] stack does have a considerable macro F_1 score drop which implies it does not classify the smaller classes to the same degree as the higher performing [7:1] stack. However, depending on the amount of available resources, the [7:1] stack may train too slowly and the [0:1] stack is better for applications that require to quickly train a new classifier while maintaining a high accuracy and weighted F_1 score. It is important to note that class

prediction remains quick. Predicting class labels for hours of data in single digit minutes with the more complex models taking longer scaling based on the number of xLSTM components present linearly.

This model, trained against the CP-140 Aurora dataset achieves an accuracy score of 0.9470 with the slower [7:1] xLSTM stack configuration and 0.9237 with the [0:1] stack. These two configurations far exceed Berry et al.'s [14] 0.764 accuracy result which we assess as being the closest in scope of work to this research in Chapter 2. This approach also manages to surpass Villa et al.'s [27] technique surpassing their F_1 score by 0.01. We note that Villa et al. do not mention which type of F_1 score they use. However, Villa et al. do not mention which type of F_1 score they use. We observe that Leoni et al. [28] outcomes exceed ours, however they are pulling their data from HUMS.

With the results discussed in this subsection, we assess that this deep learning framework is able to classify aircraft flight regimes from data originating solely from the MIL-STD-1553B data bus.

4.5 Discussion

In this section, we discuss several findings and observations made through this research.

In order to ensure timely completion of this research, we elected to perform all of our hyperparameter tuning on the synthetic dataset by Mukherjee. We basedd this decision on the fact we had higher confidence that we labelled the dataset correctly, on its smaller size and because of the restrictions on the CP-140 Aurora dataset. In particular, we highlight the stopping criteria selected. Fig. 4.3 shows the calculated loss and accuracy per iteration of two experimental runs, one CP-140 Aurora run and one Mukherjee run. On successful runs, the calculated loss in the Aurora dataset drops quickly and reaches a relatively stable plateau after 8 iterations. After the initial drop, further decreases are typically only seen immediately after the learning rate decreases. On the other hand, the Mukherjee dataset has a gradual decrease to its calculated loss, and it reaches its plateau at 77 iterations. The accuracy of the Mukherjee dataset reaches its plateau at 61 iterations after a slow increase from 20 iterations.

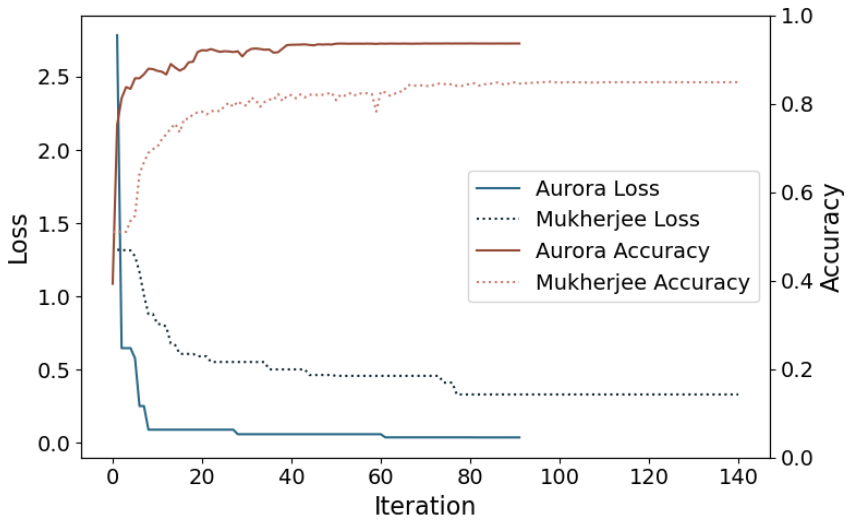

Fig. 4.3: Dataset Convergence

The differences in training time make this impact significant. A stopping criterion that is more aggressive on the CP-140 Aurora dataset which stops after 50 iterations reduces training time by 50% when the model typically reaches 100 iterations. On the [7:1] xLSTM configuration, this time savings can decrease training time by over 100 hours by cutting 50 iterations.

Fig. 4.4 displays model training time by iteration. There is a clear increase in the amount of time it takes to train models based on larger datasets which is not linear. The CP-140 Aurora dataset is approximately 20 times larger than the Mukherjee dataset, but in many cases trained 50 times slower. However, part of this difference we attribute to the presence of different hardware on the systems the models were trained on. This highlights previous recommendations that sufficient resources must be available to train this model on large datasets with a large xLSTM stack. Label prediction time depends on the number of sequences to label, this prediction time remains small in comparison to the massive training resource required to train the classification model.

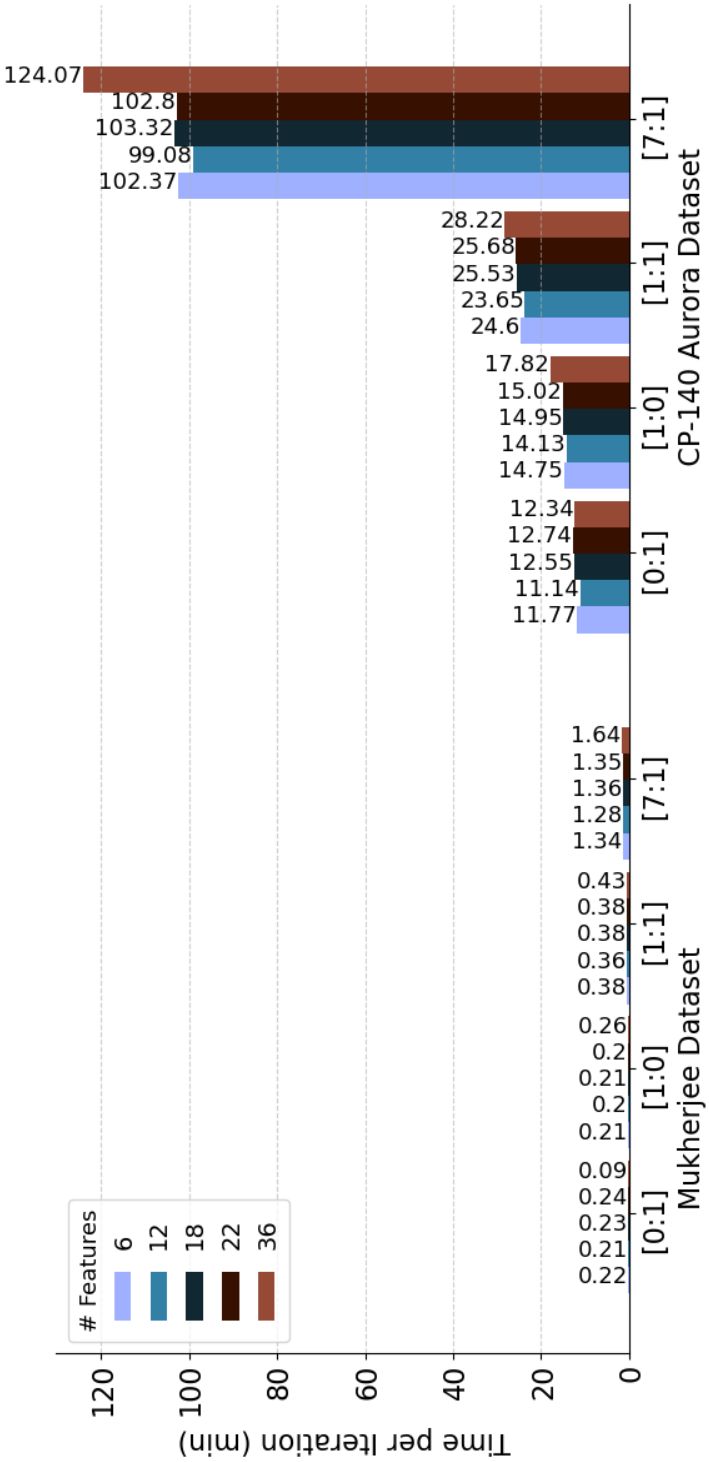

Fig. 4.4: Model Training Time by Iteration (mins)

The experimental results we obtain through this research demonstrate successful classification of aircraft flight regimes from datasets stemming solely from MIL-STD-1553B traffic. We observed weighted F_1 scores of 0.9019 processing Mukherjee's [17] dataset and 0.9468 on real avionics traffic with the CP-140 Aurora dataset [18]. Similar accuracy results are noted with values of 0.9032 and 0.9470 respectively. Our results compare well to leading techniques in aircraft flight regime classification using data stemming from the HUMS. Interestingly, we observed better performance on the real avionics data versus the synthetic traffic.

In summary, this chapter covered the experimental results that we observed and validated that we met our aim statement. While there are alternative techniques to classify aircraft flight regimes, this research is the first to do so using data solely extracted from MIL-STD-1553B data buses.

5 Conclusion

This chapter provides a summary of the work we conducted to validate this research's aim statement. We present an overview of work completed, highlight contributions to the field this research makes, propose areas for future work and provide recommendations that stem from this research. This body of work focuses on classification of aircraft flight regimes using data extracted solely from MIL-STD-1553B data bus traffic using an xLSTM deep learning pipeline. We use two datasets to validate the aim which we analyse using traditional statistical analysis techniques.

5.1 Overview

Message traffic transmitted over MIL-STD-1553B buses consist of complex avionics data that permit safe effective flight. Contemporary research demonstrates the MIL-STD-1553B bus is vulnerable to attacks [1, 3] for which it has few defences. Modern security research efforts on the MIL-STD-1553B bus [8, 11, 34] focus on intrusion detection models to detect attacks against the data bus. Under these premises, the malicious traffic must diverge from normal traffic enough that it gets flagged as anomalous. Our work permits the separation of MIL-STD-1553B traffic into narrower datasets by flight regime making it more difficult for malicious traffic to hide in plain sight. In addition, some research [10, 15] tangentially considers flight regimes through their two distinct flight regimes observing a difference in detection rates between the two datasets.

This research implements an xLSTM deep learning classification pipeline as described in Fig. 3.3, classifying both Mukherjee's synthetic dataset [17] and a real avionics dataset provided by DTAES [18] into distinct aircraft flight regimes. We evaluate performance using accuracy and weighted F_1 scores. We perform this analysis over various xLSTM stack configurations to compare and contrast both classification outcomes and training resource requirements.

This approach is likely limited to require training of models for each different set of aircraft hardware and software versions which could be numerous depending on the size of aircraft fleets and modification status. The flexibility of the configuration of the xLSTM stack permits quicker training with acceptable results using a [0:1] xLSTM stack or a longer training set with additional xLSTM components such as the [7:1] xLSTM stack which provides higher accuracy classifying aircraft flight regimes.

Our research is the first to perform aircraft flight regime classification using solely data extracted from the MIL-STD-1553B data bus. While regime classification is not new, contemporary techniques source their data from the aircraft's HUMS [24, 27, 28]. Our model performs better using real avionics traffic surpassing most contemporary research using real aircraft data and approaches the best classification technique performed by Leoni et al. [28]. Our approach also surpasses results of the research we assess as being closest in scope, Berry et al. [14].

5.2 Contributions

This research makes the following contributions to the field:

1. Provides a solution to classify aircraft flight regimes using solely messages extracted from MIL-STD-1553B data buses.
2. Demonstrated xLSTM's applicability to MIL-STD-1553B research.
3. Creates a pathway to produce MIL-STD-1553B datasets divided by aircraft flight regimes for the purposes of intrusion detection or other research.
4. Provides common link between MIL-STD-1553B security research and the broader aviation flight regime classification research.

5.3 Future Work

This research takes MIL-STD-1553B message traffic and classifies the messages into aircraft flight regimes. We suggest this research can be applied to the following areas:

1. MIL-STD-1553B Intrusion Detection research that divides the datasets into smaller closely related datasets by flight regime.
2. Further research focusing on model convergence speed which may further decrease model training time.
3. Examining model portability between different MIL-STD-1553B data buses including differing software versions of RTs, the presence of ad-

ditional or fewer RTs, different aircraft of the same type and different types of aircraft.

5.4 Recommendations

This research classifies MIL-STD-1553B message traffic into distinct flight regimes. For practical applications, a larger framework must be developed to produce labelled data to perform initial model training and to train a series of models applicable to a wider selection of military aircraft. Further research would be needed in order to determine the model's flexibility in classifying differences in network traffic due to alternate configurations of RTs. While the premise of this research hinged on security research, in the absence of other data, this work demonstrates the ability to classify aircraft flight regimes to a level similar to contemporary approaches.

Bibliography

[1] Daniel Lydiate, "Military Aviation's Cyber Challenge; Are Cyber-Vulnerabilities a Credible Threat to a Modern Airforce?," *Air and Space Power Review*, vol. 22, no. 1, pp. 6–38, 2019.

[2] United States. Department of Defense, "Aircraft Internal Time Division Command/Response Multiplex Data Bus," Sept. 1976.

[3] K. Lounis, Z. Mansour, M. Wrana, M. A. Elsayed, S. H. H. Ding, and M. Zulkernine, "A Review and Analysis of Attack Vectors on MIL-STD-1553 Communication Bus," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, pp. 5586–5606, Dec. 2022.

[4] Jeremy Paquet, *Uncovering MIL-STD-1553 vulnerabilities: Exploitability of Military Aircraft Networks*. M.A.Sc Thesis, Royal Military College of Canada, Apr. 2014. [Protected A].

[5] Canada. Department of National Defence, "CP-140 Aurora fleet modernization and life extension," June 2023. canada.ca. https://www.canada.ca/en/department-national-defence/services/procurement/cp-140-aurora.html. (accessed Feb. 7, 2025).

[6] Thuy Nguyen, "Towards MIL-STD-1553B Covert Channel Analysis," Technical Report NPS-CAG-15-001, Naval Postgraduate School, Monterey, California, Jan. 2015.

[7] Canada, Department of National Defence, Royal Canadian Air Force, "CH148822 Cyclone - Epilogue," tech. rep., May 2021.

[8] E. Levy, N. Maman, A. Shabtai, and Y. Elovici, "AnoMili: Spoofing Hardening and Explainable Anomaly Detection for the 1553 Military Avionic Bus," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 60, pp. 7738–7753, Dec. 2024.

[9] F. Onodueze and D. Josyula, "Anomaly Detection on MIL-STD-1553 Dataset using Machine Learning Algorithms," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, (Guangzhou, China), pp. 592–598, IEEE, Dec. 2020.

[10] M. A. Elsayed, M. Wrana, Z. Mansour, K. Lounis, S. H. H. Ding, and M. Zulkernine, "AdaptIDS: Adaptive Intrusion Detection for Mission-Critical Aerospace Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 23459–23473, Dec. 2022.

[11] D. Soucy and B. Lachine, "MIL-STD-1553B Intrusion Detection System," in *Proceedings of the 23rd European Conference on Cyber Warfare and Security*, vol. 23, (Jyväskylä, Finland), pp. 528–537, June 2024.

[12] S. J. J. Généreux, A. K. H. Lai, C. O. Fowles, V. R. Roberge, G. P. M. Vigeant, and J. R. Paquet, "MAIDENS: MIL-STD-1553 Anomaly-Based Intrusion Detection System Using Time-Based Histogram Comparison," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, pp. 276–284, Feb. 2020.

[13] O. Stan, A. Cohen, Y. Elovici, and A. Shabtai, "Intrusion Detection System for the MIL-STD-1553 Communication Bus," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, pp. 3010–3027, Aug. 2020.

[14] J. Berry, R. Vaughan, J. Keller, J. Jacobs, P. Grabill, and T. Johnson, "Automatic Regime Recognition Using Neural Networks," in *Proceedings of the American Helicopter Society 60th Annual Forum*, pp. 9–11, May 2006.

[15] M. M. Wrana, M. Elsayed, K. Lounis, Z. Mansour, S. Ding, and M. Zulkernine, "OD1NF1ST: True Skip Intrusion Detection and Avionics Network Cyber-attack Simulation," *ACM Transactions on Cyber-Physical Systems*, vol. 6, pp. 1–27, Oct. 2022.

[16] D. He, X. Liu, J. Zheng, S. Chan, S. Zhu, W. Min, and N. Guizani, "A Lightweight and Intelligent Intrusion Detection System for Integrated Electronic Systems," *IEEE Network*, vol. 34, pp. 173–179, July 2020.

[17] S. Mukherjee, "Extended the MIL-STD-1553B Cyber Trainer for Dataset Generation And Intrusion Detection Research." MEng Project, Royal Military College of Canada, Kingston, ON, 2026.

[18] Canada. Department of National Defence, Directorate of Technical Airworthiness and Engineering Support, "MIL-STD-1553 CP-140 Aurora Dataset," 2023. unpublished.

[19] United States. Department of Defense, "Digital Time Division Command/Response Multiplex Data Bus," 2018.

[20] European Space Agency, "Mil-STD-1553." esa.int. https://www.esa.int/Enabling/Space/Onboard/Mil-STD-1553. (accessed Jan. 26, 2026).

[21] Aaron Fansler, "Protection of the MIL-STD-1553 With Discrete Wavelet Algorithms," in *Proceedings of the International Telemetering Conference*, vol. 54, (Glendale, AZ), pp. 709–719, International Foundation for Telemetering, Nov. 2018.

[22] J. Leoni, F. Zinnari, E. Villa, M. Tanelli, and A. Baldi, "Flight regimes recognition in actual operating conditions: A functional data analysis approach," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105016, 2022.

[23] Canada. Transport Canada, "Aeroplane Flight Data Recorder (FDR) Specifications," Dec. 2009.

[24] J. Wu, C. Sun, C. Zhang, X. Chen, and R. Yan, "Deep clustering variational network for helicopter regime recognition in HUMS," *Aerospace Science and Technology*, vol. 124, p. 107553, 2022.

[25] J. Wu, C. Hu, C. Sun, X. Chen, and R. Yan, "Aircraft flight regime recognition with deep temporal segmentation neural network," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105840, Apr. 2023.

[26] K. M. Jensen, I. F. Santos, L. K. H. Clemmensen, S. Theodorsen, and H. J. P. Corstens, "Mass estimation of ground vehicles based on longitudinal dynamics using IMU and CAN-bus data," *Mechanical Systems and Signal Processing*, vol. 162, p. 107982, 2022.

[27] E. Villa, M. Tanelli, G. Cazzulani, F. Zinnari, G. Coral, A. Baldi, U. Mariani, and D. Mezzazanica, "Optimizing Automatic Flight Condition Recognition through a Multi-Strategy Machine-Learning Based Approach," (West Palm Beach, Florida), pp. 1–9, Vertical Flight Society, May 2023.

[28] J. Leoni, E. Villa, G. Cazzulani, A. Baldi, U. Mariani, and M. Tanelli, "Enhancing Flight Condition Recognition Performance Through Functional Similarity," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 61, pp. 4270–4283, Apr. 2025.

[29] D. Musso and J. Rogers, "Interacting Multiple Model Estimation for Helicopter Regime Recognition," *Journal of Aircraft*, vol. 57, pp. 1134–1147, Nov. 2020.

[30] M. Şenipek and U. Kalkan, "Learning-Based Clustering for Flight Condition Recognition," in *European Rotorcraft Forum*, Sept. 2019. Place: Warsaw, Poland.

[31] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] The AlphaStar Team, "AlphaStar: Mastering the real-time strategy game StarCraft II," Jan. 2019.

[33] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xLSTM: Extended Long Short-Term Memory," in *Advances in Neural Information Processing Systems* (A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds.), vol. 37, pp. 107547–107603, Curran Associates, Inc., 2024.

[34] A. Harlow, B. Lachine, and V. Roberge, "Anomaly Detection for the MIL-STD-1553 Multiplex Data Bus using an LSTM Autoencoder," in *Proceedings of the 19th International Conference on Cyber Warfare and Security*, (Johannesburg, South Africa), pp. 103–111, Mar. 2024.

[35] I. Uluocak and M. Bilgili, "Daily air temperature forecasting using LSTM-CNN and GRU-CNN models," *Acta Geophysica*, vol. 72, pp. 2107–2126, June 2024.

[36] M. Alharthi and A. Mahmood, "xLSTMTime: Long-Term Time Series Forecasting with xLSTM," *AI*, vol. 5, no. 3, pp. 1482–1495, 2024.

[37] L. Kang, X. Fu, J. Vazquez-Corral, E. Valveny, and D. Karatzas, "xLSTM-ECG: Multi-label ECG Classification via Feature Fusion with xLSTM." arXiv:2504.16101, 2025.

[38] E. Sükei, M. Oghbaie, U. Schmidt-Erfurth, G. Klambauer, and H. Bogunović, "Advancing OCT-Based Retinal Disease Classification with XLSTM: A Framework for Variable-Length Volume Processing," in *2025 IEEE 22nd International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5, Apr. 2025. ISSN: 1945-8452.

[39] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media Inc, 2nd edition ed., Sept. 2019.

[40] M. C. Hinojosa Lee, J. Braet, and J. Springael, "Performance Metrics for Multilabel Emotion Classification: Comparing Micro, Macro, and Weighted F1-Scores," *Applied Sciences*, vol. 14, p. 9863, Jan. 2024.

[41] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Wojciech Samek, and Tobias Schaeffter, "PTB-XL, a large publicly available electrocardiography dataset," Nov. 2022.

[42] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pp. 437–478. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

Appendices

A Dataset Features

This appendix provides a listing of all features (words) present on the MIL-STD-1553B data bus. In addition, the covariance ranking of each feature will be presented alongside its inclusion in different values of number of features that we use to train the xLSTM deep learning model.

Table A.1: Dataset Features

Feature Name	Mukherjee Covariance Ranking	Included in Features				
		6	12	18	22	36
Data 13	1	✓	✓	✓	✓	✓
Data 15	2	✓	✓	✓	✓	✓
Data 9	3	✓	✓	✓	✓	✓
Data 6	4	✓	✓	✓	✓	✓
Data 11	5	✓	✓	✓	✓	✓
Data 7	6	✓	✓	✓	✓	✓
Data 5	7		✓	✓	✓	✓
Data 4	8		✓	✓	✓	✓
Data 22	9		✓	✓	✓	✓
Data 26	10		✓	✓	✓	✓
Data 17	11		✓	✓	✓	✓
Data 14	12		✓	✓	✓	✓
Data 3	13			✓	✓	✓
Data 10	14			✓	✓	✓
Data 20	15			✓	✓	✓
Data 8	16			✓	✓	✓
Command 2	17			✓	✓	✓
Data 21	18			✓	✓	✓
Data 24	19				✓	✓
Data 18	20				✓	✓
Data 27	21				✓	✓
Data 25	22				✓	✓
Data 12	23					✓
Status 2	24					✓
Data 16	25					✓
Data 19	26					✓
Data 1	27					✓
Data 23	28					✓
Data 31	29					✓
Data 28	30					✓
Data 29	31					✓
Data 32	32					✓
Status 1	33					✓
Command 1	34					✓
Data 30	35					✓
Data 2	36					✓
Time Stamp	Removed prior to training					
Label	Removed prior to training					

B Additional Graphical Results

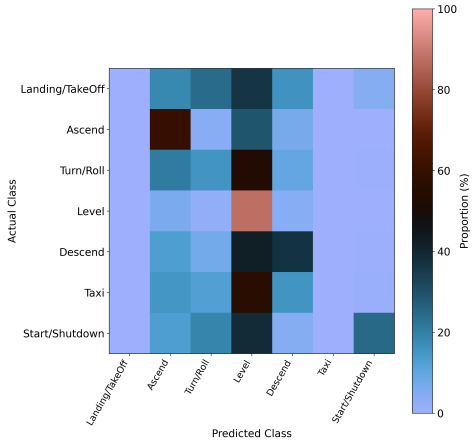

Fig. B.1: Mukherjee [0:1] xLSTM Stack with 6 Features on Mukherjee Dataset

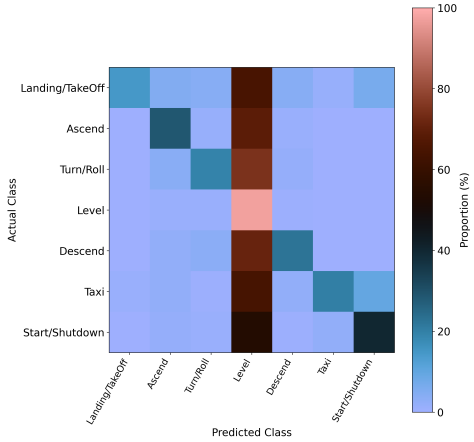

Fig. B.2: Mukherjee [0:1] xLSTM Stack with 12 Features on Mukherjee Dataset

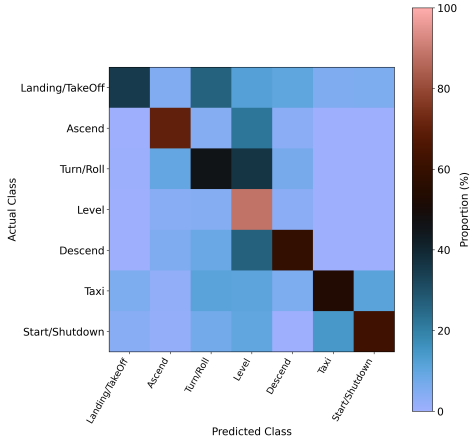

Fig. B.3: Mukherjee[0:1] xLSTM Stack with 18 Features on Mukherjee Dataset

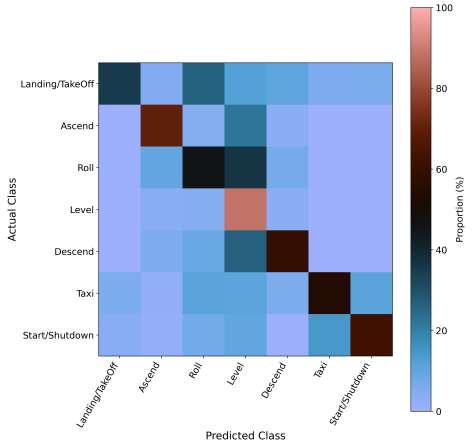

Fig. B.4: Mukherjee[0:1] xLSTM Stack with 22 Features on Mukherjee Dataset

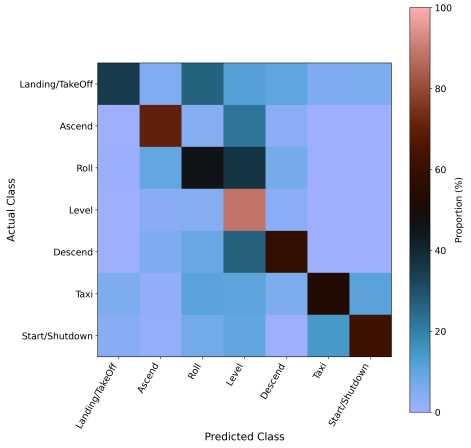

Fig. B.5: [0:1] xLSTM Stack with 36 Features on Mukherjee Dataset

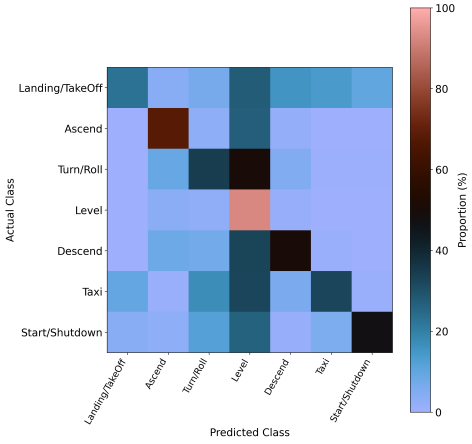

Fig. B.6: [1:0] xLSTM Stack with 6 Features on Mukherjee Dataset

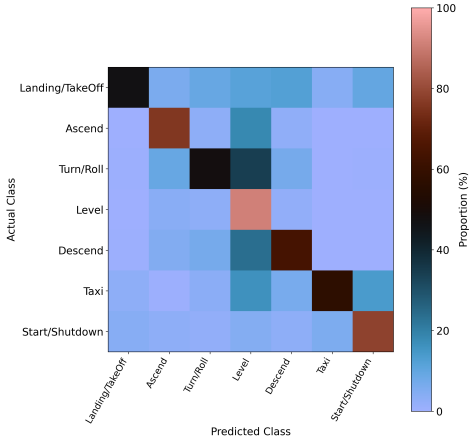

Fig. B.7: [1:0] xLSTM Stack with 12 Features on Mukherjee Dataset

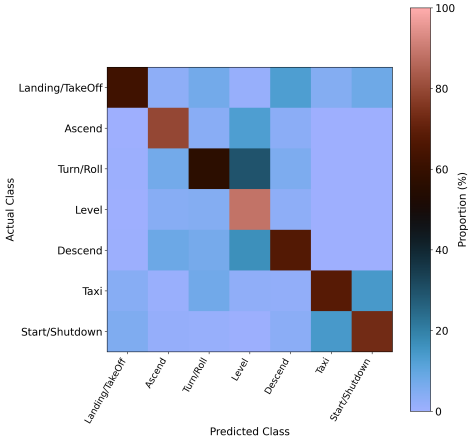

Fig. B.8: [1:0] xLSTM Stack with 18 Features on Mukherjee Dataset

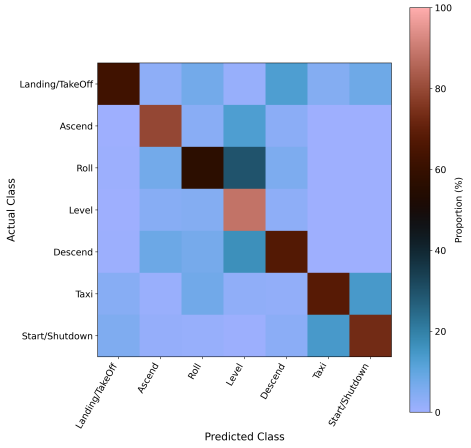

Fig. B.9: [1:0] xLSTM Stack with 22 Features on Mukherjee Dataset

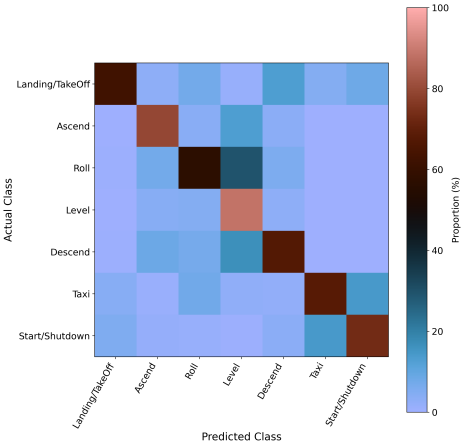

Fig. B.10: [1:0] xLSTM Stack with 36 Features on Mukherjee Dataset

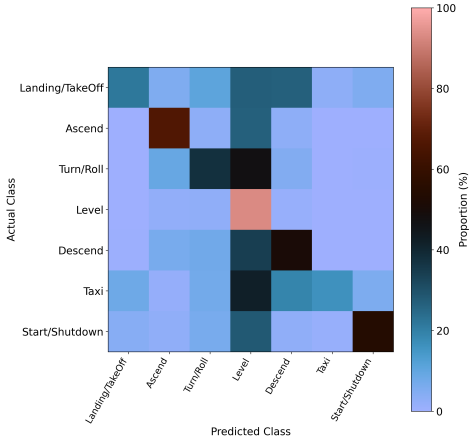

Fig. B.11: [1:1] xLSTM Stack with 6 Features on Mukherjee Dataset

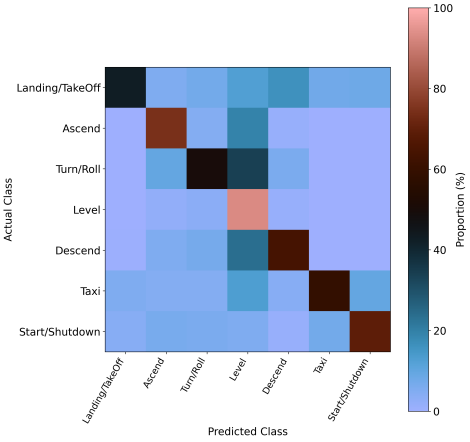

Fig. B.12: [1:1] xLSTM Stack with 12 Features on Mukherjee Dataset

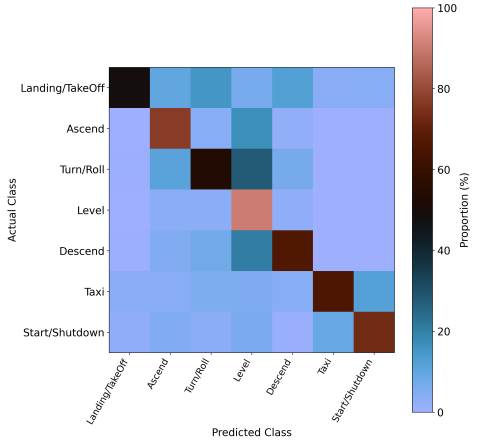

Fig. B.13: [1:1] xLSTM Stack with 18 Features on Mukherjee Dataset

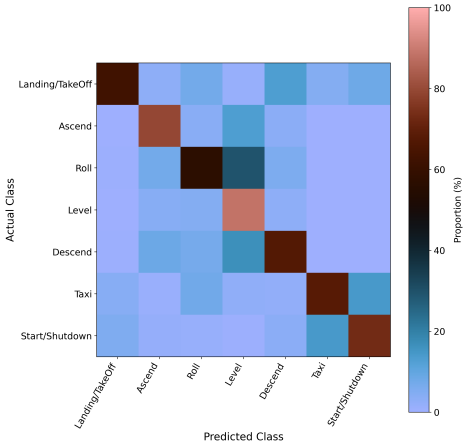

Fig. B.14: [1:1] xLSTM Stack with 22 Features on Mukherjee Dataset

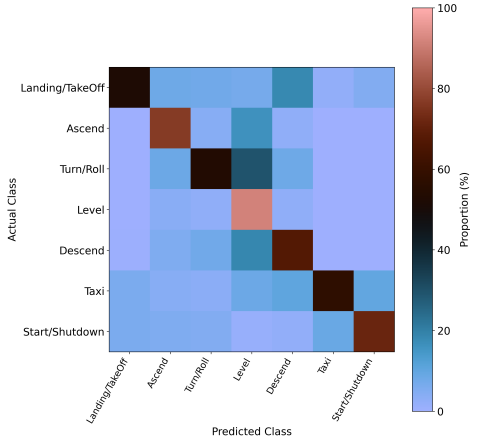

Fig. B.15: [1:1] xLSTM Stack with 36 Features on Mukherjee Dataset

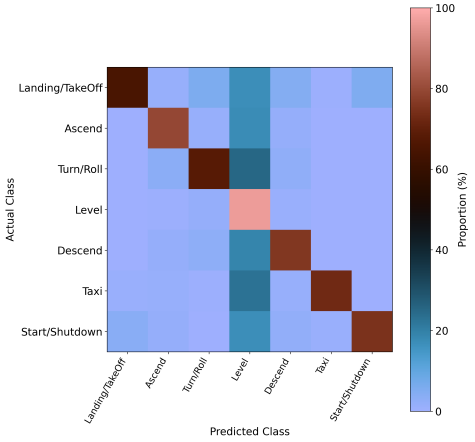

Fig. B.16: [7:1] xLSTM Stack with 6 Features on Mukherjee Dataset

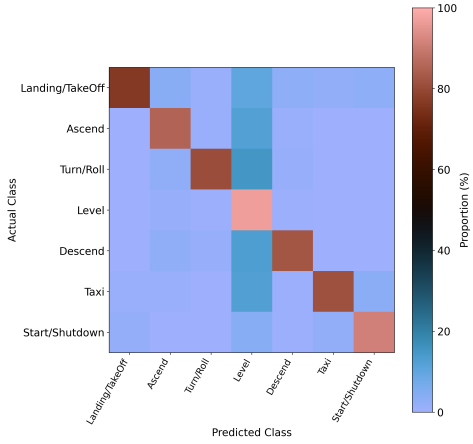

Fig. B.17: [7:1] xLSTM Stack with 12 Features on Mukherjee Dataset

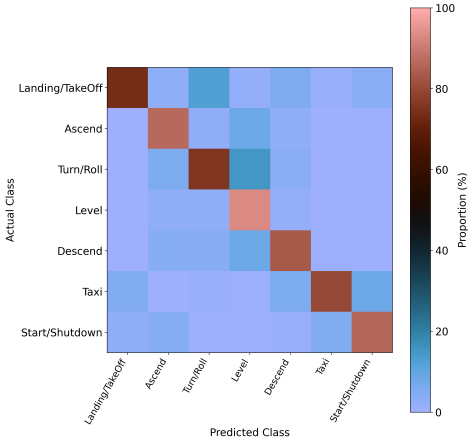

Fig. B.18: [7:1] xLSTM Stack with 18 Features on Mukherjee Dataset

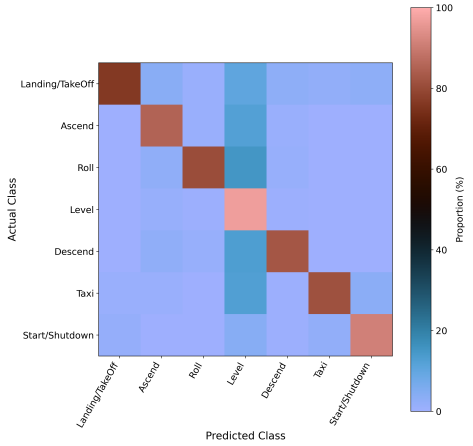

Fig. B.19: [7:1] xLSTM Stack with 22 Features on Mukherjee Dataset

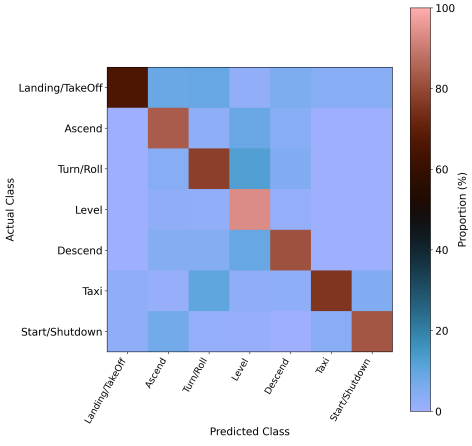

Fig. B.20: [7:1] xLSTM Stack with 36 Features on Mukherjee Dataset

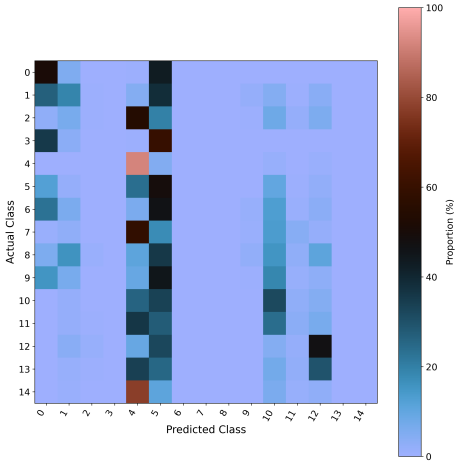

Fig. B.21: [0:1] xLSTM Stack with 6 Features on CP-140 Aurora Dataset

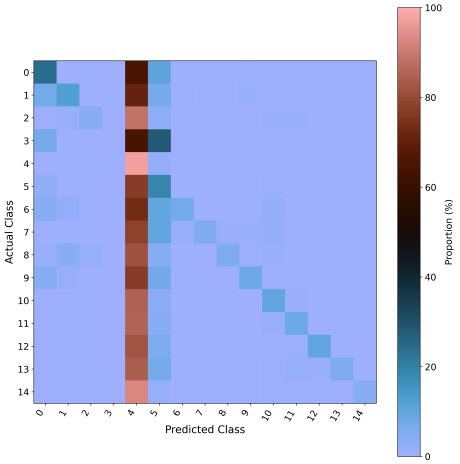

Fig. B.22: [0:1] xLSTM Stack with 12 Features on CP-140 Aurora Dataset

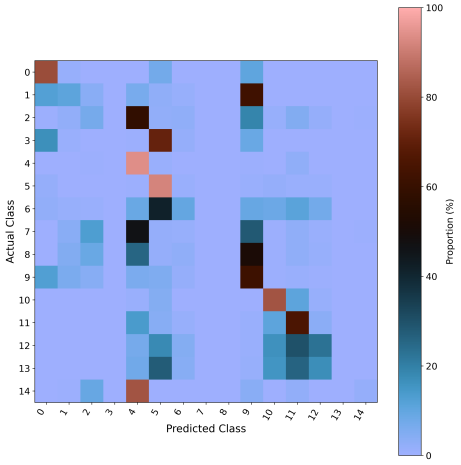

Fig. B.23: [0:1] xLSTM Stack with 18 Features on CP-140 Aurora Dataset

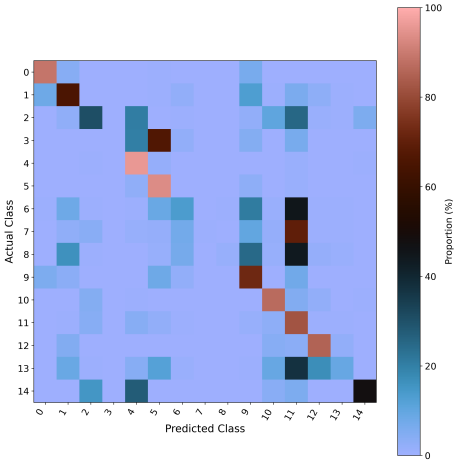

Fig. B.24: [0:1] xLSTM Stack with 22 Features on CP-140 Aurora Dataset

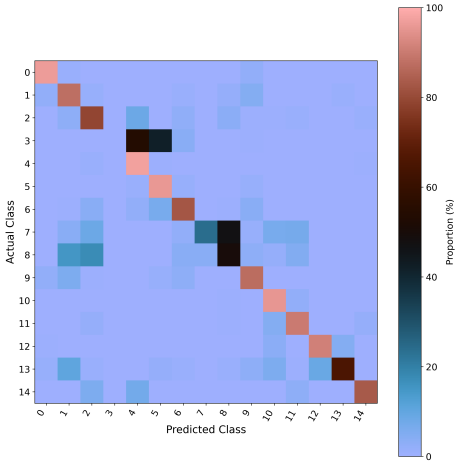

Fig. B.25: [0:1] xLSTM Stack with 36 Features on CP-140 Aurora Dataset

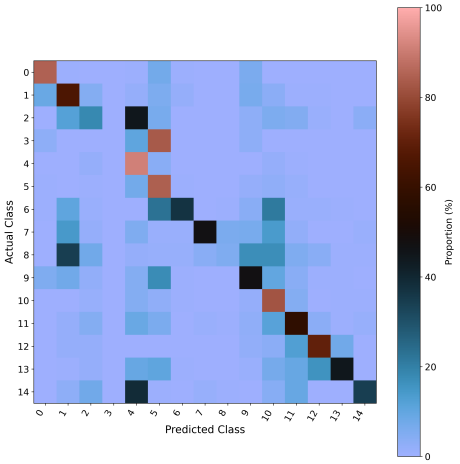

Fig. B.26: [1:0] xLSTM Stack with 6 Features on CP-140 Aurora Dataset

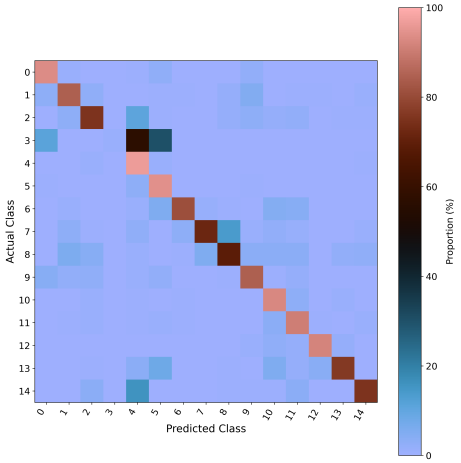

Fig. B.27: [1:0] xLSTM Stack with 12 Features on CP-140 Aurora Dataset

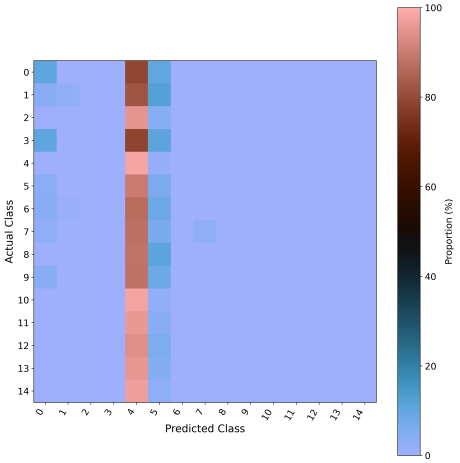

Fig. B.28: [1:0] xLSTM Stack with 18 Features on CP-140 Aurora Dataset

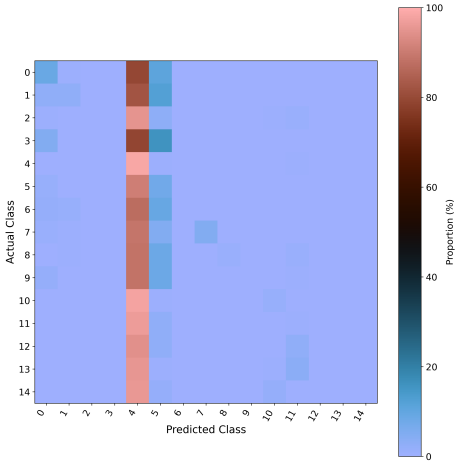

Fig. B.29: [1:0] xLSTM Stack with 22 Features on CP-140 Aurora Dataset

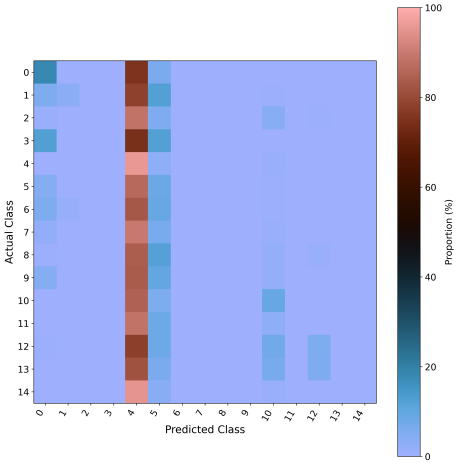

Fig. B.30: [1:0] xLSTM Stack with 36 Features on CP-140 Aurora Dataset

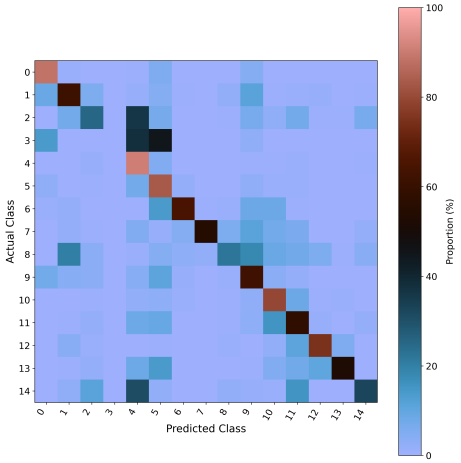

Fig. B.31: [1:1] xLSTM Stack with 6 Features on CP-140 Aurora Dataset

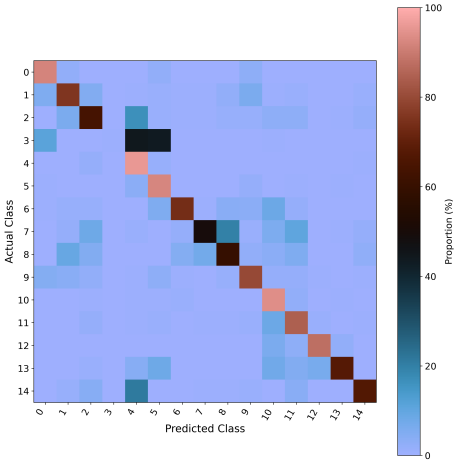

Fig. B.32: [1:1] xLSTM Stack with 12 Features on CP-140 Aurora Dataset

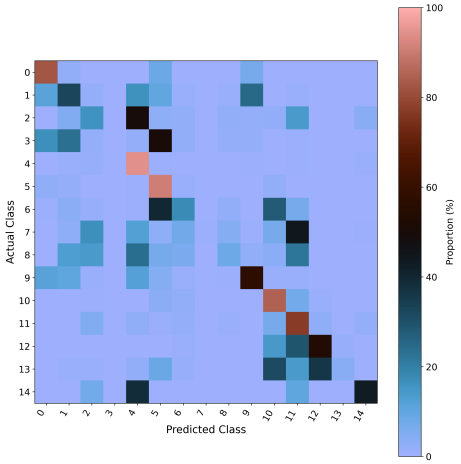

Fig. B.33: [1:1] xLSTM Stack with 18 Features on CP-140 Aurora Dataset

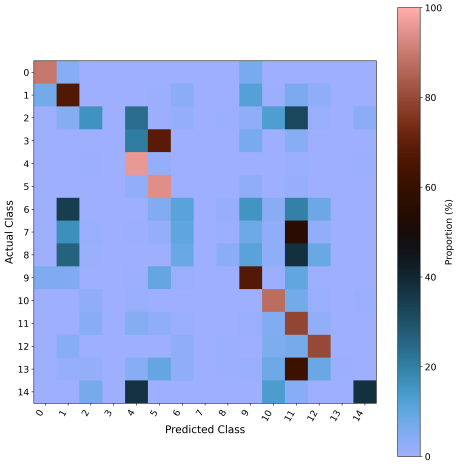

Fig. B.34: [1:1] xLSTM Stack with 22 Features on CP-140 Aurora Dataset

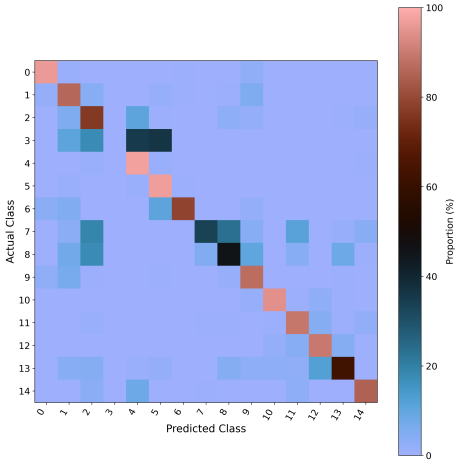

Fig. B.35: [1:1] xLSTM Stack with 36 Features on CP-140 Aurora Dataset

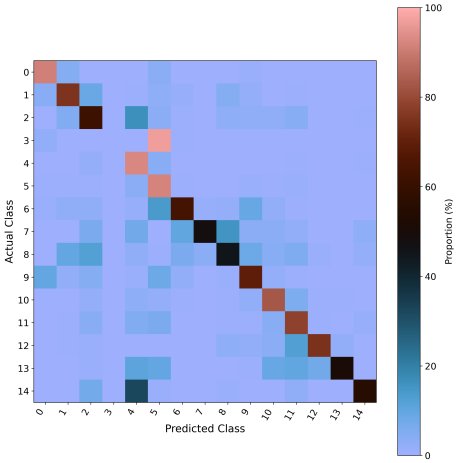

Fig. B.36: [7:1] xLSTM Stack with 6 Features on CP-140 Aurora Dataset

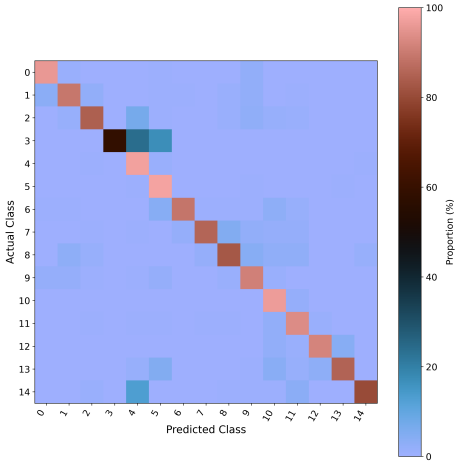

Fig. B.37: [7:1] xLSTM Stack with 12 Features on CP-140 Aurora Dataset

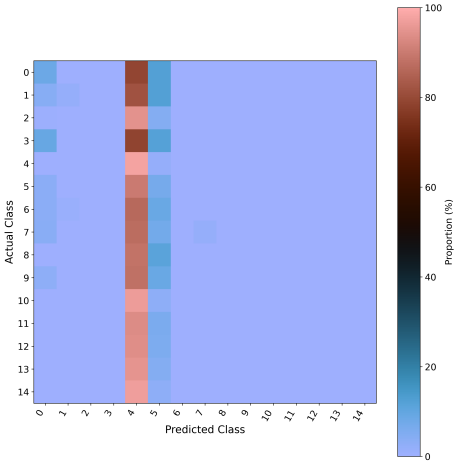

Fig. B.38: [7:1] xLSTM Stack with 18 Features on CP-140 Aurora Dataset

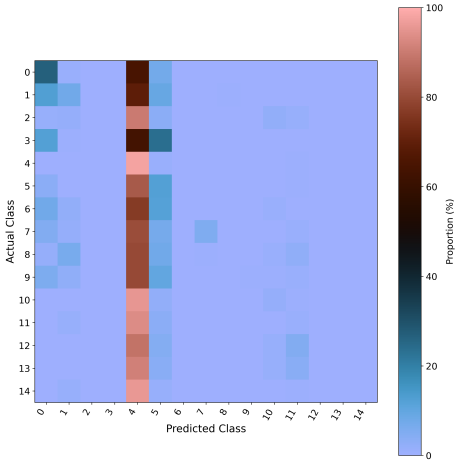

Fig. B.39: [7:1] xLSTM Stack with 22 Features on CP-140 Aurora Dataset

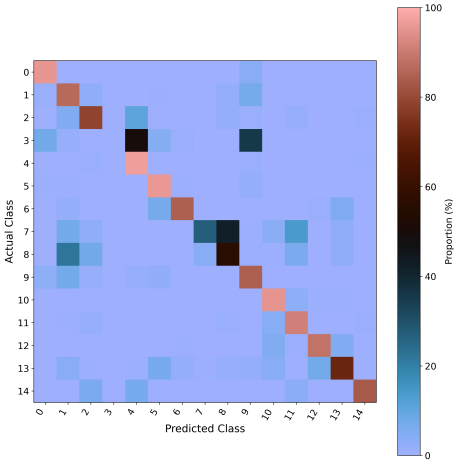

Fig. B.40: [7:1] xLSTM Stack with 36 Features on CP-140 Aurora Dataset

C Listing of xLSTM Hyperparameters

This appendix details the hyperparameters intrinsic to the xLSTM model published by Beck et al. [33] found within their Python GitHub repository. In addition, we list the default values of each hyperparameter that they define.

Table C.1: xLSTM Hyperparameters

Hyperparameter	Default Value
Activation Function	Gaussian Error Linear Unit
Batch Size	8
Embedding Dimension	None
Feedforward Projection Factor	1.3
Location of sLSTM(s) blocks	1 (0 indexed)
Number of Heads	4
Number of mLSTM Blocks	1
Number of sLSTM Blocks	1