FEATURE ENGINEERING FOR A MIL-STD-1553B LSTM AUTOENCODER ANOMALY DETECTOR

EXTRACTION DE CARACTÉRISTIQUES POUR UN DÉTECTEUR D'ANOMALIES PAR AUTO-ENCODEUR RÉCURRENT À MÉMOIRE COURT-TERME ET LONG TERME POUR MIL-STD-1553B

A Thesis Submitted to the Royal Military College of Canada
by

Dakotah Soucy, BSc, RMC
Captain

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

May, 2022

Acknowledgements

I would like to thank my supervisor, Brian Lachine, for the constant support and advice during this entire process.

Abstract

The MIL-STD-1553B data bus protocol is used to enable communications between aircraft subsystems. These interconnected subsystems are responsible for core services such as communications, instrument data and aircraft control. With fleet modernization, more threat vectors are introduced through increased inter-connectivity. These additional threat vectors create an opportunity in which adversaries have the ability to exploit vulnerabilities in the MIL-STD-1553B protocol. This potential for exploitation introduces a requirement for an Intrusion Detection System in order to maintain the reliability of the MIL-STD-1553B protocol and safety of the aircraft. Current research into MIL-STD-1553B Intrusion Detection Systems utilize signature or anomaly-based approaches. These methods demonstrated detection of malicious traffic; however, they require further improvements in order to improve their effectiveness.

The aim of this research is to refine the feature engineering component of an existing MIL-STD-1553B deep learning anomaly detector in order to improve its overall effectiveness. In this work, feature engineering includes generation of new features, feature selection and dimensionality reduction. The generation of new features creates an extended dataset derived from the primary features. Using three different supervised feature selection and one feature reduction technique, different feature sets are created for training and testing with an existing Long-Short Term Memory autoencoder anomaly detector.

In order to accomplish this aim, sixteen models are created. Twelve of these models are attack specific, created from four distinct attack types and three feature selection techniques against the original and generated feature set. The remaining four are general models. Three are based on features identified across all four attack types using the three selection techniques and the full feature set. The fourth general model is based on the dimensionality reduction technique that processed only the original feature set and did not consider attack type. These sixteen models are then evaluated using common performance metrics and compared to those of the original anomaly detector. This research is validated by the marked performance improvement achieved by the feature engineering refinements made in comparison to those of the original model. In addition, this research also showed a significant reduction in the number of features required to achieve this performance gain.

Résumé

Le protocole de bus de données MIL-STD-1553B est utilisé pour permettre les communications entre les sous-systèmes de l'avion. Ces sous-systèmes interconnectés sont responsables des services de base tels que les communications, les données des instruments et le contrôle de l'aéronef. Avec la modernisation des avions, de nouveaux vecteurs d'attaques sont introduits grâce à une interconnectivité accrue. Ces vecteurs d'attaque créent de nouvelles opportunités dans lesquelles les adversaires ont la capacité d'exploiter les vulnérabilités du protocole MIL-STD-1553B. Cette nouvelle menace amène une exigence pour un système de détection d'intrusion afin de maintenir la fiabilité du protocole MIL-STD-1553B et la sécurité de l'avion. Les recherches actuelles sur les systèmes de détection d'intrusion MIL-STD-1553B utilisent des approches basées sur les signatures ou les anomalies. Ces méthodes ont démontré la capacité de détecter du trafic malveillant; cependant, ils nécessitent d'autres affermissements afin d'améliorer leur efficacité.

L'objectif de cette recherche est d'améliorer l'extraction des caractéristiques d'un détecteur d'anomalies par apprentissage profond MIL-STD-1553B existant afin d'améliorer son efficacité globale. Dans ce travail, l'extraction de caractéristiques comprend la génération de nouvelles caractéristiques, la sélection des caractéristiques et la réduction de la dimensionnalité. La génération de nouvelles caractéristiques crée un jeu de données étendu dérivé des caractéristiques principales. À l'aide de trois sélections de caractéristiques supervisées différentes et d'une technique de réduction de caractéristiques, différents ensembles de caractéristiques sont créés pour la formation et les tests avec un détecteur d'anomalie par auto-encodeur récurrent à mémoire à long terme existant.

Pour atteindre cet objectif, seize modèles sont créés. Douze de ces modèles sont spécifiques à l'attaque, créés à partir de quatre types d'attaque distincts et trois techniques de sélection de caractéristiques par rapport à l'ensemble des caractéristiques original et généré. Les quatre autres sont des modèles généraux. Trois sont basés sur des caractéristiques identifiées dans les quatre types d'attaques à l'aide des trois techniques de sélection et de l'ensemble complet des caractéristiques. Le quatrième modèle général est basé sur la technique de réduction de la dimensionnalité qui ne traite que l'ensemble de caractéristiques d'origine et ne prend pas en compte le type d'attaque. Ces seize modèles sont ensuite évalués à l'aide de paramètres de performance communs et comparés à ceux du détecteur d'anomalies d'origine. Cette recherche est validée par la nette amélioration des performances obtenue grâce aux améliorations apportées à l'extraction des caractéristiques par rapport à celles du modèle d'origine. De plus, cette recherche a également montré une réduction significative du nombre de caractéristiques nécessaires pour atteindre ce gain de performance.

Contents

List of Tables

List of Figures

List of Acronyms

ANOVA	Analysis of Variance
AUROCC	Area Under Receiver Operating Characteristic Curve
BC	Bus Controller
Bi-LSTM	Bidirectional LSTM
BM	Bus Monitor
BMDX	Bus Monitor Data Files Extended
CAN	Controller Area Network
DoS	Denial of Service
FN	False Negative
FOS	Fast Orthogonal Search
FP	False Positive
FPR	False Positive Rate
IDS	Intrusion Detection System
LASSO	Least Absolute Shrinkage and Selection Operator
LOF	Local Outlier Factor
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MCC	Matthews Correlation Coefficient
MCD	Minimum Co-variance Determination
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
OCSVM	One Class Support Vector Machine
PPS	Predictive Power Score
RNN	Recurrent Neural Network
ROCC	Receiver Operating Characteristic Curve
RT	Remote Terminal
SVC	Support Vector Classification
TN	True Negative

TP	True Positive
TPR	True Positive Rate
tsfresh	Time Series FeatuRe Extraction on basis of Scalable Hypothesis
UMAP	Uniform Manifold Approximation and Projection

1 Introduction

The MIL-STD-1553B data bus protocol is used to enable communications between subsystems in civilian and military aircraft. These interconnected subsystems are responsible for core services such as communications, instrument data and aircraft control. MIL-STD-1553B was introduced in 1978 and was designed for reliability and safety in an air-gapped environment. With fleet modernization efforts, the increased connectivity to the systems external to the aircraft has created additional threat vectors to the MIL-STD-1553B data bus.

These additional threat vectors to the MIL-STD-1553B data bus create an opportunity in which adversaries have the ability to exploit vulnerabilities in the MIL-STD-1553B protocol. This potential for exploitation of the MIL-STD-1553B protocol introduces a requirement for Intrusion Detection System (IDS) in order to maintain the reliability of the MIL-STD-1553B protocol and safety of the aircraft. Both signature and anomaly-based IDS have recently been researched and provide viable options for monitoring vulnerabilities in the MIL-STD-1553B data bus introduced by these new threat vectors. This paper outlines the current vulnerabilities in the MIL-STD-1553B data bus protocol and discusses current detection techniques. Furthermore, it highlights the performance of an existing anomaly detector based on a Long-Short Term Memory (LSTM) model and introduces feature engineering refinements to improve the performance metrics of the LSTM based model. Performance metrics in this context refer to those derived from the confusion matrix.

1.1 Motivation

With fleet modernization, threat vectors are being introduced through connections such as data links and maintenance diagnostic tools. The exploitation of MIL-STD-1553B by an adversary with a presence on the network can result in three potential impacts: Denial of Service (DoS), data leakage and data in-

tegrity violations if adversaries accessed the data bus through the previously mentioned threat vectors [1], [2]. The time and financial resources needed to build additional security into the protocol itself is significant and given this gap in protection, the ability to monitor bus traffic and develop cyber-attack detection techniques is a very real requirement.

Recent research efforts in this area include machine learning, deep learning, statistical and signature-based approaches [2], [3], [4], [5]. A LSTM based deep learning model by Harlow[3] has demonstrated an initial proof of concept approach to effectively detect MIL-STD-1553B cyber attacks. The success of this initial work is promising and has highlighted areas where additional focus may add further improvement in detection capabilities.

1.2 Statement of Deficiency

Current anomaly detection techniques for the MIL-STD-1553B data bus protocol demonstrate effectiveness, though there is room for improvement and refinement to their respective pipelines that could improve their overall performance. The anomaly detection method by Harlow[3] demonstrated the ability and effectiveness of implementing a deep learning model for a MIL-STD-1553B IDS. Though, as highlighted by Harlow's research, there are areas for improvement. Specifically, feature engineering is one of these areas which has the ability to improve anomaly detection capabilities.

1.3 Aim

The aim of this research is to refine the feature engineering component of an existing MIL-STD-1553B deep learning anomaly detector in order to improve its overall effectiveness. The existing anomaly detector is the LSTM autoencoder developed by Harlow[3]. The feature engineering component includes generation of new features, feature selection and dimensionality reduction. The measure of effectiveness is based on the following performance metrics; precision, recall, Area Under Receiver Operating Characteristic Curve (AUROCC) and Matthews Correlation Coefficient (MCC). Validation is accomplished by comparing results from this research with the results attained by related research [3].

1.4 Research Activities

To improve the effectiveness of this existing model, modifications are made to the feature engineering component of this pipeline. In order to accomplish this, the following research activities are conducted:

1. MIL-STD-1553B traffic from Harlow [3] is put through the data munging process to form the primary features for the deep learning pipeline;
2. Feature Engineering:
   a) Using this primary feature set as a starting point, new features are generated through polynomial expansion, arithmetic combinations and utilization of feature generation tools such as Time Series FeatuRe Extraction on basis of Scalable Hypothesis (tsfresh) used by Stan et al. [6].
   b) Feature selection is used to create different feature sets to be input into the deep learning model. The feature selection algorithms evaluated are Fast Orthogonal Search (FOS), Predictive Power Score (PPS) and Analysis of Variance (ANOVA);
   c) A single dimensionality reduction algorithm is used to reduce the number of dimensions of the primary feature set. The dimensionality reduction algorithm evaluated is Uniform Manifold Approximation and Projection (UMAP) [7].
3. Training the LSTM model developed by Harlow [3] for anomaly detection using selected sets.
4. Evaluation of the different models created using the new features sets is conducted by comparing the performance metrics between the respective models.

1.5 Organization

The remaining chapters in this document provide further detail regarding the research conducted. Chapter 2 describes the MIL-STD-1553B protocol and provides security specific context regarding currently known attack and detection techniques. Additionally, Chapter 2 also includes background on feature engineering, deep learning models, specifically LSTM, as well as model tuning and evaluation metrics. Chapter 3 provides an overview of the refined feature engineering component of the pipeline and the different models that are developed. Chapter 4 includes the experiment results and discussion. Finally, Chapter 5 discusses the research contribution, suggested future work and recommendations.

2 Background

This chapter will provide background on the MIL-STD-1553B data bus protocol, its vulnerabilities and a review of current anomaly detection techniques. It will also review feature engineering techniques used in this research. Additionally, this chapter will lay out relevant information pertaining to LSTM models.

2.1 MIL-STD-1553B

The MIL-STD-1553B standard was published in 1978 by the United States Department of Defense which defines the mechanical, electrical and functional characteristics of a serial data bus. This standard defines a multi-point, serial communications bus allowing communications between terminals controlled through a command and response protocol. The typical architecture for MIL-STD-1553B is shown in Fig. 2.1 which consists of terminals connected through a dual redundant communications bus. The standard defines three distinct terminals:

1. Bus Controller (BC): The terminal responsible for initiating and directing information transfer on the bus.
2. Bus Monitor (BM): The terminal responsible for receiving and storing select bus traffic for use at a later time.
3. Remote Terminal (RT): Any terminal not operating as a BC or BM.

Additionally, there is a maximum of 32 addresses on the MIL-STD-1553B bus. Address 31 is reserved for broadcast transactions and the remaining 0 to 30 addresses are assignable to RTs [8].

2.1.1 Data Link Layer

Data transfer on the bus is accomplished through messages which are comprised of 20-bit components called words. MIL-STD-1553B defines three word

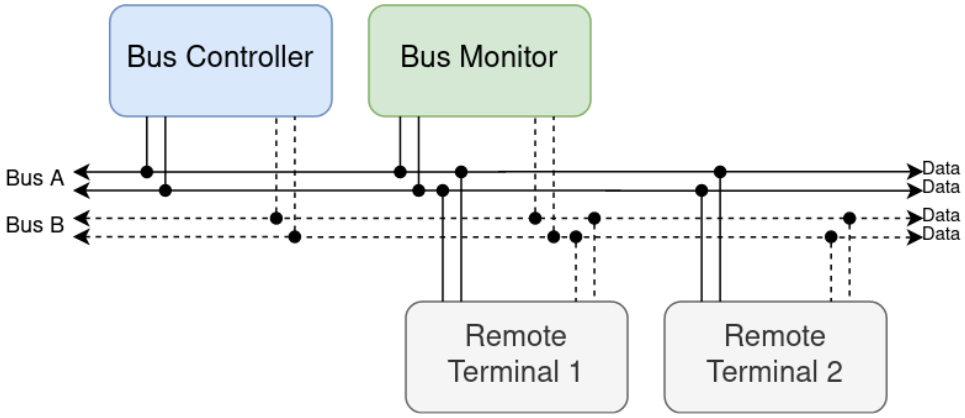

Figure 2.1: Example of MIL-STD-1553B bus topology with a BC, BM, and two RTs connected by a Dual Redundant Bus

types: command, data and status. Fig. 2.2 depicts the structure for each type of word. These words combine to form the larger messages and the standard defines two types of message formats: data messages and control messages.

Data messages are initiated by the BC issuing command words on the bus followed by the RTs transmitting and/or receiving the data on the bus. All RTs which are connected to the data bus have the ability to read all messages transmitted, but only the addressed RTs are expected to carry out the command sent by the BC. These data messages are further separated into communications between specific RTs and broadcast communications. There are three data message types between specific RTs: BC to RT, RT to BC and RT to RT. Additionally, there are two broadcast data message types: BC to RTs and RT to RTs. These messages and their required word sequences are outlined in Fig. 2.3.

Control messages enable the BC to monitor and control the bus by issuing mode commands to the RTs. Control messages are a set of predefined functions and can contain command and data words, or solely mode codes from the BC as seen in Table 2.1. When directed to a specific RT, the response can contain status and data words, or solely a status word depending on the initial mode command word. When the control message is a broadcast message, there are no responses from the RT(s). Fig. 2.4 illustrates the required words for each type of control message.

The BC directs all communications on the bus between RTs following a predefined schedule. This schedule contains two different types of message schedules: periodic and aperiodic. Periodic messages are transmitted at fixed

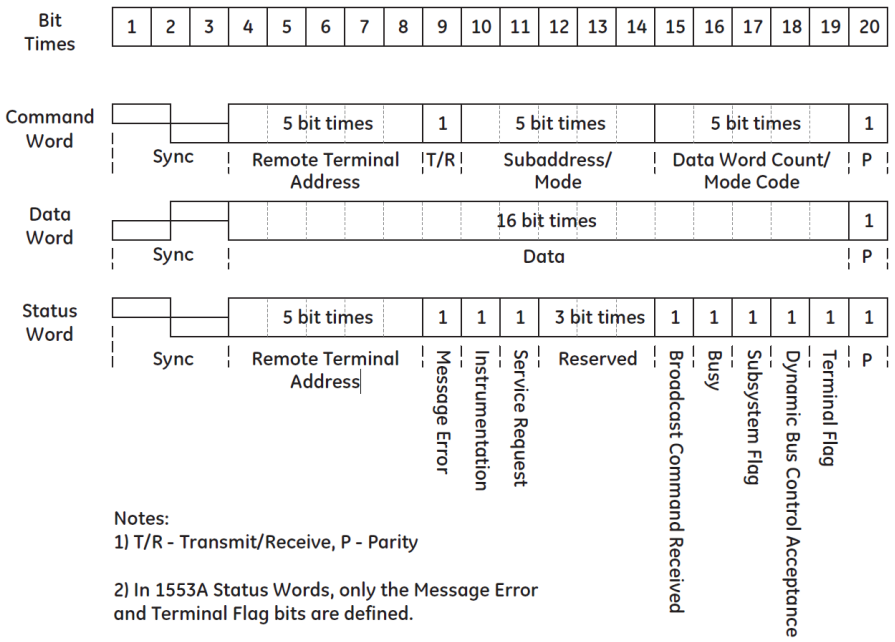

Figure 2.2: MIL-STD-1553B Word Formats [9]

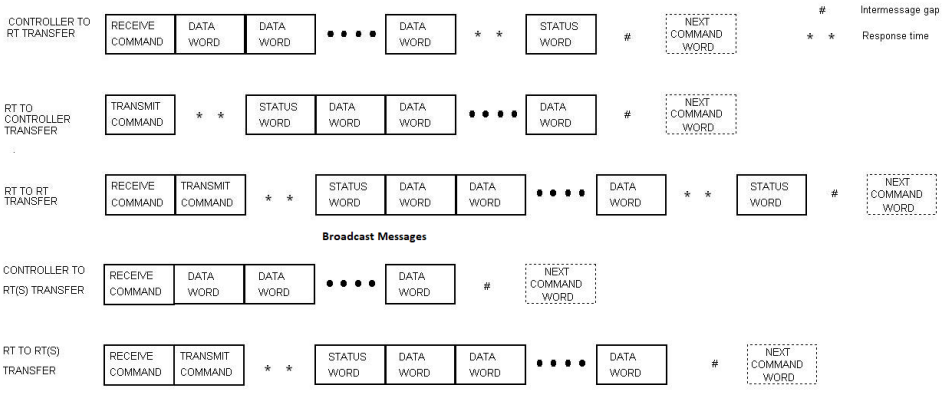

Figure 2.3: MIL-STD-1553B Data Message Formats [8]

T/R bit	Mode code	Function	Data word	Broadcast command
1	00000	Dynamic bus control	No	No
1	00001	Synchronize	No	Yes
1	00010	Transmit status word	No	No
1	00011	Initiate self-test	No	Yes
1	00100	Transmitter shutdown	No	Yes
1	00101	Override transmitter shutdown	No	Yes
1	00110	Inhibit terminal flag bit	No	Yes
1	00111	Override inhibit terminal flag bit	No	Yes
1	01000	Reset remote terminal	No	Yes
1	01001 01111	Reserved	No	TBD
1	10000	Transmit vector word	Yes	No
0	10001	Synchronize	Yes	Yes
1	10010	Transmit last command	Yes	No
1	10011	Transmit bit word	Yes	No
0	10100	Selected transmitter shutdown	Yes	Yes
0	10101	Override selected transmitter shutdown	Yes	Yes
1 or 0	10110 11111	Reserved	Yes	TBD

Table 2.1: MIL-STD-1553B Mode Codes

time intervals according to the schedule. Aperoidic messages are conditional and therefore not sent at a fixed interval, although they still retain a fixed time slot in the schedule if they are to be transmitted. Collections of perodic and aperiodic messages are combined to form a minor frame within the schedule. Furthermore, a collection of minor frames form the main schedule, also known as the major frame as seen in Fig. 2.5.

2.1.2 Security and Vulnerabilities

The MIL-STD-1553B data bus protocol was designed for reliability and safety of the system. As stated by the standard [8], all bus communications follow a predetermined cyclical schedule controlled by the BC. All RTs manufactured are also to follow the standards defined by MIL-STD-1553B. However, adver-

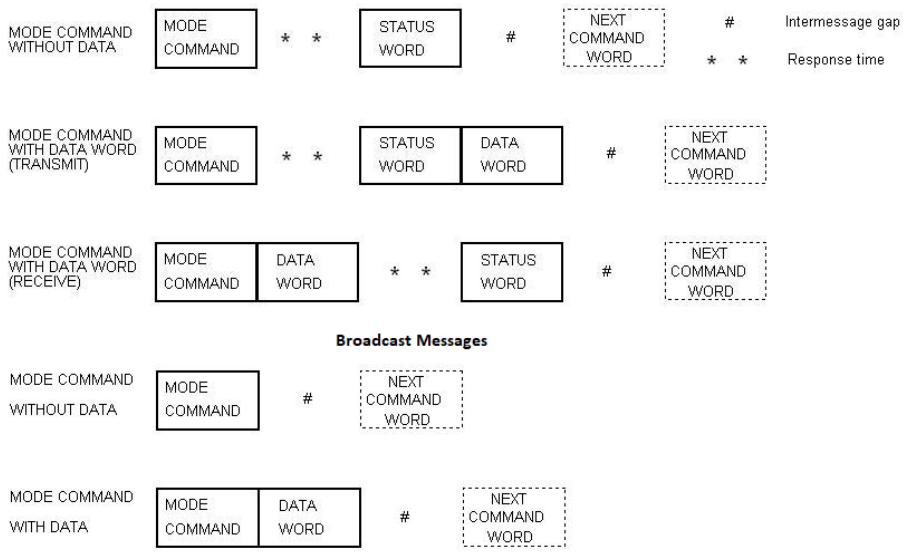

Figure 2.4: MIL-STD-1553B Control Message Formats [8]

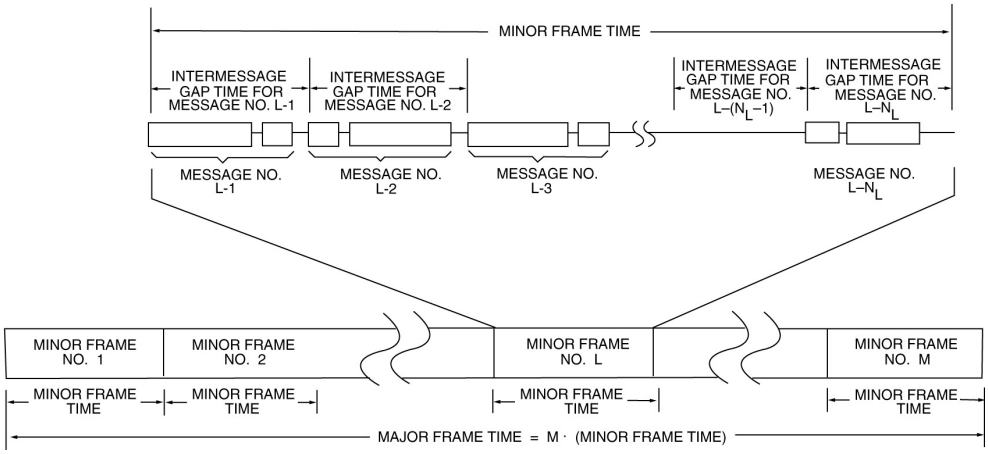

Figure 2.5: MIL-STD-1553B Major and Minor Frames [8]

saries are not constrained by these standards and can manipulate the protocol to their advantage to achieve their desired outcome.

Stan et al. [2] specify two main attack methods; message manipulation and behaviour manipulation.

1. Message manipulation: Modification of legitimate words that are transmitted over the data bus.
2. Behavior manipulation: Altering the normal behaviour of a compromised component such as transmitting fake messages in an unusual timing or order.

Stan et al.[2] then outline three attack outcomes that can be accomplished via the two attack methods. These attack consequences are DoS, data leakage, and data integrity violation. These attack consequences can be caused by either of the two attack methods discussed above. Furthermore, the attack methods can be physically carried out by two different categories of RTs; a rogue RT or a compromised RT. A rogue RT is a component that was not part of the original configuration of the system and is not authorized to be connected to the system. A compromised RT is a component the attacker can manipulate, but is part of the original system.

A DoS blocks and/or disrupts communication by targeting specific RT communications or flooding the entire data bus. This disruption can be achieved with messages from either a rogue or compromised RT. DoS using a message manipulation attack method is accomplished by modifying selected messages. Modifying specific messages can deny specific RTs recent information [2]. DoS using the behaviour manipulation method can have the same effects just as described, with the addition of delaying the response of the compromised RT. Additionally, DoS using behaviour manipulation can also flood the data bus, creating communication collisions and prevent all communications from occurring on the data bus.

Data leakage is defined by [2] as unauthorized transmission of data between system components. When utilizing message manipulation, a device can instruct a RT to transmit more data words than required by changing the command word, or the device can transmit additionally unauthorized data through the use of the reserved bits in a status word. When utilizing behaviour manipulation, unauthorized data can be transmitted during the idle time in the BC schedule.

A data integrity violation is the introduction of invalid data onto the MIL-STD-1553B data bus. This can be accomplished through a fake or compromised RT on the data bus [2]. Since there is no authentication of devices in MIL-STD-1553B, every RT on the bus will assume that this illegitimate communication was transmitted by the legitimate RT. These spoofing methods

allow rogue and compromised devices to provide false data, extract data and control legitimate devices.

2.2 MIL-STD-1553B Intrusion Detection

MIL-STD-1553B is a protocol designed with reliability and safety as the primary concern. Ironically, these priorities have become vulnerable due to the lack of security in the MIL-STD-1553B data bus protocol. Through the increased digitization and inter-connectivity between MIL-STD-1553B systems, there is an increase in potential for MIL-STD-1553B systems to be compromised by adversaries. The ability to accurately monitor MIL-STD-1553B bus traffic has become a requirement in order to maintain reliability, safety and integrity of these systems.

2.2.1 Intrusion Detection Overview

Intrusions are defined by Khraisat et al. [10] as any unauthorized activities that can cause damage to system. These damages can include any possible threat to the confidentiality, integrity or availability of the system. An IDS has the ability to identify these malicious and unauthorized activities on systems to allow the security of the system to be maintained. IDSs can be categorized into two main groups; signature-based IDS and anomaly-based IDS.

Signature-based IDSs use pattern matching techniques in order to detect known attacks against a system [10]. The concept behind signature-based IDS requires the designer to know the possible attacks in advance to create a pattern matching algorithm to detect unauthorized access on the system. The main drawback to signature-based systems is that knowledge of the attack is required to create the signature, therefore signature-based IDS cannot protect the system from novel attacks, though remains a complementary method to anomaly detection.

Anomaly-based IDSs were developed with the intention to overcome the limitation of signature-based IDS. An anomaly-based IDS models normal behaviour of a system and any threshold deviation from this normal observed behaviour is regarded as abnormal operation [10]. This approach has the benefit of allowing detection of unauthorized behaviour without prior knowledge of specific signature or exploits.

2.2.2 Signature-Based Detection

Bernard [4] developed a signature-based detection method for MIL-STD-1553B traffic. This signature-based method was successful in detecting command word DoS and status word data leakage attacks. However, this detection technique does not have the ability to estimate the time of the attack nor the period of the disruption. The status word data leakage attack signature was alerted whenever the reserved bits were set to non-zero values, although this would only be effective if the attacker utilized the reserved bits. The prior knowledge required to create the rule of non-zero values for the reserved bits highlight the main disadvantage to solely using a signature-based detection method.

2.2.3 Sequence-Based Anomaly Detection

Stan et al. [11] created a sequence-based anomaly detection method comparing the investigated traffic to known traffic from a specific bus implementation. The sequence-based method was implemented through the creation of a Markov chain model representing valid transitions between messages. These valid transitions were defined by the command and timing features from the known MIL-STD-1553B messages and the model was separated into peroidic and aperiodic messages. The model successfully detected the DoS anomalies and spoofing anomalies with a zero false positive rate in their testing environment. Although it was noted since the features extracted were based on command words, the model would be ineffective in detecting anomalies where only data or status words were manipulated.

2.2.4 Statistical-Based Anomaly Detection

Bernard [4] through frequency analysis demonstrated the ability to detect anomalous traffic. Command word DoS attacks were detected through RT frequency analysis, identifying when a target RT was transmitting less data than expected by the BC schedule. This detection technique does not estimate the time of the attack nor the period of the disruption.

MAIDENS [5] is a time-based histogram comparison approach that was based on the research completed by Losier et al. [12]. MAIDENS used time-based features from known MIL-STD-1553B traffic, creating a baseline histogram to represent normal traffic and a deviation threshold was chosen based on the baseline traffic. Unknown MIL-STD-1553B traffic was plotted and compared to the baseline histogram and flagged as anomalous behaviour when exceeding the threshold. The results from MAIDENS [5] demonstrated

the ability of this method to identify a period of time in which a DoS, spoofing or message manipulation attack occurred, however MAIDENS lacks the ability to directly identify the messages or the specific RT's responsible for the attack. Additionally, this method was successful in detecting spoofing attacks that only utilized data words, addressing part of the deficiency identified by Stan et al. [11].

2.2.5 Machine Learning and Deep Learning Based Anomaly Detection

Stan et al. [6] implemented their proposed RT authentication module from their previous paper [11], which utilized an ensemble method leveraging both unsupervised and supervised machine learning techniques. The RT Authentication module utilized the K-means algorithm in determining if the traffic was legitimate traffic or if a rogue RT was communicating over the data bus. Next, a classification algorithm was utilized to detect if a legitimate RT was masquerading as another RT. The ensemble approach for RT authentication was: Step 1 using the unsupervised k-means model, then step 2 using a supervised model is illustrated in Fig. 2.6. Decision tree, Gaussian naive Bayes, Random Forest, Support Vector Classification (SVC), K-Nearest Neighbors and Multi-Layer Perceptron (MLP) classification algorithms from the scikit-learn library were chosen for evaluation. The results in [6] showed MLP and SVC having the worst performance, while the remaining classifiers had comparable performance. Ultimately random forest was chosen due to having low computational requirements and high precision and recall performance. The research concluded the two part machine learning approach detected both rogue RT and spoofing RT attacks with more than 98% level of precision and recall.

Onodeuze et al. [13] also proposed machine learning and deep learning models to detect anomalous behaviour in MIL-STD-1553B traffic. The algorithms evaluated were MLP, Bidirectional LSTM (Bi-LSTM), One Class Support Vector Machine (OCSVM), Isolation Forest, Minimum Co-variance Determination (MCD), Local Outlier Factor (LOF) and AWS XGBoost. Overall, the majority of the algorithms showed poor performance, either classifying the majority of normal and malicious traffic as malicious, or the opposite and classifying all traffic as benign. The highest performing model was the OCSVM, having the highest overall scores across the datasets. Onodeuze et al. [13] did not describe the type of malicious traffic that was introduced into the datasets so it is difficult to determine the anomaly types that their models are capable of detecting.

Harlow [3] proposed implementation of a LSTM based model for anomaly

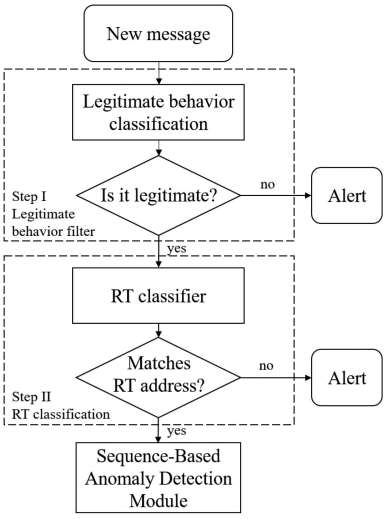

Figure 2.6: RT Authentication Module [6]

detection for MIL-STD-1553B traffic. This model is based on the method used by Taylor et al. [14] for anomaly detection in the Controller Area Network (CAN). Currently, Harlow's [3] initial research shows initial success in implementing this method for MIL-STD-1553B anomaly detection. Additionally, Harlow's [3] model and dataset are utilized in this research and are described further in 2.4.3 and 3.2 respectively.

2.3 Feature Engineering

A feature in the context of machine learning is simply an individual measurable property or characteristic of the object being observed [15]. A feature is derived from raw data and acts as an input into our model. The aggregation of features form the dataset we use to train the machine learning models to predict outcomes. Since the features are the building blocks of our datasets, their generation and selection have a major impact on the overall quality and efficiency of the model being created. In order to assist with the selection of features, researchers use both expertise and algorithms to determine the optimal features used to train a model. In this paper, the general definition of feature engineering will be used and will include the topic of feature generation, feature selection and demensionality reduction as seen in Fig. 2.7.

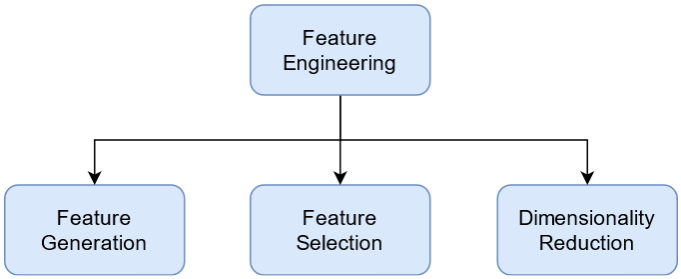

Figure 2.7: Feature Engineering

2.3.1 Feature Generation

Feature generation is the process of creating new input variables from available data. There are many different approaches for feature engineering and they are specific to the type of data utilized. An example of this is McGaughey et al, [16] use of the processing module netAI [17] to create network specific flow features. Stan et al. [6] also implemented the use of a feature generation module called tsfresh [18], [19] which allows the calculation of features from time-series datasets. Brownlee [20] highlights other feature generation methods such as polynomial transformation. Polynomial transformation utilizes simple mathematical operations to create additional features that may transform the data into more effective features. The polynomial transformation method was also utilized by McGaughey et al. [16] on their set of flow features to create further derived feature sets. The results demonstrated that these features created through polynomial transformation allowed the model to perform predictions with a higher level of accuracy.

2.3.2 Feature Selection

Feature selection is the technique of selecting a subset of features that will provide the most relevant data for input into the model [20]. Feature selection can be grouped into two main categories as seen in Fig. 2.8; unsupervised and supervised. Unsupervised feature selection does not use labelled data whereas supervised feature selection does use labelled data. Supervised feature selection can further be grouped into three categories; intrinsic, wrapper and filter methods [21]. Intrinsic feature selection refers to machine learning models that have embedded processes for selecting the best features, such as Least Absolute Shrinkage and Selection Operator (LASSO) [22] which uses penalization functions or decision trees. The wrapper feature selection method recursively

selects a subset of the features, trains the model on these features then evaluates the performance. Lastly the filter method selects features independent of the machine learning algorithm, and instead uses statistical methods to determine which features to use. In this section, the following three feature selection methods will be explained: ANOVA (filter method), PPS (intrinsic method) and FOS (filter method).

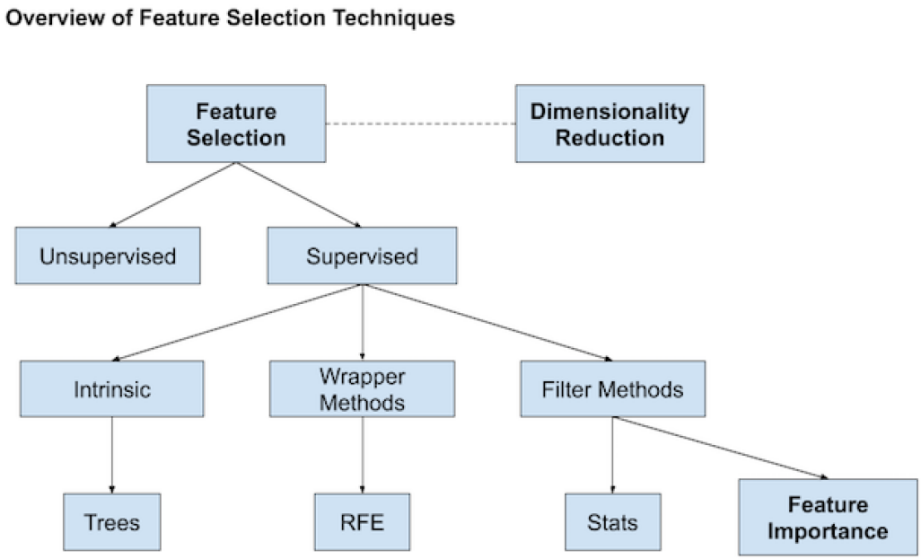

Figure 2.8: Overview of Feature Selection Techniques [20]

Analysis of variance

Correlation is a common filter feature selection method to measure the dependency between two features and how strongly the features relate. The type of statistical measure implemented depends on the type of input data and type of problem: classification or regression. Fig. 2.9 shows common methods for statistically measuring the correlation between data points. The anomaly detection in MIL-STD-1553B is considered a binary categorization problem, therefore the ANOVA is an applicable feature selection technique. ANOVA calculates the linear correlation coefficient through a F-statistic method. A F-statistic is a ratio between two variances, and ANOVA is specifically calcu-

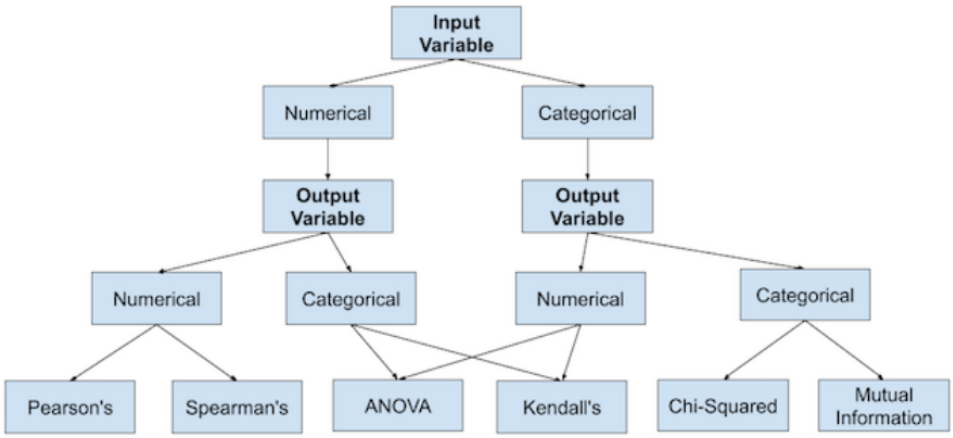

Figure 2.9: Filter Feature Selection Methods for Machine Learning [20]

lated using:

$$F = \frac{MSB}{MSW} \tag{2.1}$$

Where F = ANOVA coefficient, MSB = Mean sum of squares between the groups and MSW = Mean sum of squares within the groups.

Predictive Power Score

PPS is an intrinsic feature selection method which is relatively new, and designed by 8080labs [23] and shows good results in recent research [24], [25]. This method allows both linear and non linear as well as categorical and numerical variables to be utilized and evaluated. In the case of a classification problem, PPS evaluates features by using a decision tree classifier and the resultant F1 score as the metric. Decision trees are a type of supervised machine learning which have an intrinsic feature selection method. PPS uses the decision tree models to select the highest performing features following a similar approach utlized by Adhao et al.[26]. Decision trees work by creating a tree that consists of nodes, branches and leafs as seen in Fig. 2.10 . A node represents a feature attribute, a branch represents a decision rule and each leaf represents an outcome. Decision trees use algorithms to determine if a node is to be created and which feature attribute is to be used. After the decision tree model is used, PPS then calculates the F1 score by using a baseline score $F1_{naive}$, which is the maximum of:

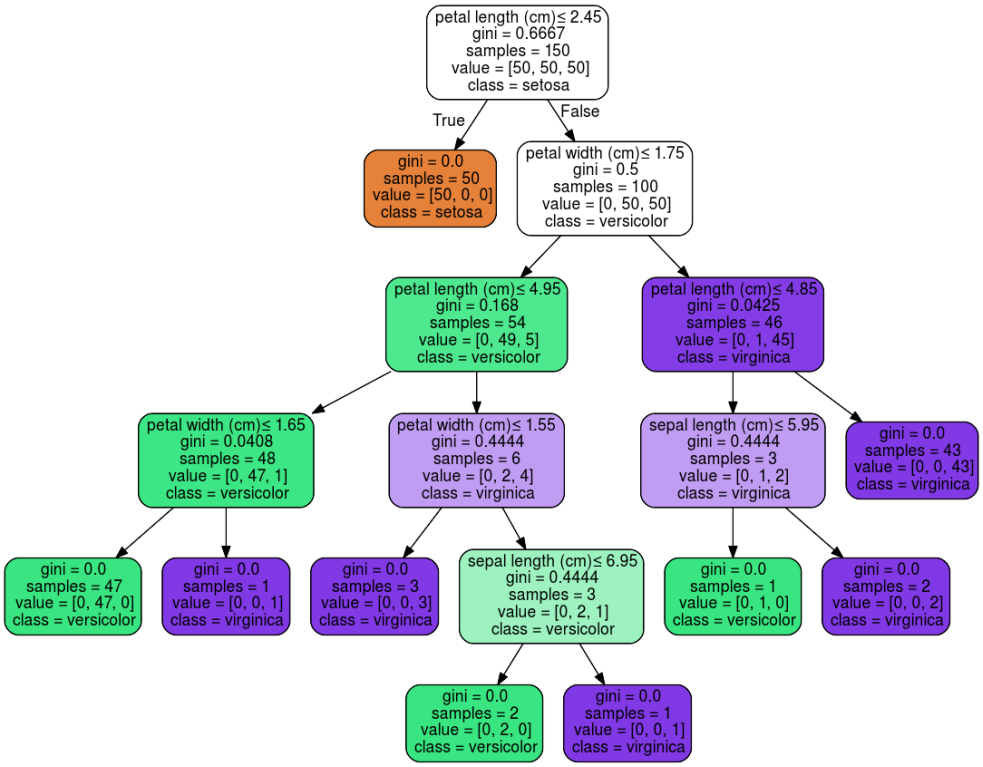

Figure 2.10: Decision Tree Classifier using Iris Dataset [27]

1. $F1_{most\ common}$: The F1 score for a model that always predicts the most common class of the target.
2. $F1_{random}$: The F1 score of a model that predicts random values.

PPS then uses this baseline $F1_{naive}$ score along with the $F1_{model}$, score of the feature to create the PPS as seen in the equation below.

$$PPS = \frac{F1_{model} - F1_{naive}}{1 - F1_{naive}} \tag{2.2}$$

Similar to other correlations methods, this PPS score then has to be compared to the score of other features and a threshold has to be selected in order to select features with the highest value. The F1 score will be further explained in 2.4.4

Fast Orthogonal Search

FOS is algorithm that was developed by Krorenberg et al. [28] as an autoregressive process for modeling time-series data. This process was initially used for modeling biological data. McGaughey et al. [16] researched its use as a feature selection method in time-series network traffic. FOS demonstrated its ability as an effective filter feature selection method in this instance, allowing it to outperform the Best First Search method in performance and selecting features that allowed the classifier to have better prediction. The ability of FOS to select features from non-orthogonal candidates through the process of building an internal orthogonal model makes it a novel candidate for feature selection on the time-series traffic on a MIL-STD-155B data bus. The FOS algorithm is a recursive algorithm which loops through features, selecting features that have the highest Mean Squared Error (MSE) reduction value, Q, to be part of the feature set as seen in Fig. 2.11. FOS indirectly assesses the value by successively adding terms and defining the recursively computed vector C and matrices D and α as follows:

$$C(m) = \overline{y(n)P_m(n)} - \sum_{r=0}^{m-1} \alpha_{mr} C(r),\ m = 1, \ldots, M \tag{2.3}$$

$$D(m,r) = \overline{P_m(n)P_r(n)} - \sum_{i=0}^{r-1} \alpha_{mi} D(m,i),\ m = 1, \ldots, M\ ;\ r = 1, \ldots, m \tag{2.4}$$

$$\alpha_{mr} = \frac{D(m,r)}{D(r,r)},\ m = 1, \ldots, M\ ;\ r = 0, \ldots, m-1 \tag{2.5}$$

As m increases C, D and α grow in the form of a Cholesky decomposition. For each successive m^{th} term, the corresponding coefficient for the orthogonal basis is computed by:

$$g_m = \frac{C(m)}{D(m,m)},\ m = 0, \ldots, M \tag{2.6}$$

Next, the reduction in MSE from adding the m^{th} term is calculated as follows:

$$Q = g_m^2 D(m,m) \tag{2.7}$$

This MSE reduction value, Q, is then stored in order to be compared against the Q value of other features. This process is then repeated until the M^{th} feature is reached. FOS then selects the feature with the highest Q value,

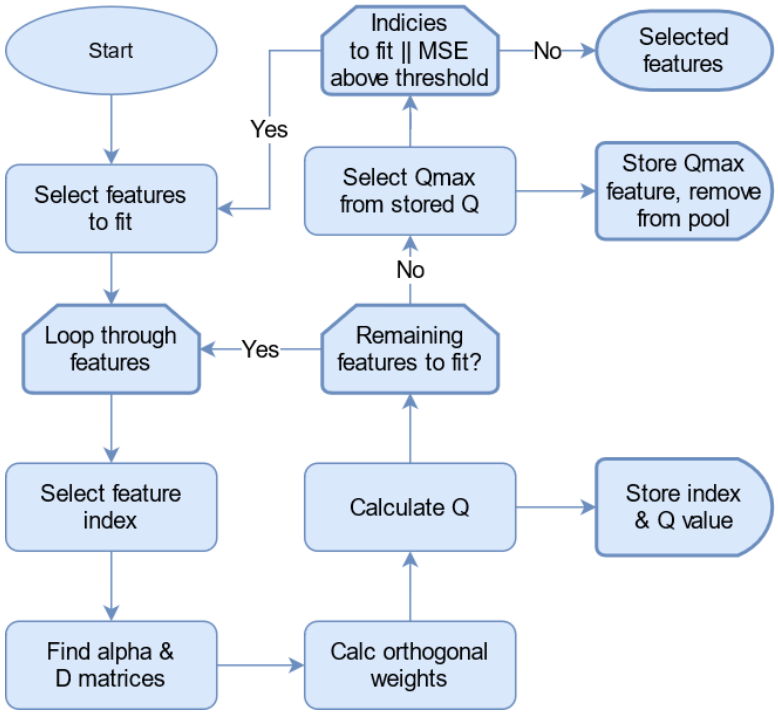

Figure 2.11: Flow diagram of Fast Orthogonal Search

removes that term from the remaining features, and repeats the above process until a threshold for MSE is reached or there are no more features to fit.

To help illustrate how FOS works, consider a simple example with a 3X3 matrix below.

$$Labelled\ features = \begin{bmatrix} 2 & 8 & 7 \\ 3 & 6 & 3 \\ 5 & 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

We will take the first feature column, P_1 and create the two matrices D and α using 2.4 and 2.5 respectively.

$$D = \begin{bmatrix} 1 & 0 \\ 3.33 & 1.56 \end{bmatrix} \quad \alpha = \begin{bmatrix} 1 & 0 \\ 3.33 & 0 \end{bmatrix}$$

From matrices D and α we can now calculate C_1 and g_1 from 2.3 and 2.6.

$$C_1 = 0.22 \quad g_1 = 0.14$$

We then can calculate, from 2.7, that Q = 0.032. This Q value is then stored, and the above process is repeated for the remaining features P_2 and P_3. The

feature with the highest Q value is then selected and matrix D and C are then updated with the selected feature column. The selected feature column is then removed from matrix P, and the above process in its entirety is calculated again, but this time only with the remaining two feature columns.

2.3.3 Dimensionality Reduction

Dimensionality in machine learning is considered the number of input feature types in a given dataset. Bellman [29] states that a large number of input features can make the task of creating machine learning models more challenging and is generally referred to as the curse of dimensionality. Feature selection is a method to reduce the number of features utilized and dimensionality reduction is another method that can be used to reduce the number of input features, either in conjunction with, or separate from feature selection. Dimensionality reduction reduces the number of inputs by projecting the data into a lower dimensional space, while still preserving the information contained in the data [30]. This reduction of input variables while keeping the information allows the models to become simpler, which results in less overfitting and reduced computational requirements. The main categories of dimensionality reduction are manifold learning, model based and matrix factorization, with examples of each method seen in Fig. 2.12.

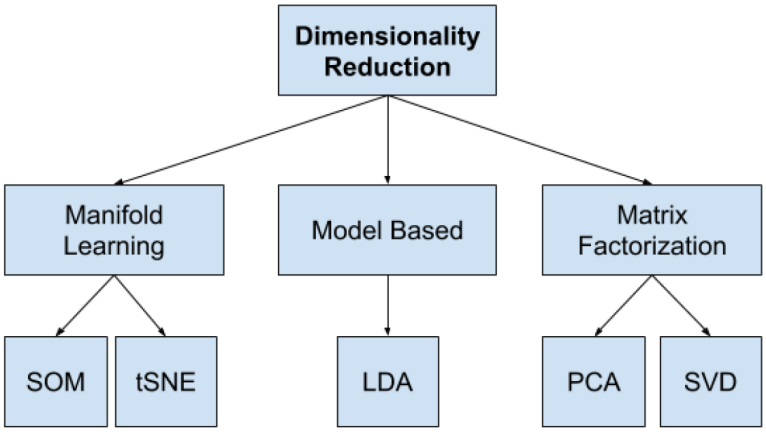

Figure 2.12: Overview of Dimensionality Reduction Techniques [20]

2.3.4 Uniform Manifold Approximation and Projection

UMAP is a novel method for dimensionality reduction researched by McInnes et al. [7]. UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph through "fuzzy simplicial set representations". UMAP accomplishes this by extending a radius outwards from each point and connecting points when those radii overlap. UMAP then makes the graph "fuzzy" by decreasing the probability that two points are connected as the radius grows. Then, by stipulating that each point must be connected to at least its closest neighbor, the local structure is preserved in balance with global structure [31]. Fig. 2.13 depicts the radii and connected points using UMAP, resulting in a cluster visualization for high dimensional datasets. Having been created in 2020, UMAP does not have any documented implementation in the IDS field, but has been used dimension reduction for other temporal data [32]. This initial research into the UMAP technique shows it offers advantages of increased speed and better preservation of the data's structure in comparison to similar techniques [7], [32].

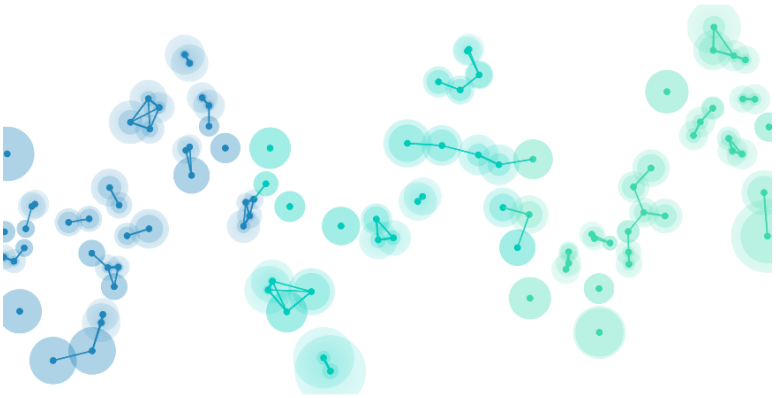

Figure 2.13: Representation of Connecting Points using UMAP [31]

2.4 Deep Learning

Deep learning is a subset of machine learning that refers to a class of models that use multiple layers of simple statistical components to learn the representation of data [33]. Deep learning can utilize supervised, unsupervised or semi-supervised methods for model creation. Deep learning models consist of multiple layers of processing units called neurons, which interpret the data

being input into the model. Neurons interpret the data by taking the weight inputs from either the initial input, or from a previous layer and feeding them through an activation function. If the neuron activates, it then will feed its information forward to the neurons in the next layer, or as an output target. There are many different types of activation functions and their use depends on the problem and desired outcome, such as regression or binary classification. Current research using deep learning models for anomaly detection in MIL-STD-1553B traffic show mixed results, ranging from poor performance [6], [13] to effective anomaly detection [3]. This research will seek to recreate and improve upon the effective anomaly detection demonstrated by Harlow [3].

2.4.1 Long-Short Term Memory

An LSTM network is a subset of Recurrent Neural Network (RNN). A RNN can evaluate data from the previous input and current input, making it a powerful tool in sequential data modeling. The major drawbacks of vanilla RNNs are the vanishing gradient and exploding gradient problems. The vanishing gradient problem makes the RNN poor at remembering long sequences and forgets the initial historic input as seen in Fig. 2.14. Conversely, the exploding gradient problem is when the historic input overwhelms the remaining inputs and render the model ineffective.

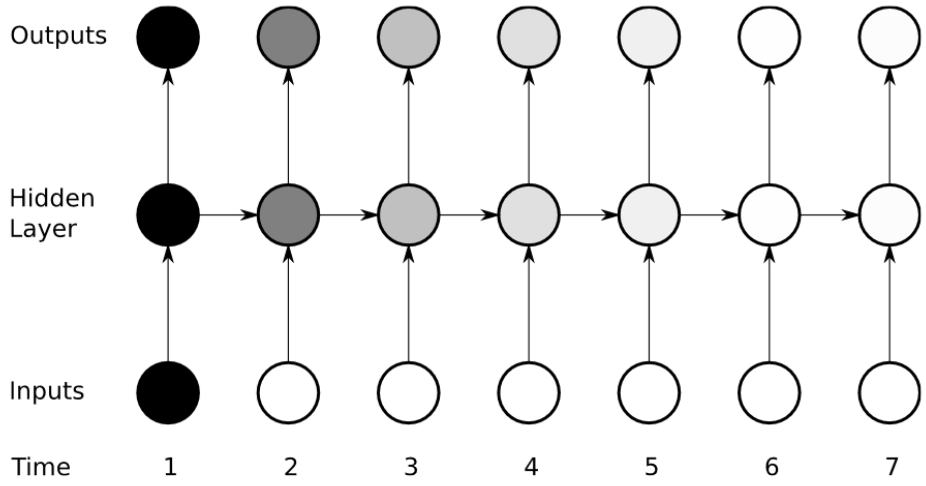

Figure 2.14: Vanishing Gradient for RNNs [34]

LSTM networks were developed to solve the vanishing and exploding gradient issues with vanilla RNNs [35]. A LSTM utilizes a memory cell that allows it to store and access information over longer periods of time, mitigating the vanishing gradient problem [34]. The memory cells consist of an input gate, output gate and a forget gate that control the preservation of information. Fig. 2.15 depicts a simplified version of a LSTM network illustrating the main components:

1. Nodes: Each node is represented either as fully activated (black) or inactive (white).
2. Input Gate: Shown below each node in the hidden layer.
3. Forget Gate: Shown to the left of each node in the hidden layer.
4. Output Gate: Shown above each node in the hidden layer.

For simplicity each gate will be either open (O), or closed (–). The LSTM cell will retain the first input as long as the forget gate is open and the input gate is closed. Additionally the sensitivity of the output layer is controlled by opening and closing the output gate which does not affect the memory of the cell [34].

LSTM properties for sequential data prediction make them a viable candidate for anomaly detection of time-based systems. Taylor et al. [14] implemented a LSTM network for anomaly detection on a CAN bus network, Kundu et al. [36] implemented a LSTM network on industrial control network, both demonstrating successful results in the detection of anomalies in real-time systems. Additionally, Harlow's [3] implementation of a LSTM network for a MIL-STD-1553B anomaly detector provided successful results.

2.4.2 LSTM Network Hyper-parameter Tuning

Deep learning models have many different attributes, called hyper-parameters that are used to modify and tune the model in order to increase performance. Hyper-parameters are variables that are set by the creator for tuning the model, whereas parameters are values which are estimated from the data itself automatically, such as weights in a neural network. Some of the more common LSTM hyper-parameters consist of the following:

1. Number of neurons and layers: This is the number of neurons that make up the individual layers, and number of layers are in the network. There is no set rule as to the number of neurons and number of layers, as it is dependant on the type of data and model that is created.
2. Batch size: This is defined as the number of samples to iterate through before updating the internal model's parameters.
3. Number of epochs: The number of times the algorithms will iterate over the entire dataset.

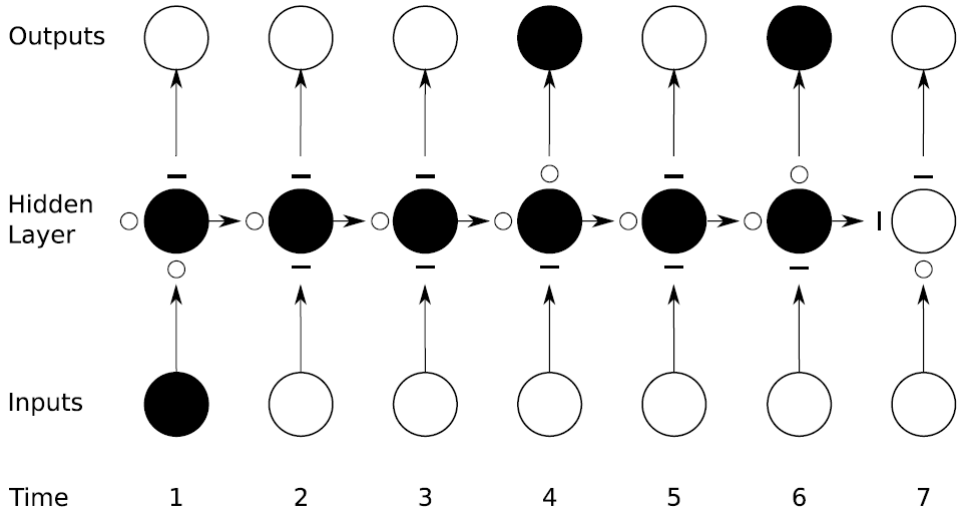

Figure 2.15: Preservation of Gradient Information by LSTM [34]

4. Learning rate: As in traditional machine learning, this controls the rate in which the model is adapted to the problem. Small rates require more epochs due to the smaller adjustments, whereas large rates can adjust rapidly but can sometimes overshoot the optimal solution.
5. Dropout: This is a regularization method in which neurons are randomly ignored, which forces the neurons to take on more or less responsibility. Dropout in turn reduces over-fitting and allows a model to become more robust.
6. Bi-LSTM layer: When there are two layers implemented alongside each other in opposite directions, allowing the network to read the input from past to future as well as future to past. The bi-directional aspect allows them to also consider elements in the future as they are processing them in the opposite direction.
7. Attention mechanism: A mechanism that allows the model to map the importance and associations between input sequences, enabling the model to focus on certain inputs with more "attention" by assigning higher weights to these inputs.

2.4.3 LSTM Autoencoder

An LSTM autoencoder uses an unsupervised LSTM which learns the compressed representation of the data [37]. This it done by having the model

encode the data through a purposefully created bottleneck, and then reconstruct the data through the decoding phase as seen in Fig. 2.16. In the case of

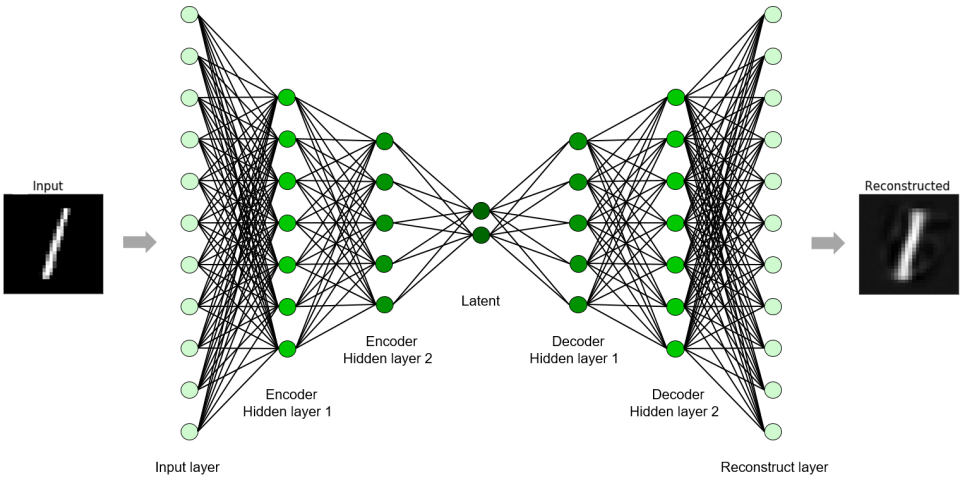

Figure 2.16: LSTM Autoencoder [37]

anomaly detection systems, these LSTM autoencoders are trained on known "good" traffic to reduce the reconstruction error of the data. The trained model's resultant reconstruction error is then is then used to set a threshold value for the acceptable reconstruction error. Harlow's [3] model was created using the tensorflow platform, consisting of three LSTM layers. The model's input consisted of 155 features, followed by a 30 node LSTM encoding layer, a 15 node LSTM bottleneck layer, a 30 node LSTM decoding layer followed by the output as seen in Fig. 2.17. This base model kept all other hyperparameters unchanged. The reconstruction error threshold chosen is based on Mean Absolute Error (MAE) loss on the training data, and in the case of Harlow's research [3] the threshold was set to 10% above the MAE loss. The trained model is then used with unknown traffic, and any traffic that exceeds the reconstruction error threshold value is deemed anomalous traffic. Provotar et al.'s [37] use of LSTM autoencoders on false data injection attack proved effective in detecting anomalous behaviour. Additionally, Harlow's [3] use of LSTM autoencoders on MIL-STD-1553B data bus traffic also proved effective in detecting anomalous behaviour.

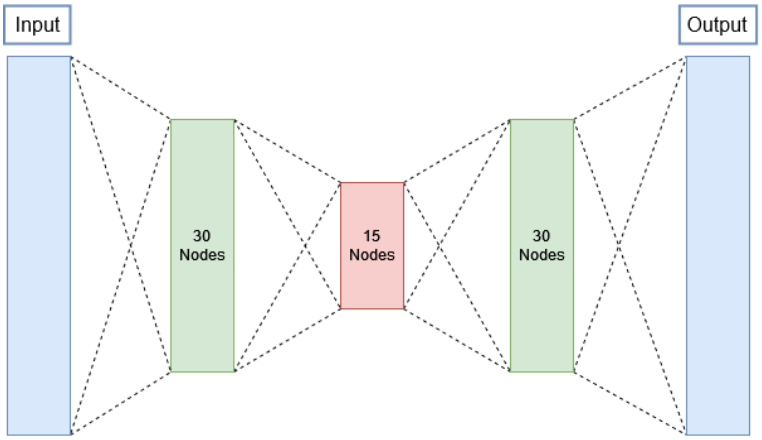

Figure 2.17: LSTM Autoencoder Model

2.4.4 Evaluation Metrics for Deep Learning

The effectiveness of a deep learning network can be measured by several metrics. The metrics to be discussed are derived from the confusion matrix seen in Fig. 2.18. A confusion matrix is composed of four values:

1. True Positive (TP): An anomalous event that has been classified correctly.
2. False Positive (FP): A benign event that has been classified incorrectly as an anomalous event.
3. False Negative (FN): An anomalous event that has been classified incorrectly as a benign event.
4. True Negative (TN): A benign event that has been classified correctly.

Using the values from the confusion matrix, metrics can be created such as accuracy, precision, recall, AUROCC and MCC. Accuracy is a common method to assess how often a model classifies data correctly and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

Precision is a metric that assesses how often positive classifications were correct and is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

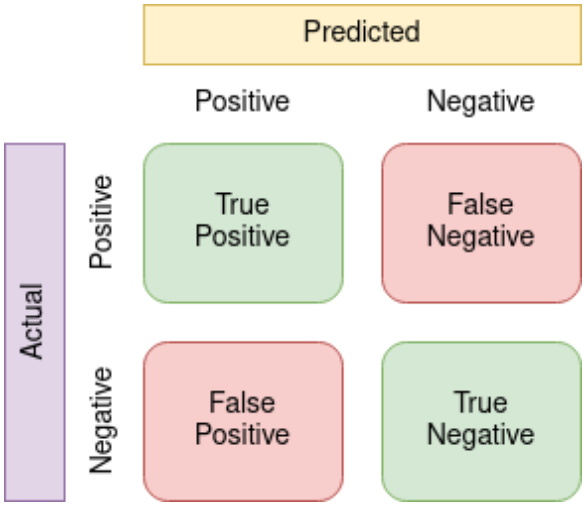

Figure 2.18: Confusion Matrix

Recall is a metric that assesses the proportion of actual positives that were correct and is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

The F_1 score combines precision and recall into a single metric and is the harmonic mean. This results in a high score if both precision and recall are high, a low score if both are low and a medium score if one is high and the other is low. The F_1 score is calculated as follows:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.11}$$

AUROCC is an evaluation method that illustrates the ability for the model distinguish between the classes. In order to calculate AUROCC, a Receiver Operating Characteristic Curve (ROCC) is created by plotting the True Positive Rate (TPR) and False Positive Rate (FPR). The TPR isdefined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{2.12}$$

The FPR is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{2.13}$$

These values are then plotted and then the AUROCC is found simply by calculated the area under the curve. MCC is a another method for evaluating performance of a model. MCC is calculated as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.14}$$

MCC aims to overcome the issues associated with accuracy and F_1 score with unbalanced data. Chicco et Jurman's research [38] showed overall more reliable statics using the MCC over the F_1 score.

2.5 Summary

In this chapter a detailed overview of the MIL-STD-1553B data bus protocol, vulnerabilities and current research for MIL-STD-1553B IDS were provided. Recent research outlined signature based detection methods, sequence-based methods as well as deep learning based anomaly detection methods. Next, feature engineering methods for MIL-STD-1553B traffic were discussed which included feature generation, feature selection and dimensionality reduction. Furthermore, current deep learning methods utlizing LSTM and LSTM autoencoder models were outlined and their effectiveness in detecting anomalous behaviour was highlighted. Additionally, LSTM hyper-parameters and their effect improving model performance were discussed. Finally, the metrics used to evaluate the effectiveness of deep learning models were presented.

3 Methodology and Design

In this chapter, the experimentation design is described in four phases. The initial phase is data collection and munging, describing the datasets used as well as the data manipulation to prepare it for the downstream processes. The second phase is feature engineering, outlining the tools and methods used to generate features, select features and reduce the dimensions of the datasets. The third phase outlines the method used to train the LSTM model for anomaly detection. The final phase will discuss the method in which the results are validated against the aim of this research.

3.1 Deep Learning Pipeline

In order to understand how model effectiveness may be impacted by feature engineering, the entire pipeline for a deep learning anomaly detector needs to be designed and constructed. The same MIL-STD-1553B datasets Harlow [3] collected using Abaco Bus Tools were used as the starting point for this pipeline. The new feature engineering component of the deep learning pipeline is built on Harlow's model [3] and augmented using feature generation, feature selection and feature reduction techniques. The feature generation component includes two different techniques to create new features based on the initially extracted feature set. The first feature generation method chosen was polynomial expansion that leverages the sklearn library. The second technique is based on time series characteristics using tsfresh, an application also used by Stan et al. [6]. After feature generation, selecting the most useful features would then need to be conducted. This is approached in two ways, first with feature selection, then with dimensionality reduction. The feature selection methods selected are ANOVA, FOS and PPS. The dimensionality reduction method selected, UMAP, was introduced by McInnes et al. [7]. These datasets are then processed using the design from Harlow's [3] anomaly detection LSTM auto-encoder. The evaluation of the models utilize common

evaluation metrics in the machine learning field. These metrics are then compared against previously completed research in order to validate the model and determine if there is a marked improvement in effectiveness.

3.2 Data Collection and Munging

The raw data collected by Harlow [3] forms the datasets used for this research. This data includes both baseline data and anomalous data, the latter which was created using the evaluation tool created by Paquet [1]. The datasets can then be further divided into into two main groups based on anomalous or attack types: DoS and data integrity violation. The dataset details are as follows:

1. DoS
    a) Network Disruption: A single dataset that shows flooding the network with status words.
    b) RT Deny: A single dataset where there is a targeted DoS on RT18.
    c) RT SA Deny: Two separate recordings of datasets where there is a targeted DoS on RT18 and subaddress 1.
2. Data Integrity Violation
    a) RT Hijack: One recording of a dataset where a data word is injected into the network traffic from RT18, subaddress 6 and word number 56.

These datasets have been collected using the Abaco BusTools-1553 software suite in a simulated aircraft environment. Abaco BusTools-1553 records data bus traffic in the Bus Monitor Data Files Extended (BMDX) message format as seen in Fig. 3.1. Within each of the n messages, the data contained in the words is in the format outlined in Fig. 3.2. Since this data structure is simply the combined data from each message, it is munged to provide more granular information such as RT address, transmit/receive, subaddress and number of words to better correlate to how words are typically presented in MIL-STD-1553B, as seen in Fig. 3.3. This is accomplished by separating the data contained in the command word(s), data word(s) and/or status word(s) into their respective formats as outlined in Fig. 2.2.

Using the recorded response times from the BMDX file as seen in Fig. 3.2, an intermessage response time is calculated to supplement the already provided RT intramessage response time(s). This is done in order to create a time feature representing the duration between messages. This intermessage response time is the sum of the previous message's initial transmit time and the intramessage response time(s).

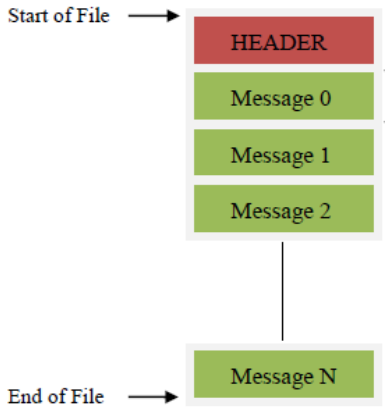

Figure 3.1: Abaco BusTools-1553 BMDX Structure [9]

```
typedef struct api_bm_mbuf
  {
  BT_U32BIT         messno;     // Message number (generated by API, 1-based)
  BT_U32BIT         int_status; // Interrupt status from board
  BT1553_TIME       time;       // Time of message (64-bits, 1 us  or 1ns LSB)
  BT1553_COMMAND    command1;   // 1553 command word #1 (Receive for RT-RT)
  BT_U16BIT         status_c1;  // 1553 command word #1 error status
  BT1553_COMMAND    command2;   // 1553 command word #2 (Transmit for RT-RT)
  BT_U16BIT         status_c2;  // 1553 command word #2 error status

  BT1553_BMRESPONSE response1;  // 1553 response time #1 (byte)
  BT1553_BMRESPONSE response2;  // 1553 response time #2 (byte)
  BT1553_STATUS     status1;    // 1553 status word #1 (Transmit for RT-RT or Broadcast RT-RT)
  BT_U16BIT         status_s1;  // 1553 status word #1 error status

  BT1553_STATUS     status2;    // 1553 status word #2 (Receive for RT-RT, NULL for BCST RT-RT)
  BT_U16BIT         status_s2;  // 1553 status word #2 error status

  BT_U16BIT         value[BT1553_MBUFCOUNT];   // 1553 data words
  BT_U16BIT         status[BT1553_MBUFCOUNT];  // 1553 status for data words (16-bit words)
 }
API_BM_MBUF;
```

Figure 3.2: Abaco BusTools-1553 BMDX Message Structure [9]

Msg #	CMD1	Err/Sts	STS1	Err/Sts.2	RSP1	DATA01	E/S
1	10110010011111	0	10100000000000	0	6	0x4CFC	0x00
2	10110011011111	0	10100000000000	0	6	0x4CFC	0x00
3	10110101101111	0	10100000000000	0	6	0x4CFC	0x00
4	101010111100101	0	101000000000000	0	6	0x0000	0x00
5	10110010000001	0	10100000000000	0	6	0x4CFC	0x00

Msg #	CMD1 addr	CMD1 TR	CMD1 subaddr	CMD1 numword	RSP1	STS1 addr	STS1 Error	STS1 Inst	STS1 SerReq	STS1 Reserved	STS1 BCRecv	STS1 Busy	STS1 SubFlag	STS1 DBAcc	STS1 TerFlag
1	5	1	4	31	3	5	0	0	0	0	0	0	0	0	0
2	5	1	6	31	3	5	0	0	0	0	0	0	0	0	0
3	5	1	11	15	3	5	0	0	0	0	0	0	0	0	0
4	10	1	15	5	3	10	0	0	0	0	0	0	0	0	0
5	5	1	4	1	3	5	0	0	0	0	0	0	0	0	0

Figure 3.3: Abaco BusTools-1553 Data Munging

3.2.1 Data Labelling

In order to utilize supervised feature selection methods, the anomalous datasets require labelling. Due to the use of the pre-created dataset, the data is manually labelled using timestamps and attack parameters recorded by Harlow as well as the implied schedule of the system. The labelling method was different for each of the main groups: DoS and data integrity violation.

The DoS labelling method assumes that any message not part of the baseline schedule is considered anomalous. The message information comprising of the RT address, transmit/receive flag, and subaddress is used to create this baseline. The premise behind this assumption is that MIL-STD-1553B utilizes a predictable real-time schedule and the messages transmitted on the bus should not deviate from that during normal operation. Additionally since our simulated environment has a limited schedule it was noted there were only 80 unique messages in the baseline datasets by filtering on the previously discussed message information. The assumption that the baseline messages will not deviate from the schedule is made as Abaco BusTools-1553 attempts to interpret the MIL-STD-155B words on the databus and create a message based on that data. Since Abaco BusTools-1553 was a tool developed for design and development of production MIL-STD-1553B RTs, it assumes they will adhere to the standard and as such the output can misrepresent actual traffic on the databus. Application of the DoS technique could cause a collision and the message would be rejected by the RT, although in the simulated setup, Abaco BusTools-1553 may attempt to "read" the words and state there was a message sent. In this case, the message would then not match any of the messages from the baseline schedule. Additionally, the attack may also collide

partway through a message, where the initial part of the message would be intact but the remaining data would not match the baseline. In both these scenarios, the message is then labelled as anomalous.

In the case of the data integrity violation dataset, the messages are intercepted and sent with altered data. The method used to label this dataset assumed that all messages that matched the targeted message, its subaddress and was within the attack time as noted by Harlow [3] is considered as anomalous. These methods of labeling will not result in a perfectly labelled dataset as some messages may have been mis-labeled, although the impact of this is assumed to be minimal due to the real-time nature and predictability of the baseline traffic.

3.3 Feature Engineering

Feature engineering is the next step in the pipeline in order to prepare the data for use in the anomaly detection model. This step first consists of feature generation followed by feature selection and dimensionality reduction as outlined in Fig. 3.4.

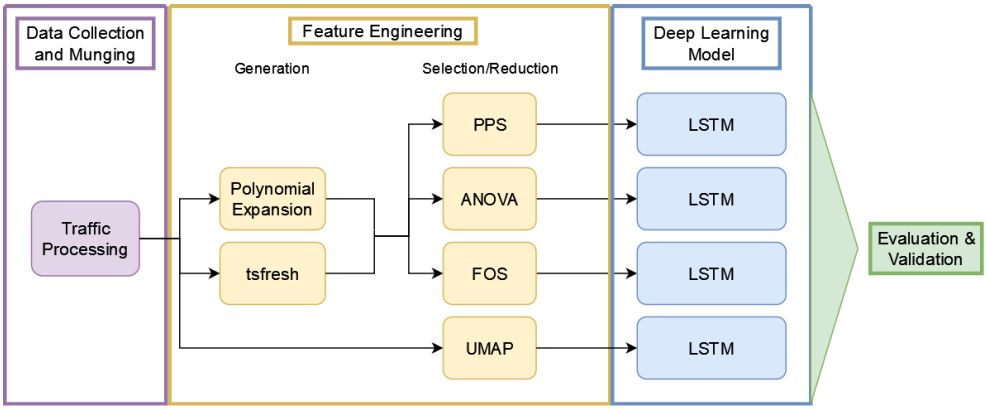

Figure 3.4: Feature Engineering Focused Pipeline

3.3.1 Feature Generation

In this step, the primary features created in the data collection and munging step are used to generate multiple new features. The first method of feature generation utilizes the scikit learn *sklearn.preprocessing.PolynomialFeatures* module [27]. This module generates a new feature matrix with all polynomial

combinations of the features up to the degree specified. For the purpose of this research the maximum polynomial degree of two is used due to processing speed and resource consumption. The second method utilizes the python tool tsfresh. tsfresh is a python package that generates time-series features using 78 different feature calculation modules [18], [19] as seen in Appendix A. tsfresh does this by running the different calculation modules with the data from the features in the dataset, and then creates additional features based on the results from the calculation modules.

3.3.2 Feature Selection and Reduction

After feature generation, feature selection and a dimensionality reduction method is used in order to select a smaller number of features to be used as input into each discrete run of the learning model. The feature selection methods ANOVA, PPS and FOS are used to select a subset of features in parallel from the same starting features created in the previous phase. This will allow for comparison of each model's results as seen in Fig. 3.4. Additionally, the UMAP dimensionality reduction method is used in parallel with the other feature selection methods, as illustrated in Fig. 3.4 in order to allow another data point for comparison. This allows the evaluation of UMAP's ability to reduce the number of features while still maintaining effectiveness.

3.4 Anomaly Detection Deep Learning Model

The final step of the anomaly detection pipeline is the training and use of the LSTM as an anomaly detector. The model is based on Harlow's [3] LSTM implementation. First, the model created by Harlow is replicated utilizing the baseline dataset using only the primary features, which becomes the reference for evaluating the effectiveness. Next sixteen models are created. Twelve of these models are attack specific, created from four distinct attack types and three feature selection techniques against the original and generated feature set. The remaining four are general models. Three are based on features identified across all four attack types using the three selection techniques and the full feature set. The fourth general model is based on the dimensionality reduction technique that processed only the original feature set and did not consider attack type. These general models enable the evaluation of a single, more attack agnostic model to effectively detect anomalous traffic when its performance metrics are compared to those of the specific models, possibly allowing for the streamlining of model creation.

After the training of these models, they will be analyzed and a threshold value is determined based on the MAE value of the baseline traffic. The trained models are then tested using the anomalous traffic datasets, where the results are then compared. With all the results from the multiple models and the reference model from Harlow's [3], the effectiveness of the feature engineering techniques can be directly compared.

3.5 Evaluation of Deep Learning Pipeline

In order to measure the effectiveness of the anomaly detection method the results were evaluated using multiple methods. The methods utilized metrics derived from the confusion matrix metric: precision, recall, accuracy, AUROCC and MCC. These metrics were used to measure the performance of the anomaly detector. These results were then compared against recent work in the field, specifically Harlow's [3] original LSTM detector.

3.6 Summary

In this chapter, the methodology for improving the feature engineering aspects of Harlow's LSTM anomaly detector through the use of feature engineering was presented. The data collection method and the data munging techniques were discussed. Additionally, the feature engineering process, including feature generation, selection and dimensionality reduction techniques used were presented. The created feature sets are then used with Harlow's model [3], selecting the feature set that provides the best results. Finally, the evaluation of the anomaly detection pipeline was presented along with how the results are validated.

4 Results

In this chapter, the anomaly detection pipeline is built and evaluated using the design outlined in chapter 3. The experimental design is detailed, followed by a discussion on the creation of the initial dataset used in the experimentation. Next, the feature generation process for producing the extended datasets is discussed. Then the output of the feature selection and reduction techniques are provided as are the details regarding the different sets of models used in the experimentation. Finally the metrics from the multiple models are discussed and compared to current work.

4.1 Experimental Design

The anomaly detection pipeline described in the previous chapter was implemented using the following hardware and software components:

- Processors: Two Intel Xeon Gold 6230, 40 cores total
- RAM: 768 GB DDR4 memory
- GPU: 8 NVidia RTX 2080Ti
- SSD: 1.92 TB NVMe Gen 3
- Operation System: Ubuntu 20.04
- NumPy package
- Pandas library
- Scikit-learn toolkit
  - metrics
  - preprocessing.PolynomialFeatures
  - feature_selection.SelectKBest
  - feature_selection.f_classif
- PPScore tool
- UMAP tool
- Matplotlib library
- tsfresh tool
- TensorFlow library

4.1.1 Datasets

As discussed in Section 3.2 the data collected comprised both of baseline and anomalous datasets. All of the datasets were representative of an aircraft in the cruise phase of flight. There was a total of three baseline datasets collected, all of which utilized the same master schedule. Through investigation of these three benign datasets and as expected, it was confirmed that they were identical. As such, only one baseline dataset (baseline inflight 5 250820) was selected for use in this experiment as the others would not contribute any additional information. The anomalous data collected comprised of a total of five datasets across two attack categories as outlined in Section 3.2.

1. DoS
    a) NetDisrupt statusword 250820 (disrupt): Network disruption
    b) RT-SA deny statusword rt18 sa32 250820 (deny): RT deny
    c) RT-SA deny statusword rt18 sa1 250820 (SA deny): RT subaddress deny
    d) RT-SA deny statusword rt18 sa1 rec2 250820 (SA deny 2): RT subaddress deny
2. Data Integrity Violation
    a) Hijack rt18 sa6 w56 250820 (hijack): RT hijack

Through an investigation the SA deny and SA deny 2 datasets contained similar data. Due to the similar data in the SA deny and SA deny 2 datasets, SA deny was utilized for feature selection and SA deny 2 was used solely for generating model performance metrics. These anomalous datasets are named using the following convention: the first part is the type of attack, followed by the associated RT details and ending in the day, month, year recorded. To avoid confusion these datasets will now be referred to by their abbreviated forms outlined in parenthesis above. As mentioned previously, these datasets were collected in a simulated lab environment by Harlow[3] utilizing an evaluation tool created by Paquet[1]. Both the baseline and anomalous datasets contain just over 1 million MIL-STD-1553B messages each. The anomalous datasets are imbalanced, containing little anomalous traffic compared to normal traffic. The specific amount of anomalous traffic for each dataset is as follows: disrupt - 17.2%, deny - 2.1%, SA deny - 5.7%, SA deny 2 - 4.2%, and hijack - 17.2%.

The data was then munged to create meaningful features. As previously shown, Abaco BusTools records data based on the structure in Fig. 3.2. The command words and response words are broken out into their individual components as explained in Section 3.2. Additionally, all the errors interpreted by Abaco BusTools are recorded in a single feature as shown in Table 4.1. Abaco

Error Name	**Error Value**	**Description**
BT1553_INT_HIGH_WORD	0x00000001	high word error
BT1553_INT_BIT_COUNT_DATA	0x00000001	bit count err, data word
BT1553_INT_INVALID_WORD	0x00000002	Invalid word error
BT1553_INT_LOW_WORD	0x00000004	low word error
BT1553_INT_INVERTED_SYNC	0x00000008	Inverted sync
BT1553_INT_MID_BIT	0x00000010	Mid Bit Error
BT1553_INT_TWO_BUS	0x00000020	data on both buses error
BT1553_INT_PARITY	0x00000040	parity error
BT1553_INT_NON_CONT_DATA	0x00000080	non-contiguous data
BT1553_INT_EARLY_RESP	0x00000100	early response
BT1553_INT_LATE_RESP	0x00000200	late response
BT1553_INT_BAD_RTADDR	0x00000400	incorrect rt address
BT1553_INT_CHANNEL	0x00000800	Bus (0=A, 1=B)
BT1553_INT_WRONG_BUS	0x00002000	Response on wrong bus
BT1553_INT_BIT_COUNT	0x00004000	bit count error
BT1553_INT_NO_IMSG_GAP	0x00008000	No/Short inter-message gap
BT1553_INT_END_OF_MESS	0x00010000	End of message
BT1553_INT_BROADCAST	0x00020000	broadcast message
BT1553_INT_RT_RT_FORMAT	0x00040000	rt-to-rt message format
BT1553_INT_RESET_RT	0x00080000	Reset rt
BT1553_INT_SELF_TEST	0x00100000	Self-test
BT1553_INT_MODE_CODE	0x00200000	Message is a Mode Code
BT1553_INT_NOCMD	0x00400000	Command unseen by decoder
BT1553_INV_RTRT_TX	0x00800000	Invalid RTRT TX CMD2
BT1553_INT_RTRT_RCV_NRSP	0x01000000	RT-RT No response on Rcv
BT1553_INT_RETRY	0x02000000	Retry
BT1553_INT_NO_RESP	0x04000000	no response (RT-RT, set if EITHER is no resp.)
BT1553_INT_ME_BIT	0x08000000	1553 status word message error bit
BT1553_INT_TRIG_BEGIN	0x10000000	message with trigger begin
BT1553_INT_TRIG_END	0x20000000	message with trigger end
BT1553_INT_BM_OVERFLOW	0x40000000	message at buffer overflow
BT1553_INT_ALT_BUS	0x80000000	retry on alternate bus

Table 4.1: Abaco BusTool Errors [9]

refers to these errors as interrupts and further details on each error/interrupt can be see in [9]. These errors are the result of the errors BusTools interprets on the databus and were separated into individual error flags. The complete list of 155 primary features created after data munging can be seen in Appendix B.

The feature generation techniques were then applied to the munged datasets. Polynomial expansion utilizing the *sklearn.preprocessing.PolynomialFeatures* tool created a total of 12,247 derived features from the existing 155 primary features. tsfresh created an additional 26,071 derived features from the existing 155 primary features through the use of the calculation modules outline in Appendix A. To note, some of the tsfresh calculation modules utilizes more than a single variable, so the number of features generated is not simply the number of modules multiplied by the primary features. Additionally, tsfresh will automatically eliminate generated features that contain no additional information, such as no variance in the values generated. Adding the features from the two generation techniques to the original primary feature set resulted in a total of of 38,473 unique features.

Using the process outlined Section 3.2.1, all of the attack datasets were labelled and specific discussion regarding this process is found in Section 4.3.5.

4.1.2 Models

After creation of the extended datasets, the feature selection techniques were implemented to determine the features to be used for each model. Model development consisted of two approaches; attack specific and general. The first approach was to use a specific model for each attack type as shown in Fig. 4.1. Each labelled attack type dataset was fed into the three feature selection technique tools, outlined in Fig. 4.1. The top features output from each feature selection were then used to define the features selected from the baseline dataset to train the models. The process for selecting the top features utilized elbow curves and is further discussed in Section 4.3.7. Upon completion, there was a specific model for each attack type and each feature selection method, resulting in 12 specific models.

The second approach was to create a general model by using the three feature selection techniques to select the top features for each attack type. These top features for each attack type were then combined and used as the features from the baseline to train the general model, as outlined in Fig. 4.2. This resulted in a total of 3 general models, one for each feature selection method. The purpose of the general model was to compare the results of a general model to the specific model to evaluate the ability for a more stream-

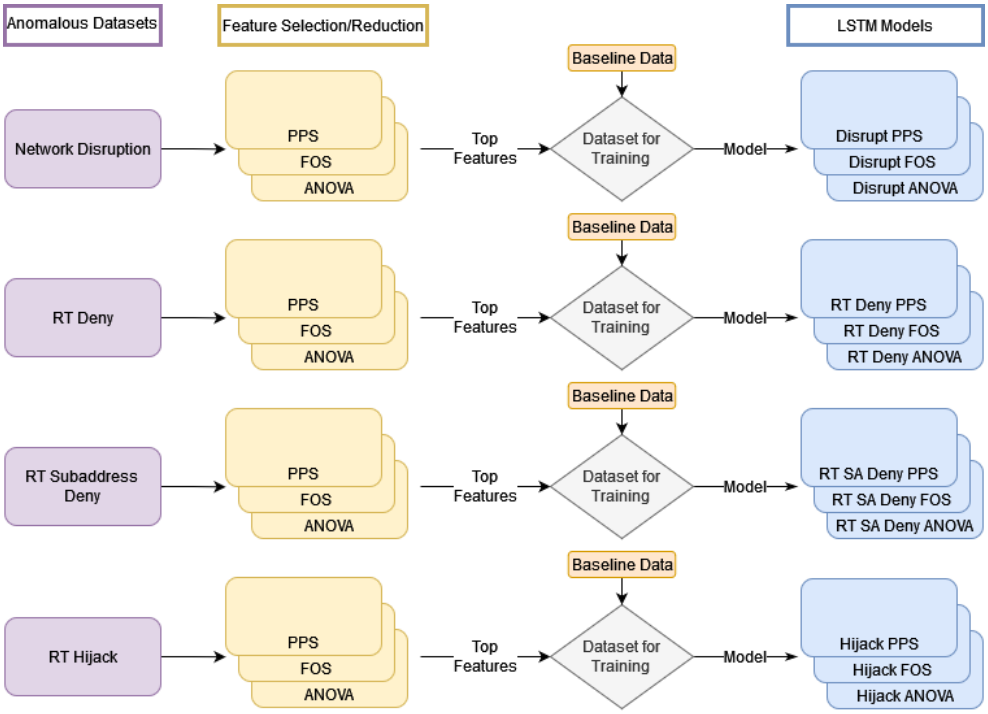

Figure 4.1: Attack Specific LSTM Model

lined approach for feature selection and model generation. The number of top features for all models was determined through the analysis of elbow curves shown in Appendix C. These elbow curves are discussed further in Section 4.3.7. Across the 15 models, there were approximately 5 to 15 features selected for each method.

Lastly, a single model was created for the feature reduction technique UMAP. UMAP was utilized in parallel with the feature selection techniques, although UMAP was only used with the primary feature dataset and not the extended dataset. The rational to this decision is explained further in Section 4.3.4.

Harlow's model [3] is utilized as a basis for this research. Next the threshold for anomalous traffic for all models was calculated based on the MAE. This reconstruction error threshold was based on Harlow's implementation. The reconstruction threshold was calculated as 10% above the MAE loss obtained during the training of the model with the baseline dataset. At the end of model creation there was a total of 16 unique models trained to be evaluated.

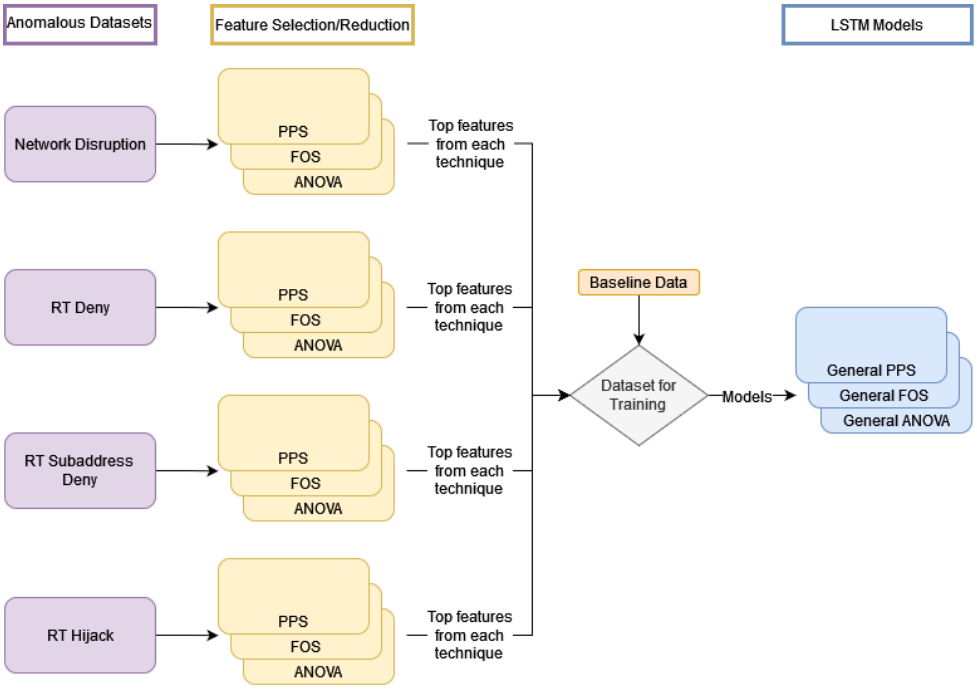

Figure 4.2: General LSTM Model

Using the process outlined in Section 3.2, all of the attack datasets were labelled. The maliciously labelled messages were compared against the start and stop timings of the attacks that were provided by Harlow [3]. Further discussion regarding labelling is found in Section 4.3.5.

4.2 Verification

The components utilized in the deep learning pipeline consisted of data munging to obtain a base set of features, feature generation to provide unique feature sets, feature selection and dimensionality reduction, and model generation. The developed pipeline was executed end-to-end to ingest the recorded MIL-STD-1553B traffic, process the data to create the primary feature set and then generate features using polynomial expansion and tsfresh techniques. Next, features to be used in training and testing each of the sixteen models were identified through three feature selection, and one feature reduction technique. The training and testing was then conducted, creating output used for analysis

with the MSE anomaly threshold. Each component of the pipeline operated as expected and based on the design outlined in Chapter 3.

4.3 Validation

The resulting performance metrics of the 16 models were then compared to the research conducted by Harlow [3] and demonstrated that there was an overall improvement in the effectiveness of the anomaly detection pipeline. These metrics included accuracy, precision, recall, AUROCC and MCC. To obtain the evaluation metrics, each anomalous dataset was processed through the respective LSTM autoencoder, which was trained on the baseline traffic. Scores were recorded for each anomalous dataset and their respective models, which are shown in Fig. 4.3. The green highlighted cells represent the highest value technique within each block, and the red text represents the highest value for each anomalous dataset, for both the general and specific models. Fig. 4.3 shows a resolution of four decimal places since that was the lowest resolution that was required to differentiate between the performance metrics of some models. The overall improvement through the addition of feature engineering was apparent in all models except for the general hijack model. All other models demonstrated an increase in performance metrics, with examples such as SA Deny increasing from 54% to 96% AUROCC and Disrupt increasing from 6% to 99% AUROCC.

4.3.1 ANOVA

The ANOVA general model outperformed all other general models in the DoS type attack (disrupt, deny, SA deny, SA deny 2). The specific ANOVA models: disrupt and hijack outperformed all other specific models for these same attack types, as seen in Fig. 4.3. The deny and disrupt specific models performed marginally better than the general model, although in the case of the SA deny dataset, the general model performed better by 4%. Furthermore, the deny, SA deny, SA deny 2, and disrupt datasets utilizing the general model obtained similar results. This is attributed to the four datasets utilizing the DoS attack method, and as such would contain similar traffic. Due to this result, it shows that these four attacks could be combined into a single DoS model, allowing a more streamlined pipeline while still yielding effective results when compared to Harlow's model [3]. In the case of the hijack dataset metrics, the specific hijack model significantly outperformed the general hijack model.

General Models

Deny

	original	anova	pps	fos	umap
Accuracy	0.9968	0.9992	0.9795	0.9763	0.9938
Precision	0.8606	0.9977	0.8231	0.4937	0.4969
Recall	0.8981	0.9825	0.5005	0.4990	0.5000
AUROCC	0.8981	0.9825	0.5005	0.4990	0.5000
MCC	0.7577	0.9800	0.0248	-0.0050	0.0000

SA Deny

	original	anova	pps	fos	umap
Accuracy	0.9913	0.9961	0.9465	0.9442	1.0000
Precision	0.5001	0.9963	0.8615	0.4771	0.5000
Recall	0.5433	0.9655	0.5006	0.4990	0.5000
AUROCC	0.5433	0.9655	0.5006	0.4990	0.5000
MCC	0.0042	0.9613	0.0290	-0.0098	0.0000

SA Deny 2

	original	anova	pps	fos	umap
Accuracy	0.9930	0.9959	0.9598	0.9580	1.0000
Precision	0.5000	0.9964	0.7656	0.4844	0.5000
Recall	0.4965	0.9506	0.5001	0.4993	0.5000
AUROCC	0.4965	0.9506	0.5001	0.4993	0.5000
MCC	-0.0004	0.9459	0.0100	-0.0068	0.0000

Disrupt

	original	anova	pps	fos	umap
Accuracy	0.7918	0.9927	0.8538	0.8387	0.8104
Precision	0.5545	0.9954	0.9096	0.4495	0.4052
Recall	0.5153	0.9755	0.5029	0.4938	0.5000
AUROCC	0.5153	0.9755	0.5029	0.4938	0.5000
MCC	0.0577	0.9707	0.0689	-0.0355	0.0000

Hijack

	original	anova	pps	fos	umap
Accuracy	0.9833	0.9710	0.9957	0.9951	0.9957
Precision	0.5769	0.4979	0.4979	0.6861	0.4979
Recall	0.8212	0.4880	0.5000	0.6164	0.5000
AUROCC	0.8212	0.4880	0.5000	0.6164	0.5000
MCC	0.3143	-0.0100	-0.0002	0.2944	0.0000

Specific Models

Deny

	original	anova	pps	fos	umap
Accuracy	0.9968	0.9996	0.9795	0.9999	0.9938
Precision	0.8606	0.9978	0.7897	0.9981	0.4969
Recall	0.8981	0.9913	0.5001	0.9989	0.5000
AUROCC	0.8981	0.9913	0.5001	0.9989	0.5000
MCC	0.7577	0.9891	0.0129	0.9971	0.0000

SA Deny

	original	anova	pps	fos	umap
Accuracy	0.9913	0.9924	0.9465	0.9940	1.0000
Precision	0.5001	0.9946	0.9590	0.9955	0.5000
Recall	0.5433	0.9307	0.5003	0.9455	0.5000
AUROCC	0.5433	0.9307	0.5003	0.9455	0.5000
MCC	0.0042	0.9231	0.0236	0.9396	0.0000

SA Deny 2

	original	anova	pps	fos	umap
Accuracy	0.9930	0.9930	0.9598	0.9951	1.0000
Precision	0.5000	0.9950	0.9513	0.9962	0.5000
Recall	0.4965	0.9142	0.5004	0.9402	0.5000
AUROCC	0.4965	0.9142	0.5004	0.9402	0.5000
MCC	-0.0004	0.9056	0.0268	0.9347	0.0000

Disrupt

	original	anova	pps	fos	umap
Accuracy	0.7918	0.9966	0.8530	0.8532	0.8104
Precision	0.5545	0.9976	0.4265	0.7735	0.4052
Recall	0.5153	0.9889	0.5000	0.5013	0.5000
AUROCC	0.5153	0.9889	0.5000	0.5013	0.5000
MCC	0.0577	0.9864	0.0000	0.0376	0.0000

Hijack

	original	anova	pps	fos	umap
Accuracy	0.9833	0.9967	0.9954	0.9943	0.9957
Precision	0.5769	0.7984	0.5196	0.5246	0.4979
Recall	0.8212	0.8262	0.5017	0.5093	0.5000
AUROCC	0.8212	0.8262	0.5017	0.5093	0.5000
MCC	0.3143	0.6240	0.0115	0.0303	0.0000

Figure 4.3: Results for general and specific models.

4.3.2 FOS

FOS performed well for the specific models: Deny and SA Deny, and performed relatively poorly for the disrupt and hijack models. FOS also performed poorly for all of the general models as seen in Fig. 4.3. Additionally, the remainder of the FOS model's metrics demonstrated poor performance. The feature scores from FOS resulted in an elbow curve that typically suggested only one feature, whereas the majority of other feature selection methods suggested 5 or more. Due to the elbow method only suggesting a single feature, it is suggested that other methods be explored for selecting the cutoff for the number of features. Except for precision, the metrics from the disrupt

dataset demonstrated similar performance between the general and specific models, although with poor performance. Finally, with the metrics from the hijack dataset, the general model outperformed the specific model, although again with overall poor performance. These results highlight the possible need for more anomalous traffic to be used for selection of the features.

4.3.3 PPS

PPS performance metrics initially appear to be ineffective across the majority of the datasets as seen in Fig. 4.3. Although on closer analysis of the metrics from the deny, SA deny and SA deny 2 datasets, those specific models shows promise. This can be demonstrated in Fig. 4.4, where the anomalous traffic can be seen in three distinct groupings where the MAE value is much larger relative to the other traffic. These three groupings of increased MAE represent the anomalous traffic, and are are similar to the ANOVA and FOS MAE values for the same dataset, as seen in Appendix E Fig. E.3, Fig. E.4 and Fig. E.5. Therefore, the MAE threshold for anomalous traffic in Harlow's model [3] could be adjusted to potentially provide more effective detection for the PPS model. This could be accomplished by reducing the MAE reconstruction threshold or utilizing other methods such as standard deviation. This reduction of the reconstruction threshold would then result with more of the traffic seen in Fig. 4.4 surpassing the threshold and thus resulting in detection by the model. The specific disrupt model demonstrated poor performance according to the PPS evaluation metrics. Additionally, the four general PPS models demonstrated poor performance especially in comparison to the top performing ANOVA model.

4.3.4 UMAP

The UMAP model did not perform well when compared to all other models. UMAP was the dimensionality reduction method selected and as such, it reduced all the primary features to the the selected number of features. The tuning for UMAP consisted of a trial and error method in adjusting the hyperparameters to allow proper clustering of data. After testing multiple iterations of hyperparameters and visualising the results, the following hyperparameters were chosen:

- N Neighbors: 10
- Min Distance: 0.1
- N Components: 10

It was decided to limit any further time invested in tuning UMAP and noted that future investigation would be required. Additionally, UMAP was used

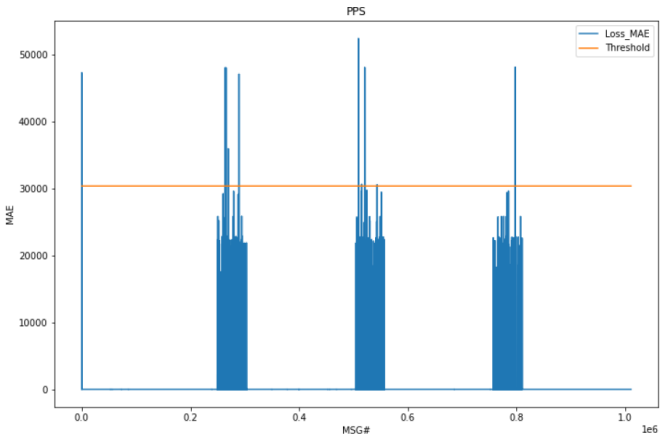

Figure 4.4: MSE for PPS feature selection Method on the Deny dataset

solely on the primary features. This decision was taken solely due to the equipment limitation, specifically because reducing the expanded datasets required substantially more RAM than was available. In order to properly investigate the use of UMAP on the extended datasets, additional RAM would be required as well as utilization of GPU resources to allow for reduced compute time. In the case of the current datasets selected through the feature selection methods in this research, the need for further feature reduction is seen as unnecessary since the selected number of features from all the methods were sufficiently small.

4.3.5 Hijack Dataset

As seen in Fig. 4.3, the hijack models received the lowest MCC and AUROCC score. This is possible for two reasons: data labelling and attack complexity. This dataset has a higher probability of being mis-labelled as the labelling method described did not account for message collisions and how Abaco BusTools-1553 may create a message with assumed data. This was accounted for in the other datasets as it was the sole method for labelling, whereas in the hijack dataset it becomes more complex because there is both data being replaced as well as the possibility for collisions taking place. Additionally, the errors could be attributed to the complexity of the attack, where the model may see similar traffic with only minor differences in the data and timing. In order to further support this reasoning, a pre-labelled dataset would need to be used in place of labelling the data after recording.

4.3.6 Feature Generation

Feature generation consisted of using *sklearn.preprocessing.PolynomialFeatures* and tsfresh packages. In Fig. 4.5, all features consisting of all capitalization are either primary features or features generated through sklearn Polynomial Features. Furthermore, the features generated through sklearn Polynomial Features are labelled how they were created, such as *RSP1 CMD1-addr* is primary features *RSP1* $*$ *CMD1-addr*. The remaining features were created by the tsfresh tool. The majority of the features in the higher performing models were the result of the sklearn Polynomial Features package. Additionally, the features generated by the tsfresh tool did not perform as well as expected despite being used in previous research completed by Stan et al[6].

4.3.7 Feature Selection and Reduction

The top 10 features for each attack method and dataset can be seen in Fig. 4.5, where the green highlighted fields show common features between at least two feature selection methods and red font represents common features between at least two datasets within the same feature selection method. Fig. 4.5 does not represent the actual number of features in each model, and is used to illustrate the similar features across the feature selection methods and models. It is noted in Fig. 4.5 that there are very few, to no common features between the feature selection methods utilizing the Net Disrupt and Hijack dataset. Additionally, as seen in Fig. 4.5 there are few to no features shared between the hijack dataset and the deny, SA deny and disrupt datasets. This further re-enforces the similarities between the DoS attacks and the differences from the hijack attacks suggesting separate model for detection of these respective attack types.

The feature selection and reduction methods were conducted to determine the top performing features for each of the four datasets (disrupt, deny, SA deny, and hijack). These results were analyzed, and it was noted where there was a pronounced drop-off in the relative feature selection scores. As seen in Appendix C, the disrupt, deny and SA deny FOS results had a sharp dropoff after only one feature, whereas in the hijack results produced a curve that was less pronounced and suggested approximately 4 features. In the case of the ANOVA method, again the disrupt, deny and SA deny results had similar curves suggesting approximately 7-10 features, whereas the hjiack results did not have a defined elbow. Finally, the PPS method showed similar curves between the deny and SA deny results, no elbow for the hijack dataset and suggesting only one feature for disrupt. These results can be split into three

NetDisrupt_statusword_250820_top_10

ANOVA	FOS	PPS
CMD1-addr BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_TWO_BUS_CMD_1 BT1553_INT_INVALID_WORD_CMD_1	BT1553_INT_NO_IMSG_GAP_CMD_1 BT1553_INT_BIT_COUNT_CMD_1	24_length
BT1553_INT_INVALID_WORD_CMD_1	19_agg_linear_trend__attr_"stderr"__chunk_len_10__f_agg_"mean"	18_sum_values
BT1553_INT_INVALID_WORD_CMD_1^2	DATA10 DATA19	18_mean
CMD1-addr BT1553_INT_MID_BIT_CMD_1	INTERMESSAGE_GAP BT1553_INT_MID_BIT_STS_1	18_length
CMD1-subaddr BT1553_INT_INVALID_WORD_CMD_1	DATA10 DATA16	19_sum_values
28_large_standard_deviation__r_0.2	INTERMESSAGE_GAP BT1553_INT_INVALID_WORD_STS_1	19_length
4_large_standard_deviation__r_0.30000000000000004	DATA10 DATA15	19_root_mean_square
4_large_standard_deviation__r_0.35000000000000003	DATA09 DATA14	20_sum_values
4_large_standard_deviation__r_0.4	INTERMESSAGE_GAP BT1553_INT_NO_IMSG_GAP_STS_1	20_length

RT-SAdeny_statusword_rt18_sa1_250820_top_10

ANOVA	FOS	PPS
BT1553_INT_INVALID_WORD_CMD_1	BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_INVALID_WORD_CMD_1^2	BT1553_INT_BIT_COUNT_CMD_1	INTERMESSAGE_GAP BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_MID_BIT_CMD_1	BT1553_INT_MID_BIT_CMD_1	BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_MID_BIT_CMD_1^2	10_agg_linear_trend__attr_"stderr"__chunk_len_50__f_agg_"var"	BT1553_INT_INVALID_WORD_CMD_1^2
BT1553_INT_MID_BIT_CMD_1 BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_MID_BIT_CMD_1	RSP1 CMD1-addr
CMD1-addr BT1553_INT_INVALID_WORD_CMD_1	CMD1-subaddr BT1553_INT_BIT_COUNT_CMD_1	20_length
BT1553_INT_LOW_WORD_CMD_1 BT1553_INT_INVALID_WORD_CMD_1	31_median	23_sum_values
BT1553_INT_LOW_WORD_CMD_1^2	24_maximum	22_length
BT1553_INT_LOW_WORD_CMD_1	31_agg_linear_trend__attr_"stderr"__chunk_len_50__f_agg_"max"	22_sum_values
CMD1-addr BT1553_INT_MID_BIT_CMD_1	31_agg_linear_trend__attr_"slope"__chunk_len_50__f_agg_"var"	21_length

RT-SAdeny_statusword_rt18_sa32_250820_top_10

ANOVA	FOS	PPS
CMD1-addr BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_INVALID_WORD_CMD_1
CMD1-addr BT1553_INT_MID_BIT_CMD_1	BT1553_INT_BIT_COUNT_CMD_1	INTERMESSAGE_GAP BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_MID_BIT_CMD_1^2	CMD1-addr BT1553_INT_MID_BIT_CMD_1	CMD1-numword BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_MID_BIT_CMD_1	27_cwt_coefficients__coeff_13__w_2__widths_(2, 5, 10, 20)	BT1553_INT_MID_BIT_CMD_1
BT1553_INT_MID_BIT_CMD_1 BT1553_INT_INVALID_WORD_CMD_1	CMD1-addr BT1553_INT_BIT_COUNT_CMD_1	BT1553_INT_MID_BIT_CMD_1^2
BT1553_INT_INVALID_WORD_CMD_1^2	31_large_standard_deviation__r_0.25	BT1553_INT_MID_BIT_CMD_1 BT1553_INT_INVALID_WORD_CMD_1
BT1553_INT_INVALID_WORD_CMD_1	9_maximum	INTERMESSAGE_GAP BT1553_INT_MID_BIT_CMD_1
CMD1-addr BT1553_INT_LOW_WORD_CMD_1	25_change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.4	CMD1-addr BT1553_INT_MID_BIT_CMD_1
CMD1-numword BT1553_INT_INVALID_WORD_CMD_1	CMD1-numword BT1553_INT_PARITY_CMD_1	CMD1-subaddr BT1553_INT_MID_BIT_CMD_1
CMD1-subaddr BT1553_INT_MID_BIT_CMD_1	CMD1-TR BT1553_INT_INVALID_WORD_CMD_1	20_length

Hijack_rt18_sa6_w56_250820_top_10

ANOVA	FOS	PPS
CMD1-TR STS1-Reserved	CMD1-TR STS1-Reserved	RSP1 CMD1-numword
CMD1-TR STS1-SerReq	STS1-Reserved DATA14	CMD1-numword BT1553_INT_BIT_COUNT_STS_1
CMD1-TR STS1-Error	17_maximum	INTERMESSAGE_GAP CMD1-TR
CMD1-TR STS1-BCRecv	19_median	INTERMESSAGE_GAP CMD1-subaddr
CMD1-TR STS1-Inst	28_ar_coefficient__coeff_10__k_10	CMD1-numword BT1553_INT_BAD_RTADDR_STS_1
CMD1-TR STS1-Busy	31_median	CMD1-subaddr STS1-SerReq
STS1-BCRecv BT1553_INT_BIT_COUNT_STS_1	31_change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.0	CMD1-numword STS1-Error
STS1-Error BT1553_INT_BIT_COUNT_STS_1	28_agg_linear_trend__attr_"slope"__chunk_len_50__f_agg_"max"	CMD1-numword STS1-Inst
STS1-Reserved BT1553_INT_BIT_COUNT_STS_1	14_maximum	CMD1-numword STS1-SerReq
STS1-Reserved BT1553_INT_BAD_RTADDR_STS_1	32_maximum	CMD1-numword STS1-BCRecv

Figure 4.5: Top 10 features selected for each attack method and feature selection method

groups: feature scores with a defined elbow curve, feature scores with no defined elbow and features with a sharp drop after the first feature. In the case of the defined elbow group, the number of features utilized in the model corresponds to the elbow curve. In the case of the no elbow curve group, it was decided to use 10 features as 10 features seems to be the common number of features among the other datasets. Finally, in the case of a single feature, it was determined to utilize 5 feature as a single feature could not provide enough data to the models for proper training.

4.3.8 Datasets

There were a total of five datasets with attack traffic present recorded. Additionally each of those five datasets contained three distinct attacks, all lasting approximately one minute in length. This results in a total of 3 minutes of attack traffic within a total of 20 minutes of recorded traffic. Though this recorded traffic has a low number of distinct attacks, is of a short duration, and reflects limited flight activity, it does provide sufficient data for this experiment.

4.3.9 Comparison

The results of the improved anomaly detection pipeline were evaluated against the previous research completed by Harlow[3]. Comparing to the original features utilized by Harlow[3] the refined anomaly detection pipeline had a marked improvement. Overall, the metrics for anomaly detection increased significantly. Additionally, the amount of features required to provide accurate predictions was reduced from 62 to less than 30 features, with some of the higher performing models requiring as low as 5 features. Finally, the increase in MAE between anomalous traffic and normal traffic demonstrates a much larger reconstruction error for the anomalous traffic. This allows for easier anomalous traffic detection when compared to Harlow's [3], as seen in Appendix D and Appendix E.

4.4 Summary

In this chapter, the results from the feature engineering pipeline were presented and discussed. The detailed data munging, feature engineering and model creation process were explained. The metrics from the feature engineering methods were then presented and discussed in detail, identifying trends and points of interest. Finally the results were compared to current research

completed by Harlow [3], identifying the benefits of the revised feature engineering pipeline for the MIL-STD-1553B anomaly detector. The refinements to the feature engineering component in the existing deep learning pipeline demonstrated a marked improvement. Specifically, half of the generated models showed increased performance compared to the original model. In addition, this performance improvement was achieved using significantly fewer features. In combination, these previous two points highlight a pipeline that involves lower processing times for feature engineering and model execution accompanied by improved effectiveness of the original LSTM anomaly detector.

5 Conclusion

This research refined the feature engineering component of an existing MIL-STD-1553B deep learning anomaly detection pipeline. This was accomplished through the use of different feature generation, feature selection and feature reduction techniques in order to improve the overall effectiveness of the anomaly detector. In this final chapter an overview is presented, reiterating the motivation for this research. Next this research's contributions of this research to the field of MIL-STD-1553B anomaly detection will be presented. Recommendations for future work based on this research will then be discussed. Finally, recommendations will be offered for actions based on the findings in this research.

5.1 Overview

Research has demonstrated the vulnerabilities of the MIL-STD-1553B databus and the ability for adversaries to exploit these vulnerabilities [1], [2]. In order to mitigate the impact of these vulnerabilities being exploited, the ability to detect malicious traffic on the MIL-STD-1553B databus is paramount. Recent research has allowed the monitoring of the MIL-STD-1553B databus and the ability to effectively detect anomalous traffic. These methods have included machine learning, deep learning, statistical and signature-based approaches [2], [3], [4], [5].

The research conducted by Harlow [3] utilized a deep learning model and was successful in demonstrating an initial proof of concept for anomalous traffic on the MIL-STD-1553B databus. This approach was further developed by refining the feature engineering portion of the deep learning pipeline. The specific methods implemented to refine the pipeline included feature generation and feature selection. This refined portion of the pipeline was then used with the existing model created by Harlow [3] to determine if there was an overall increase in the ability to detect anomalous databus traffic.

The refined deep learning anomaly detection pipeline was then verified through its ability to correctly categorize the anomalous datasets. The refined pipeline generated new features using polynomial expansion and with tsfresh that in turned served as input into the three different feature selection techniques and the dimensionality reduction technique. Sixteen distinct models based on Harlow's original design were created and processed with the new feature engineering aspects created through this research and results were calculated using common performance metrics. These metrics are based off the confusion matrix and include; accuracy, precision, recall, AUROCC and MCC. These metrics showed that the refined deep learning pipeline overall was effective in detecting anomalous traffic in the MIL-STD-1553B databus.

The results from the experiments were then validated by comparing the performance metrics against the research completed by Harlow [3] and clearly demonstrate that there was a marked improvement in anomalous traffic detection. The comparison of the results revealed there was a marked improvement in 8 of the 16 models created during this research. In addition, this research demonstrated that the improved results can be achieved with fewer features. The reduction in the required features results in lower processing times on similar hardware, or a reduction in the hardware resources required for accurate detection of anomalous traffic on the MIL-STD-1553B databus.

5.2 Contributions

The following contributions were made by this research:

1. This research showed that feature engineering techniques can achieve improvements in detection metrics while at the same time, require fewer features for processing new data.
2. A developed methodology for feature engineering using traffic collected from a MIL-STD-1553B data bus.
3. Refined feature engineering techniques for anomaly detection using a LSTM autoencoder model for MIL-STD-1553B data bus traffic.
4. An improved pipeline for anomaly detection on MIL-STD-1553B data bus traffic.
5. Determining the similarities between the DoS type attacks and the differences from the hijack attacks, identifying that separate models are suggested for the detection of these attack types.
6. Identifying of the similarities between the DoS type attacks, allowing for more streamlined model implementation.

5.3 Future Work

This research investigated the effectiveness of utilizing feature engineering in the deep learning pipeline for anomaly detection in MIL-STD-1553B traffic. This work has also highlighted other areas that are recommended to investigate to further this work:

1. Development of richer benign and malicious traffic data sets with a more complex master schedule, more RTs and aperiodic messages. These new datasets would allow for the creation of more robust training, test and validation sets providing better validation of the created model(s).
2. Development of a MIL-STD-1553B attack framework which allows for labelling of data at creation time. These datasets would allow for higher fidelity metrics.
3. Development of a model detection that detects the mode of the aircraft, allowing models to be created for both attack types and specific aircraft modes.
4. Further experimentation focused on optimization of dimensionality techniques such as UMAP that may further improve performance.

5.4 Recommendations

The use of MIL-STD-1553B monitoring solutions is limited within the Royal Canadian Air Force. This research recommends the evaluation of an anomaly detection solution for use on real MIL-STD-1553B traffic. If such a solution was tested and provided similar results it would allow for effective detection of anomalous traffic on the MIL-STD-1553B databus. This in turn would provide the ability to monitor traffic on these databuses to provide additional security to help close the gap in protection due to the fact MIL-STD-1553B does not have security built into the protocol.

References

[1] J. R. Paquet, "Uncovering MIL-STD-1553 Vulnerabilities: Exploiting Military Aircraft Networks," Master's thesis, Royal Military College of Canada, 2014.

[2] O. Stan, A. Cohen, Y. Elovici, and A. Shabtai, "On the Security of MIL-STD-1553 Communication Bus," in *Security and Safety Interplay of Intelligent Software Systems*, ser. Lecture Notes in Computer Science, B. Hamid, B. Gallina, A. Shabtai, Y. Elovici, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2019, pp. 153–171.

[3] A. Harlow, "Defending MIL-STD-1553 Vulnerabilites: Intrusion Detection on Airborne Platform Networks," Master's thesis, Royal Military College of Canada, 2021, [Draft].

[4] C. Bernard, "An Application of Network Security Monitoring to the MIL-STD-1553B Data Bus," Master's thesis, Royal Military College of Canada, Sep. 2019.

[5] 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, no. 1, pp. 276–284, Feb. 2020.

[6] 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, no. 4, pp. 3010–3027, Aug. 2020.

[7] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv:1802.03426 [cs, stat]*, Sep. 2020, arXiv: 1802.03426. [Online]. Available: http://arxiv.org/abs/1802.03426

[8] "MILSTD1553.com | Complete Online Reference for MIL-STD-1553." [Online]. Available: https://www.milstd1553.com/

[9] "BusTools-1553 BusTools Software Analyzer," Aug. 2006. [Online]. Available: https://www.abaco.com/products/bt-1553-bustools-software-analyzer

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Dec. 2019. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

[11] O. Stan, Y. Elovici, A. Shabtai, G. Shugol, R. Tikochinski, and S. Kur, "Protecting Military Avionics Platforms from Attacks on MIL-STD-1553 Communication Bus," *arXiv:1707.05032 [cs]*, Jul. 2017, arXiv: 1707.05032. [Online]. Available: http://arxiv.org/abs/1707.05032

[12] B. F. Losier, R. Smith, and V. Roberge, "Design of a Time-Based Intrusion Detection Algorithm for the MIL-STD-1553," Jan. 2019. [Online]. Available: http://roberge.segfaults.net/joomla/files/publications/Project_Report_2102.pdf

[13] 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)*, Dec. 2020, pp. 592–598, iSSN: 2324-9013.

[14] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2016, pp. 130–139.

[15] M. Elgendy, *Deep Learning for Vision Systems*. Manning, Oct. 2020. [Online]. Available: https://www.manning.com/books/deep-learning-for-vision-systems

[16] D. McGaughey, T. Semeniuk, R. Smith, and S. Knight, "A systematic approach of feature selection for encrypted network traffic classification," in *2018 Annual IEEE International Systems Conference (SysCon)*, Apr. 2018, pp. 1–8, iSSN: 2472-9647.

[17] S. Zander and N. Williams, "netAI - Network Traffic based Application Identification." [Online]. Available: http://caia.swin.edu.au/urp/dstc/netai/

[18] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh –

A Python package)," *Neurocomputing*, vol. 307, pp. 72–77, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218304843

[19] "tsfresh." [Online]. Available: https://tsfresh.readthedocs.io/en/latest/

[20] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery*, 1st ed., 2020. [Online]. Available: https://machinelearningmastery.com/data-preparation-for-machine-learning/#packages

[21] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003.

[22] R. Muthukrishnan and R. Rohini, "LASSO: A feature selection technique in predictive modeling for machine learning," in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, Oct. 2016, pp. 18–20.

[23] F. Wetschoreck, "RIP correlation. Introducing the Predictive Power Score," May 2020. [Online]. Available: https://towardsdatascience.com/rip-correlation-introducing-the-predictive-power-score-3d90808b9598

[24] K. Demertzis, K. Tsiknas, D. Takezis, C. Skianis, and L. Iliadis, "Darknet Traffic Big-Data Analysis and Network Management to Real-Time Automating the Malicious Intent Detection Process by a Weight Agnostic Neural Networks Framework," *arXiv:2102.08411 [cs]*, Feb. 2021, arXiv: 2102.08411. [Online]. Available: http://arxiv.org/abs/2102.08411

[25] N. Levi, N. Hillel, E. Zaady, G. Rotem, Y. Ziv, A. Karnieli, and T. Paz Kagan, "Soil quality index for assessing phosphate mining restoration in a hyper-arid environment," *Ecological Indicators*, vol. 125, p. 107571, Mar. 2021.

[26] R. B. Adhao and V. K. Pachghare, "Performance-Based Feature Selection Using Decision Tree," in *2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET)*, Dec. 2019, pp. 135–138.

[27] "scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

[28] M. J. Korenberg and L. D. Paarmann, "Applications of fast orthogonal search: time-series analysis and resolution of signals in noise," *Annals of Biomedical Engineering*, vol. 17, no. 3, pp. 219–231, 1989.

[29] R. E. Bellman, *Adaptive Control Processes.* NJ: Princeton University Press, 1961.

[30] K. P. Murphy, *Machine learning: a probabilistic perspective*, ser. Adaptive computation and machine learning series. Cambridge, MA: MIT Press, 2012.

[31] "Understanding UMAP." [Online]. Available: https://pair-code.github.io/understanding-umap/

[32] M. Ali, M. W. Jones, X. Xie, and M. Williams, "TimeCluster: dimension reduction applied to temporal data for visual analytics," *The Visual Computer*, vol. 35, no. 6, pp. 1013–1026, Jun. 2019. [Online]. Available: https://doi.org/10.1007/s00371-019-01673-y

[33] C. Chio and D. Freeman, *Machine Learning and Security.* O'Reilly, 2018.

[34] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 385. [Online]. Available: http://link.springer.com/10.1007/978-3-642-24797-2

[35] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[36] A. Kundu, A. Sahu, E. Serpedin, and K. Davis, "A3D: Attention-based auto-encoder anomaly detector for false data injection attacks," *Electric Power Systems Research*, vol. 189, p. 106795, Dec. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378779620305988

[37] O. I. Provotar, Y. M. Linder, and M. M. Veres, "Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders," in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, Dec. 2019, pp. 513–517.

[38] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, Jan. 2020. [Online]. Available: https://doi.org/10.1186/s12864-019-6413-7

Appendices

A tsfresh Calculation Modules

Module		Comment
1	abs energy (x)	Returns the absolute energy of the time series which is the sum over the squared values.
2	absolute maximum (x)	Calculates the highest absolute value of the time series x.
3	absolute sum of changes (x)	Returns the sum over the absolute value of consecutive changes in the series x
4	agg autocorrelation (x, param)	Descriptive statistics on the autocorrelation of the time series.
5	agg linear trend (x, param)	Calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one.
6	approximate entropy (x, m, r)	Implements a vectorized Approximate entropy algorithm.
7	ar coefficient (x, param)	This feature calculator fits the unconditional maximum likelihood of an autoregressive AR (k) process.
8	augmented dickey fuller (x, param)	Does the time series have a unit root?
9	autocorrelation (x, lag)	Calculates the autocorrelation of the specified lag, according to the formula [1]
10	benford correlation (x)	Useful for anomaly detection applications [1][2].
continued on next page		

Module		Comment
11	binned entropy (x, max bins)	First bins the values of x into max bins equidistant bins.
12	c3 (x, lag)	Uses c3 statistics to measure non linearity in the time series
13	change quantiles (x, ql, qh, isabs, f agg)	First fixes a corridor given by the quantiles ql and qh of the distribution of x.
14	cid ce (x, normalize)	This function calculator is an estimate for a time series complexity [1] (A more complex time series has more peaks, valleys etc.).
15	count above (x, t)	Returns the percentage of values in x that are higher than t
16	count above mean (x)	Returns the number of values in x that are higher than the mean of x
17	count below (x, t)	Returns the percentage of values in x that are lower than t
18	count below mean (x)	Returns the number of values in x that are lower than the mean of x
19	cwt coefficients (x, param)	Calculates a Continuous wavelet transform for the Ricker wavelet, also known as the “Mexican hat wavelet” which is
20	energy ratio by chunks (x, param)	Calculates the sum of squares of chunk i out of N chunks expressed as a ratio with the sum of squares over the whole series.
21	fft aggregated (x, param)	Returns the spectral centroid (mean), variance, skew, and kurtosis of the absolute fourier transform spectrum.
22	fft coefficient (x, param)	Calculates the fourier coefficients of the one-dimensional discrete Fourier Transform for real input by fast
23	first location of maximum (x)	Returns the first location of the maximum value of x.
24	first location of minimum (x)	Returns the first location of the minimal value of x.
continued on next page		

Module		Comment
25	fourier entropy (x, bins)	Calculate the binned entropy of the power spectral density of the time series (using the welch method).
26	friedrich coefficients (x, param)	Coefficients of polynomial , which has been fitted to
27	has duplicate (x)	Checks if any value in x occurs more than once
28	has duplicate max (x)	Checks if the maximum value of x is observed more than once
29	has duplicate min (x)	Checks if the minimal value of x is observed more than once
30	index mass quantile (x, param)	Calculates the relative index i of time series x where q% of the mass of x lies left of i.
31	kurtosis (x)	Returns the kurtosis of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G2).
32	large standard deviation (x, r)	Does time series have large standard deviation?
33	last location of maximum (x)	Returns the relative last location of the maximum value of x.
34	last location of minimum (x)	Returns the last location of the minimal value of x.
35	lempel ziv complexity (x, bins)	Calculate a complexity estimate based on the Lempel-Ziv compression algorithm.
36	length (x)	Returns the length of x
37	linear trend (x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
38	linear trend timewise (x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
continued on next page		

Module		Comment
39	longest strike above mean (x)	Returns the length of the longest consecutive subsequence in x that is bigger than the mean of x
40	longest strike below mean (x)	Returns the length of the longest consecutive subsequence in x that is smaller than the mean of x
41	matrix profile (x, param)	Calculates the 1-D Matrix Profile[1] and returns Tukey's Five Number Set plus the mean of that Matrix Profile.
42	max langevin fixed point (x, r, m)	Largest fixed point of dynamics :math:argmax x {h (x)=0}‘ estimated from polynomial ,
43	maximum (x)	Calculates the highest value of the time series x.
44	mean (x)	Returns the mean of x
45	mean abs change (x)	Average over first differences.
46	Module	Comment
47	mean change (x)	Average over time series differences.
48	mean n absolute max (x, number of maxima)	Calculates the arithmetic mean of the n absolute maximum values of the time series.
49	mean second derivative central (x)	Returns the mean value of a central approximation of the second derivative
50	median (x)	Returns the median of x
51	minimum (x)	Calculates the lowest value of the time series x.
52	number crossing m (x, m)	Calculates the number of crossings of x on m.
53	number cwt peaks (x, n)	Number of different peaks in x.
54	number peaks (x, n)	Calculates the number of peaks of at least support n in the time series x.
55	partial autocorrelation (x, param)	Calculates the value of the partial autocorrelation function at the given lag.
56	percentage of reoccurring datapoints to all datapoints (x)	Returns the percentage of non-unique data points.
continued on next page		

Module		Comment
57	percentage of reoccurring values to all values (x)	Returns the percentage of values that are present in the time series more than once.
58	permutation entropy (x, tau, dimension)	Calculate the permutation entropy.
59	quantile (x, q)	Calculates the q quantile of x.
60	query similarity count (x, param)	This feature calculator accepts an input query subsequence parameter, compares the query (under z-normalized Euclidean distance) to all subsequences within the time series, and returns a count of the number of times the query was found in the time series (within some predefined maximum distance threshold).
61	range count (x, min, max)	Count observed values within the interval [min, max).
62	ratio beyond r sigma (x, r)	Ratio of values that are more than r * std (x) (so r times sigma) away from the mean of x.
63	ratio value number to time series length (x)	Returns a factor which is 1 if all values in the time series occur only once, and below one if this is not the case.
64	root mean square (x)	Returns the root mean square (rms) of the time series.
65	sample entropy (x)	Calculate and return sample entropy of x.
66	set property (key, value)	This method returns a decorator that sets the property key of the function to value
67	skewness (x)	Returns the sample skewness of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G1).
68	spkt welch density (x, param)	This feature calculator estimates the cross power spectral density of the time series x at different frequencies.
69	standard deviation (x)	Returns the standard deviation of x
70	sum of reoccurring data points (x)	Returns the sum of all data points, that are present in the time series more than once.
continued on next page		

Module		Comment
71	sum of reoccurring values (x)	Returns the sum of all values, that are present in the time series more than once.
72	sum values (x)	Calculates the sum over the time series values
73	symmetry looking (x, param)	Boolean variable denoting if the distribution of x looks symmetric.
74	time reversal asymmetry statistic (x, lag)	Returns the time reversal asymmetry statistic.
75	value count (x, value)	Count occurrences of value in time series x.
76	variance (x)	Returns the variance of x
77	variance larger than standard deviation (x)	Is variance higher than the standard deviation?
78	variation coefficient (x)	Returns the variation coefficient (standard error / mean, give relative value of variation around mean) of x.

B Primary Features

This Appendix lists all the primary features extracted from the raw data contained from the Abaco BusTool BMDX files. Further details on specific features and flags can be found in the Abaco BusTools API documentation [9].

To save space, these 155 features are listened in two columns across a single table the covers 3 pages. Additionally E/S refers to the DATA word directly above. For example E/S.1 is the Interrupt (error) Enable/Status bits for DATA02, as this is how Abaco BusTools records and labels the data.

RSP1	E/S.23
RSP2	DATA25
INTERMESSAGE GAP	E/S.24
CMD1-addr	DATA26
CMD1-TR	E/S.25
CMD1-subaddr	DATA27
CMD1-numword	E/S.26
CMD2-addr	DATA28
CMD2-TR	E/S.27
CMD2-subaddr	DATA29
CMD2-numword	E/S.28
STS1-addr	DATA30
STS1-Error	E/S.29
STS1-Inst	DATA31
STS1-SerReq	E/S.30
STS1-Reserved	DATA32
STS1-BCRecv	E/S.31
STS1-Busy	BT1553 INT NO IMSG GAP CMD 1
STS1-SubFlag	BT1553 INT BIT COUNT CMD 1
column continued on next page	column continued on next page

STS1-DBAcc	BT1553 INT WRONG BUS CMD 1
STS1-TerFlag	BT1553 INT CHANNEL CMD 1
STS2-addr	BT1553 INT BAD RTADDR CMD 1
STS2-Error	BT1553 INT LATE RESP CMD 1
STS2-Inst	BT1553 INT EARLY RESP CMD 1
STS2-SerReq	BT1553 INT NON CONT DATA CMD 1
STS2-Reserved	BT1553 INT PARITY CMD 1
STS2-BCRecv	BT1553 INT TWO BUS CMD 1
STS2-Busy	BT1553 INT MID BIT CMD 1
STS2-SubFlag	BT1553 INT INVERTED SYNC CMD 1
STS2-DBAcc	BT1553 INT LOW WORD CMD 1
STS2-TerFlag	BT1553 INT INVALID WORD CMD 1
DATA01	BT1553 INT HIGH WORD CMD 1
E/S	BT1553 INT NO IMSG GAP CMD 2
DATA02	BT1553 INT BIT COUNT CMD 2
E/S.1	BT1553 INT WRONG BUS CMD 2
DATA03	BT1553 INT CHANNEL CMD 2
E/S.2	BT1553 INT BAD RTADDR CMD 2
DATA04	BT1553 INT LATE RESP CMD 2
E/S.3	BT1553 INT EARLY RESP CMD 2
DATA05	BT1553 INT NON CONT DATA CMD 2
E/S.4	BT1553 INT PARITY CMD 2
DATA06	BT1553 INT TWO BUS CMD 2
E/S.5	BT1553 INT MID BIT CMD 2
DATA07	BT1553 INT INVERTED SYNC CMD 2
E/S.6	BT1553 INT LOW WORD CMD 2
DATA08	BT1553 INT INVALID WORD CMD 2
E/S.7	BT1553 INT HIGH WORD CMD 2
DATA09	BT1553 INT NO IMSG GAP STS 1
E/S.8	BT1553 INT BIT COUNT STS 1
DATA10	BT1553 INT WRONG BUS STS 1
E/S.9	BT1553 INT CHANNEL STS 1
DATA11	BT1553 INT BAD RTADDR STS 1
E/S.10	BT1553 INT LATE RESP STS 1
DATA12	BT1553 INT EARLY RESP STS 1
E/S.11	BT1553 INT NON CONT DATA STS 1
DATA13	BT1553 INT PARITY STS 1
column continued on next page	column continued on next page

E/S.12	BT1553 INT TWO BUS STS 1
DATA14	BT1553 INT MID BIT STS 1
E/S.13	BT1553 INT INVERTED SYNC STS 1
DATA15	BT1553 INT LOW WORD STS 1
E/S.14	BT1553 INT INVALID WORD STS 1
DATA16	BT1553 INT HIGH WORD STS 1
E/S.15	BT1553 INT NO IMSG GAP STS 2
DATA17	BT1553 INT BIT COUNT STS 2
E/S.16	BT1553 INT WRONG BUS STS 2
DATA18	BT1553 INT CHANNEL STS 2
E/S.17	BT1553 INT BAD RTADDR STS 2
DATA19	BT1553 INT LATE RESP STS 2
E/S.18	BT1553 INT EARLY RESP STS 2
DATA20	BT1553 INT NON CONT DATA STS 2
E/S.19	BT1553 INT PARITY STS 2
DATA21	BT1553 INT TWO BUS STS 2
E/S.20	BT1553 INT MID BIT STS 2
DATA22	BT1553 INT INVERTED SYNC STS 2
E/S.21	BT1553 INT LOW WORD STS 2
DATA23	BT1553 INT INVALID WORD STS 2
E/S.22	BT1553 INT HIGH WORD STS 2
DATA24	

C Feature Selection Scores

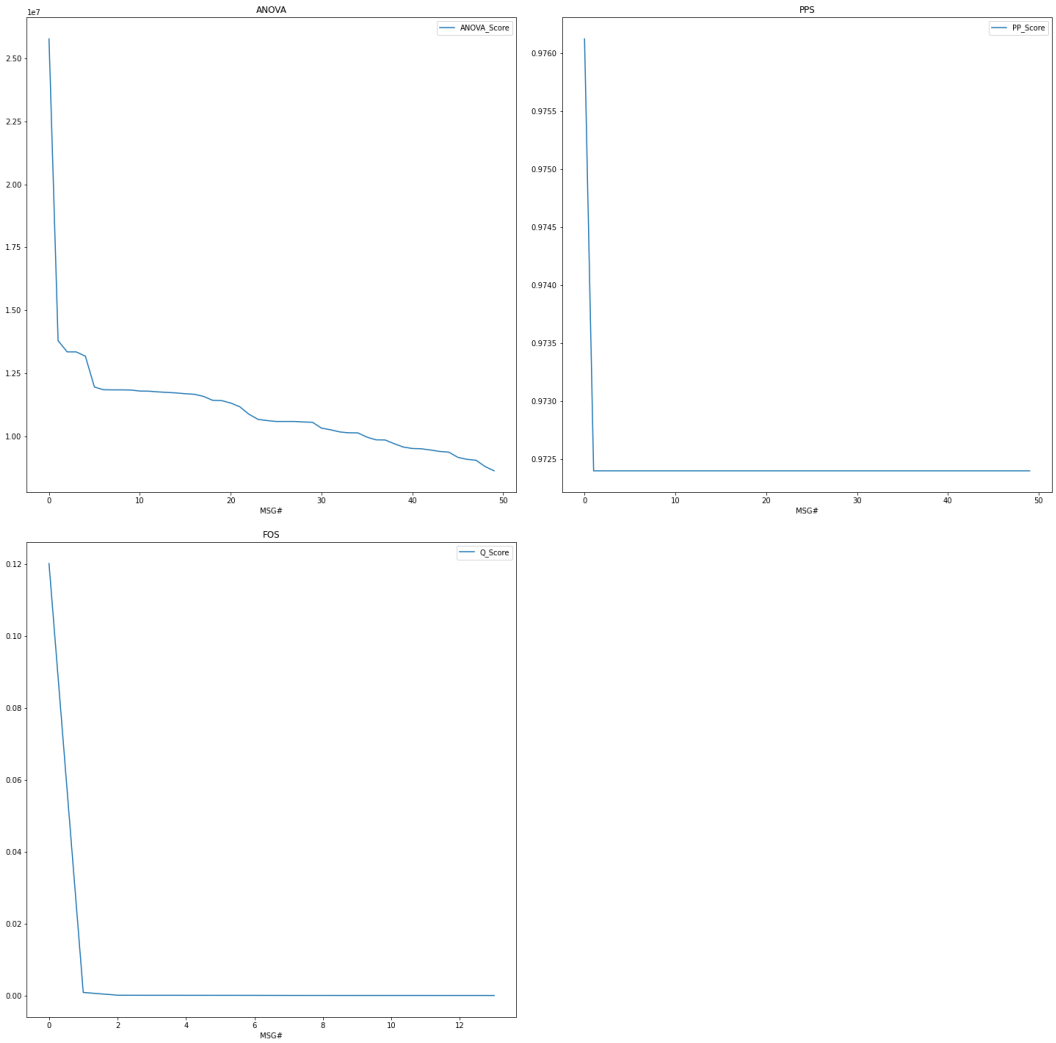

Figure C.1: NetDisrupt statusword 250820 feature score

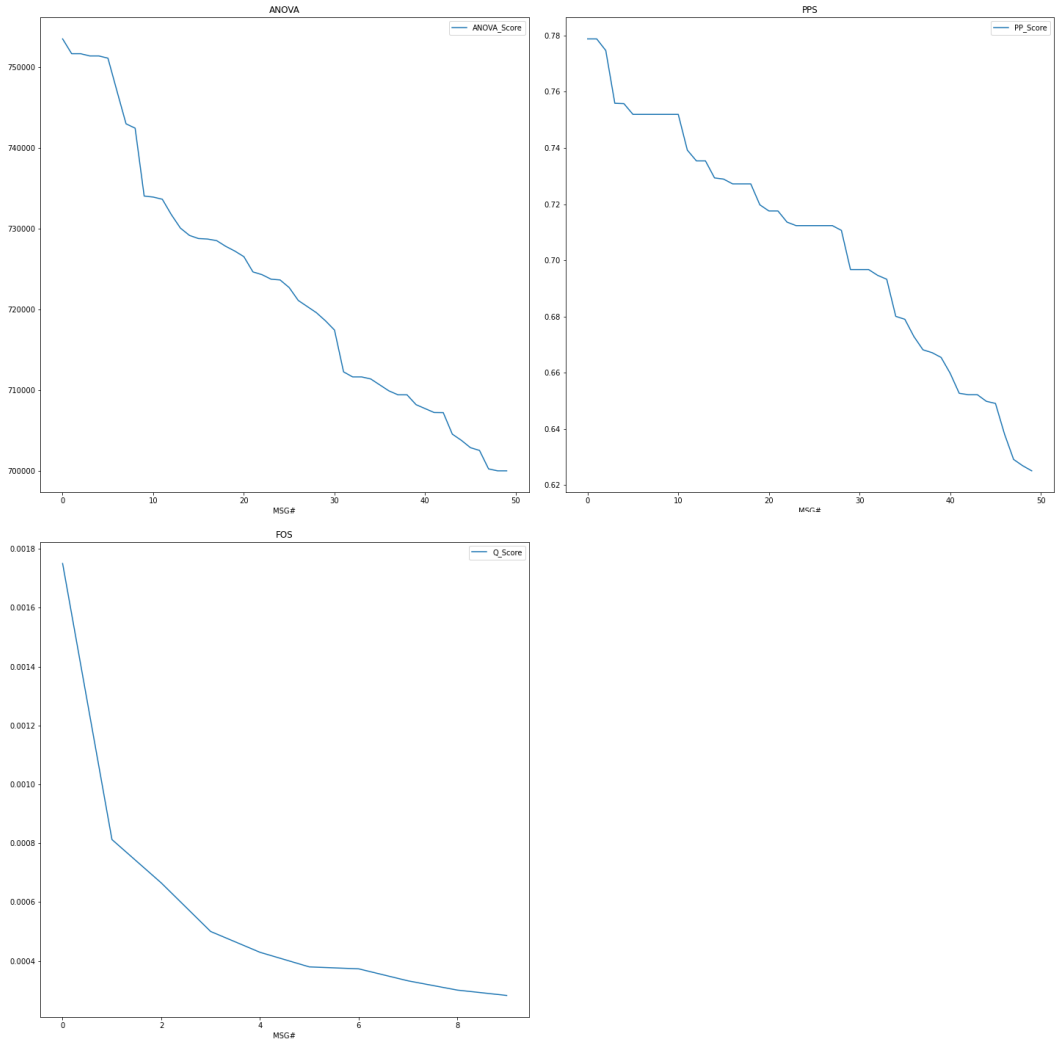

Figure C.2: Hijack rt18 sa6 w56 250820 feature score

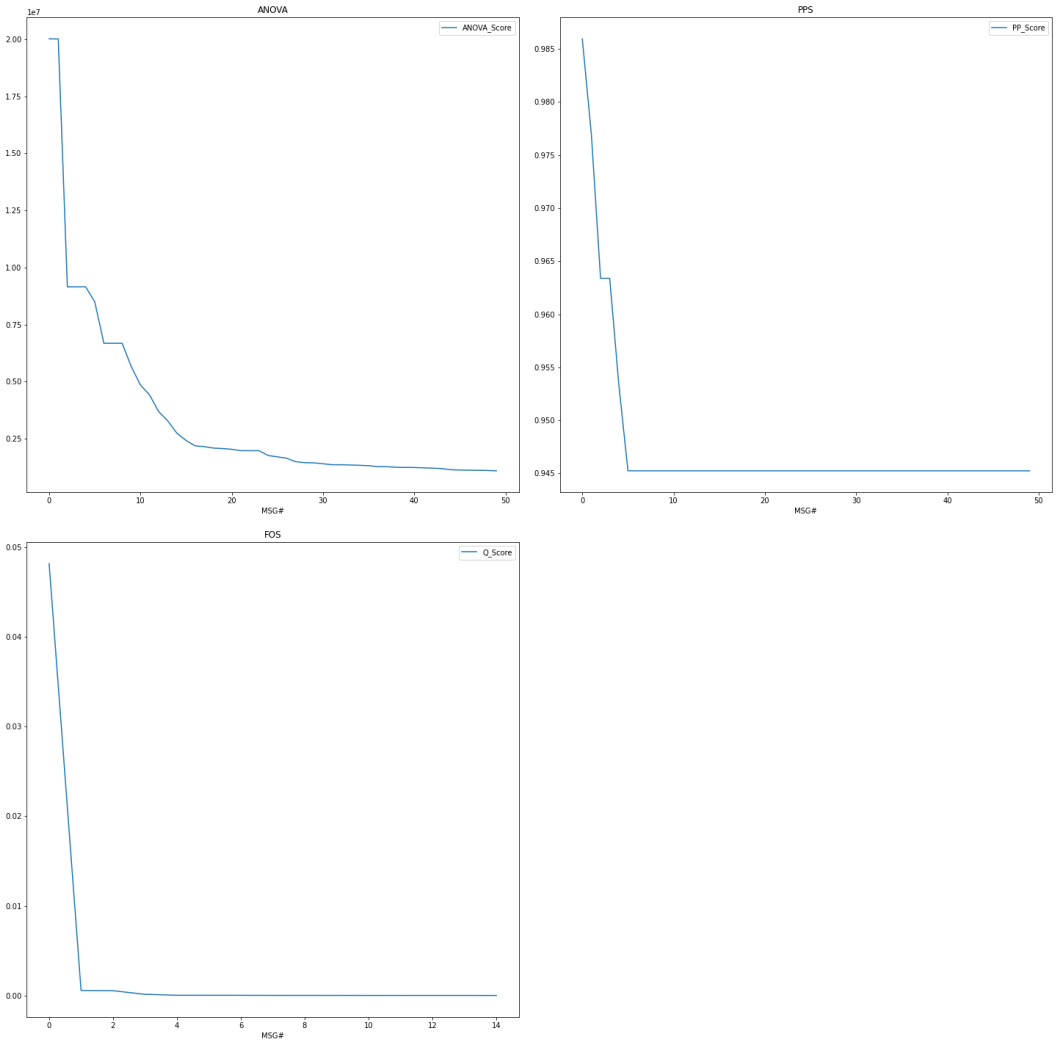

Figure C.3: RT-SA deny statusword rt18 sa1 250820 feature score

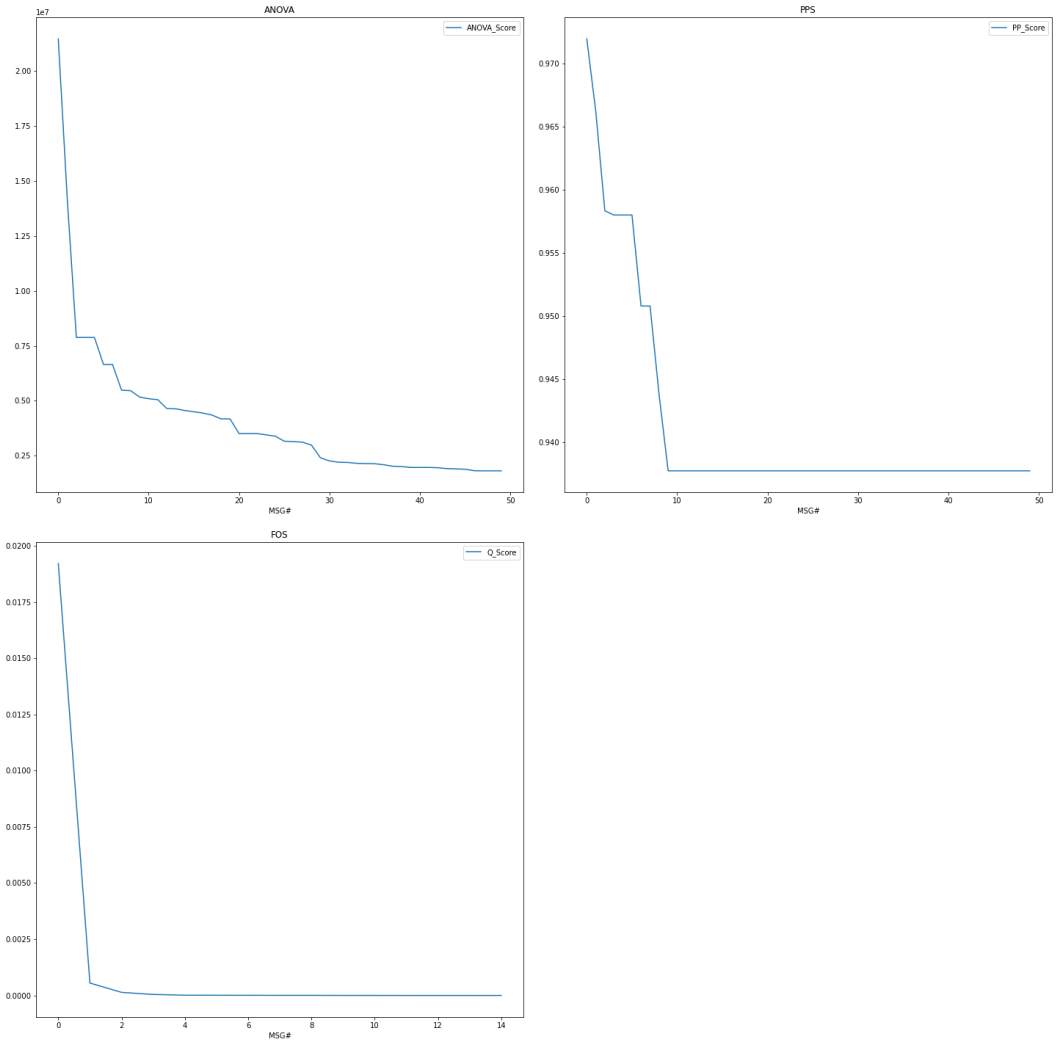

Figure C.4: RT-SA deny statusword rt18 sa32 250820 feature score

D General Model MAE Graphs

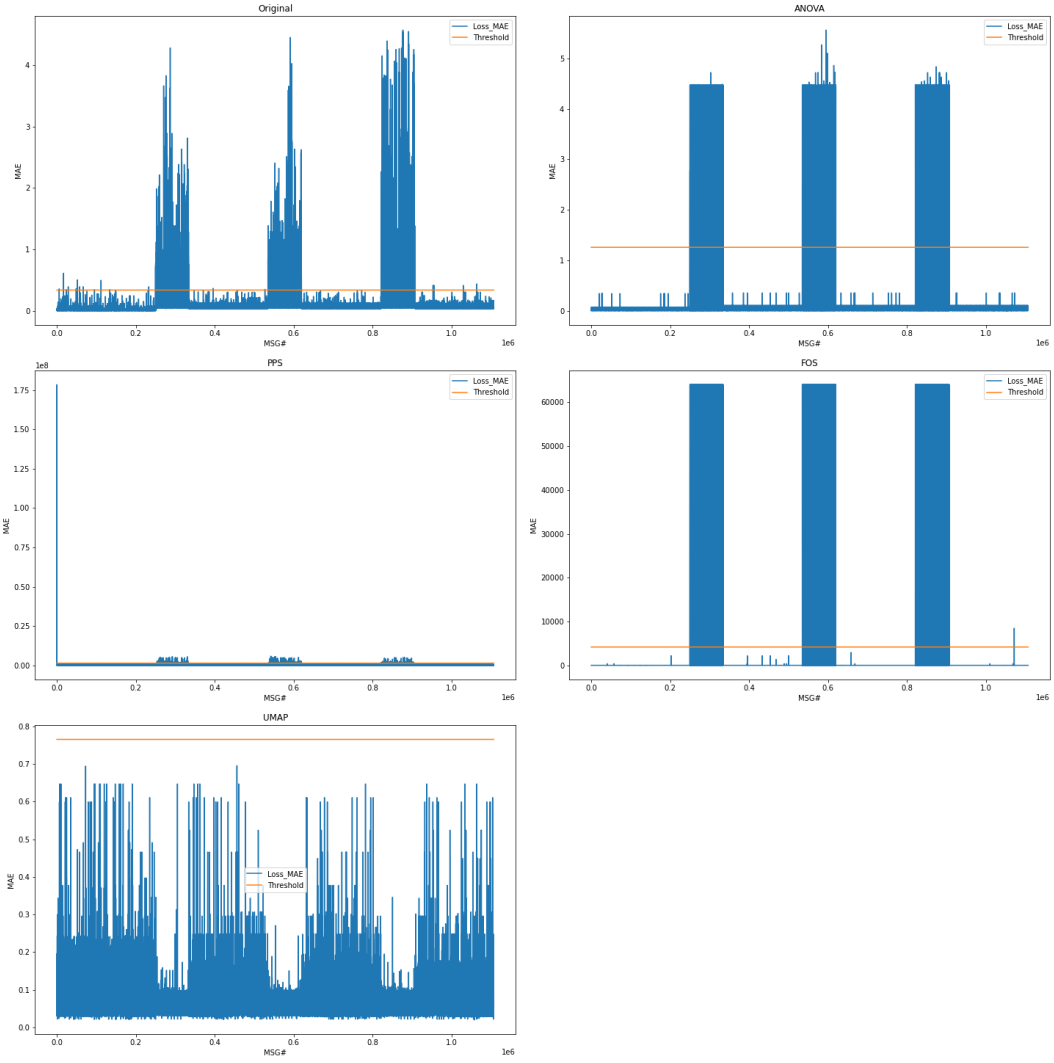

Figure D.1: MSE for Net Disrupt statusword 250820

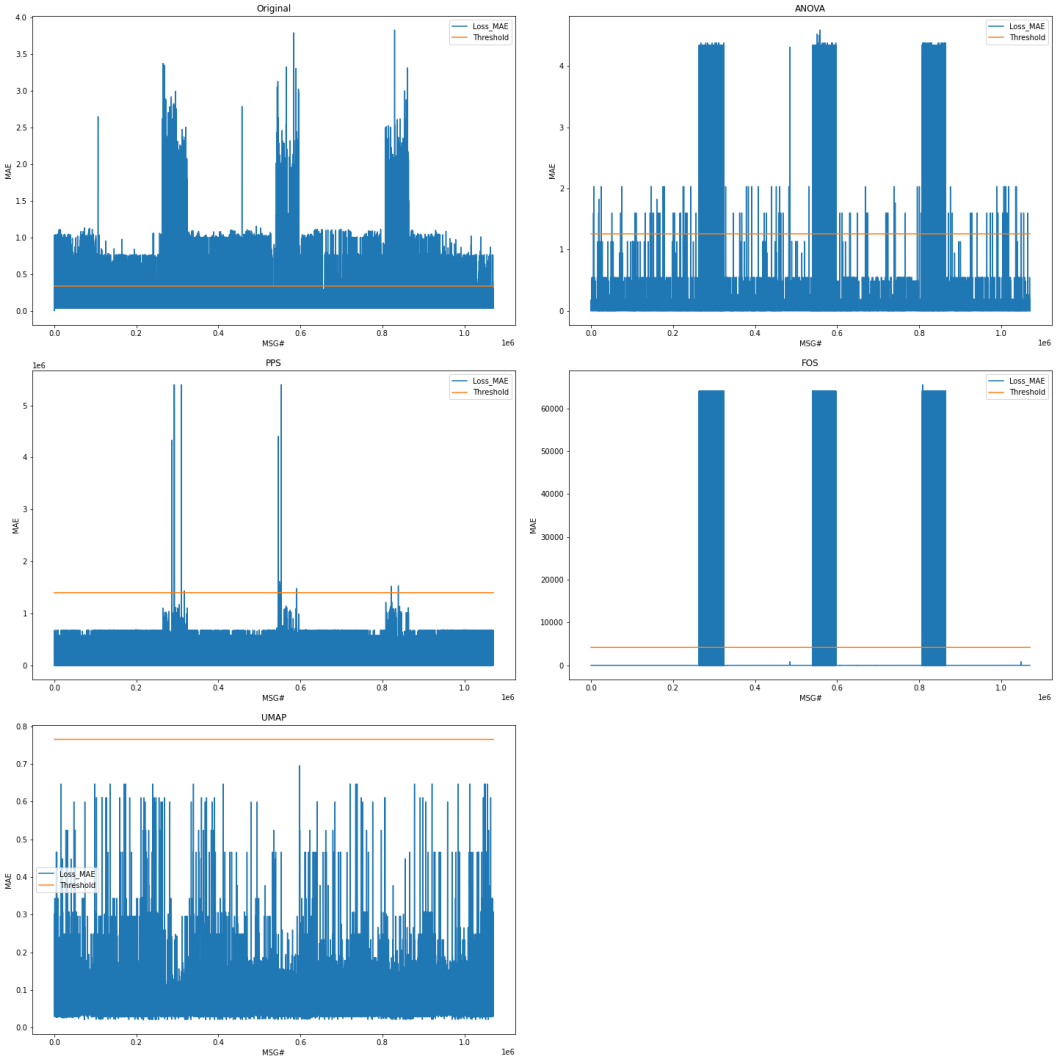

Figure D.2: MSE for Hijack rt18 sa6 w56 250820

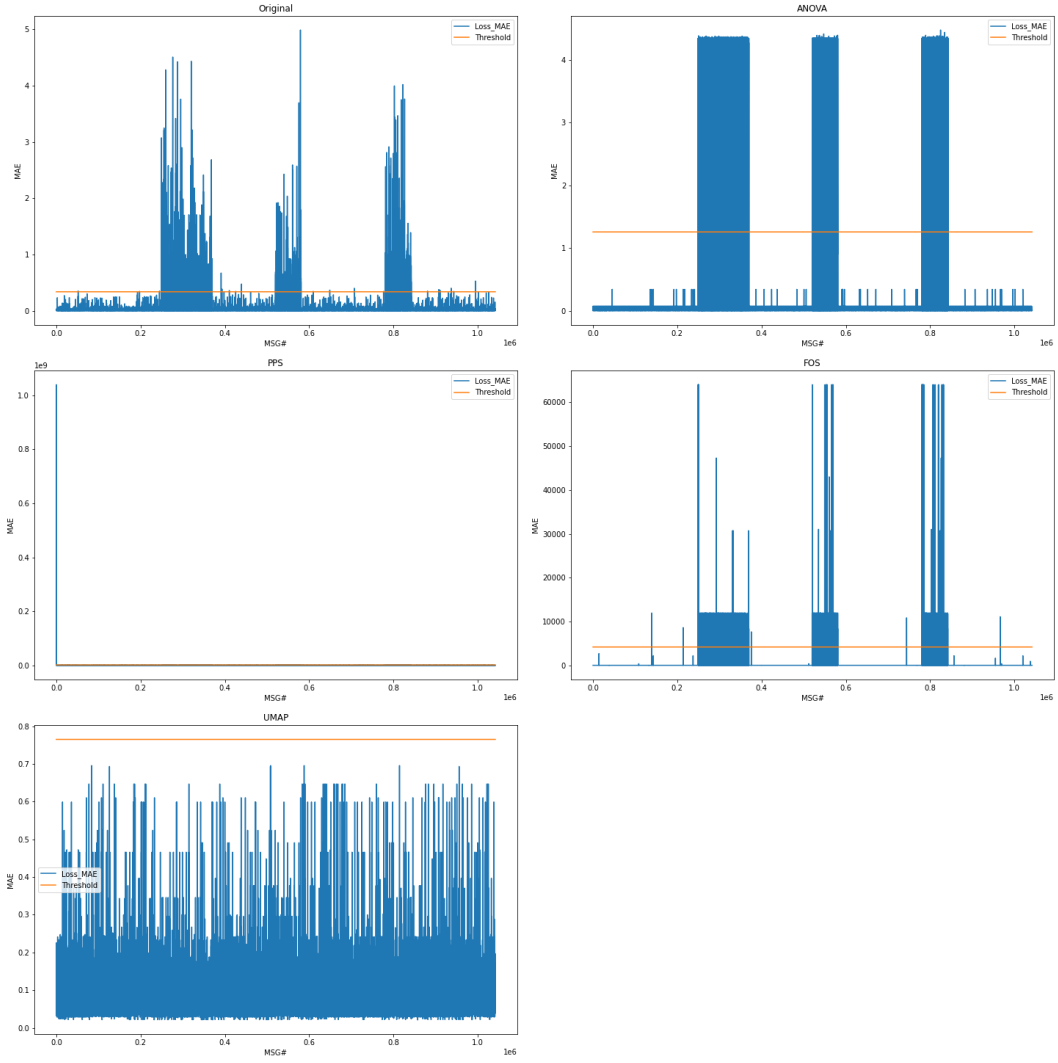

Figure D.3: MSE for RT-SA deny statusword rt18 sa1 250820

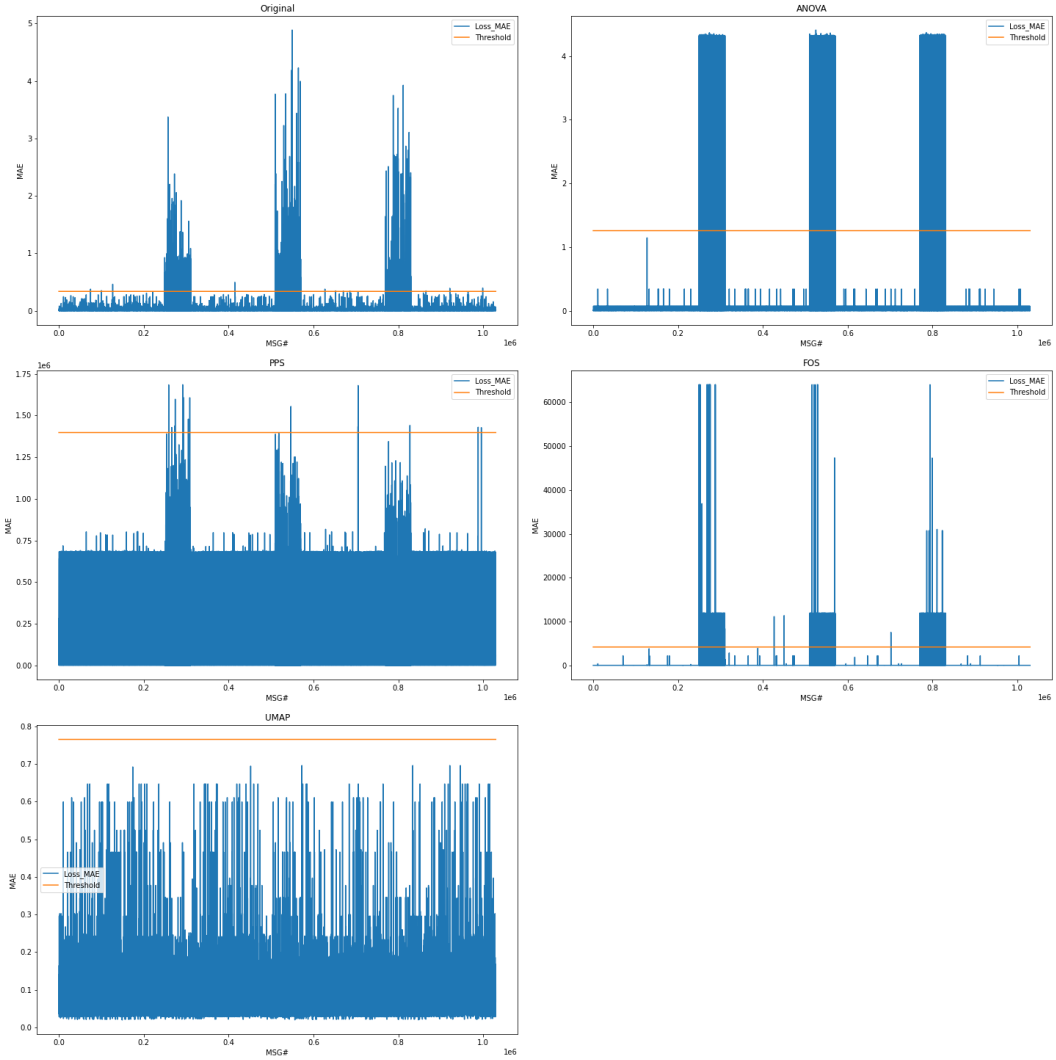

Figure D.4: MSE for RT-SA deny statusword rt18 sa1 rec2 250820

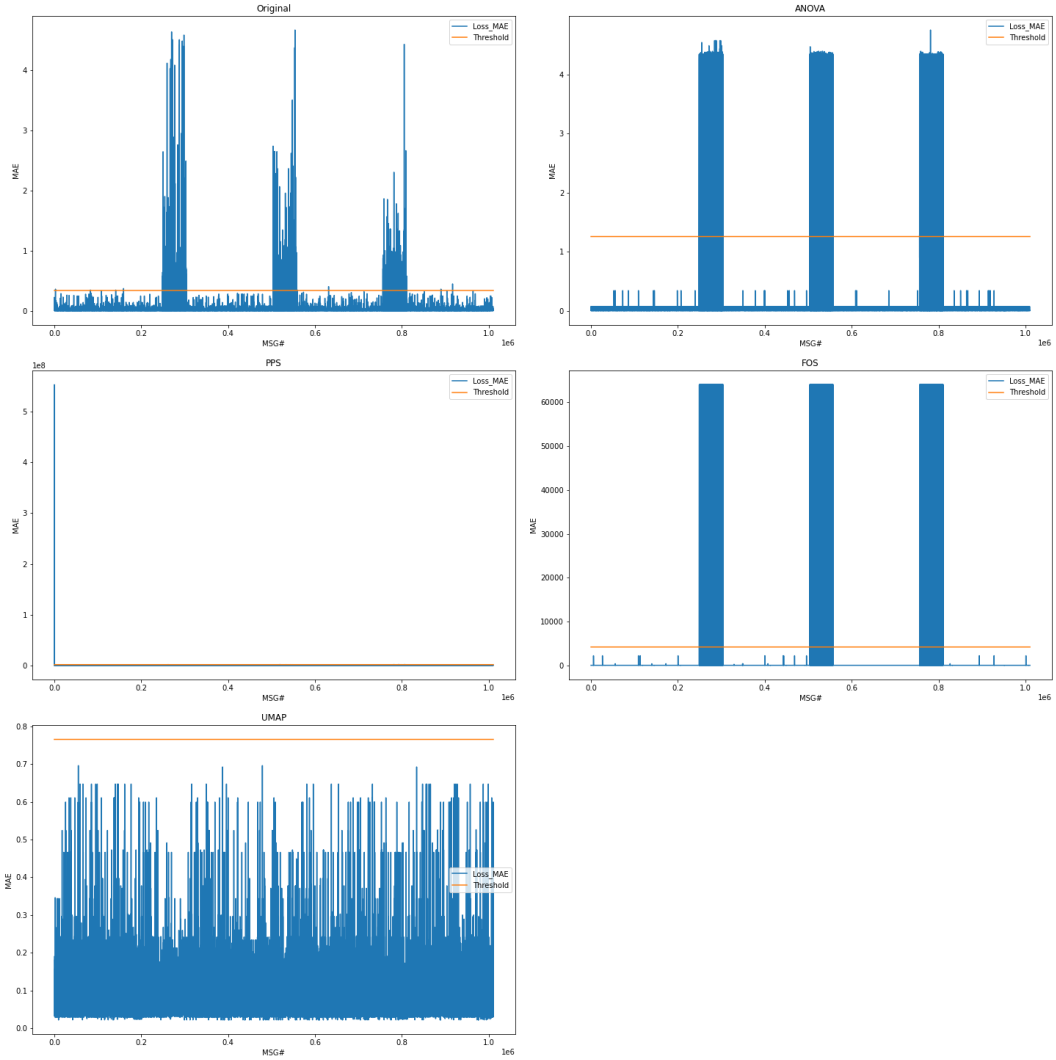

Figure D.5: MSE for RT-SA deny statusword rt18 sa32 250820

E Specific Model MAE Graphs

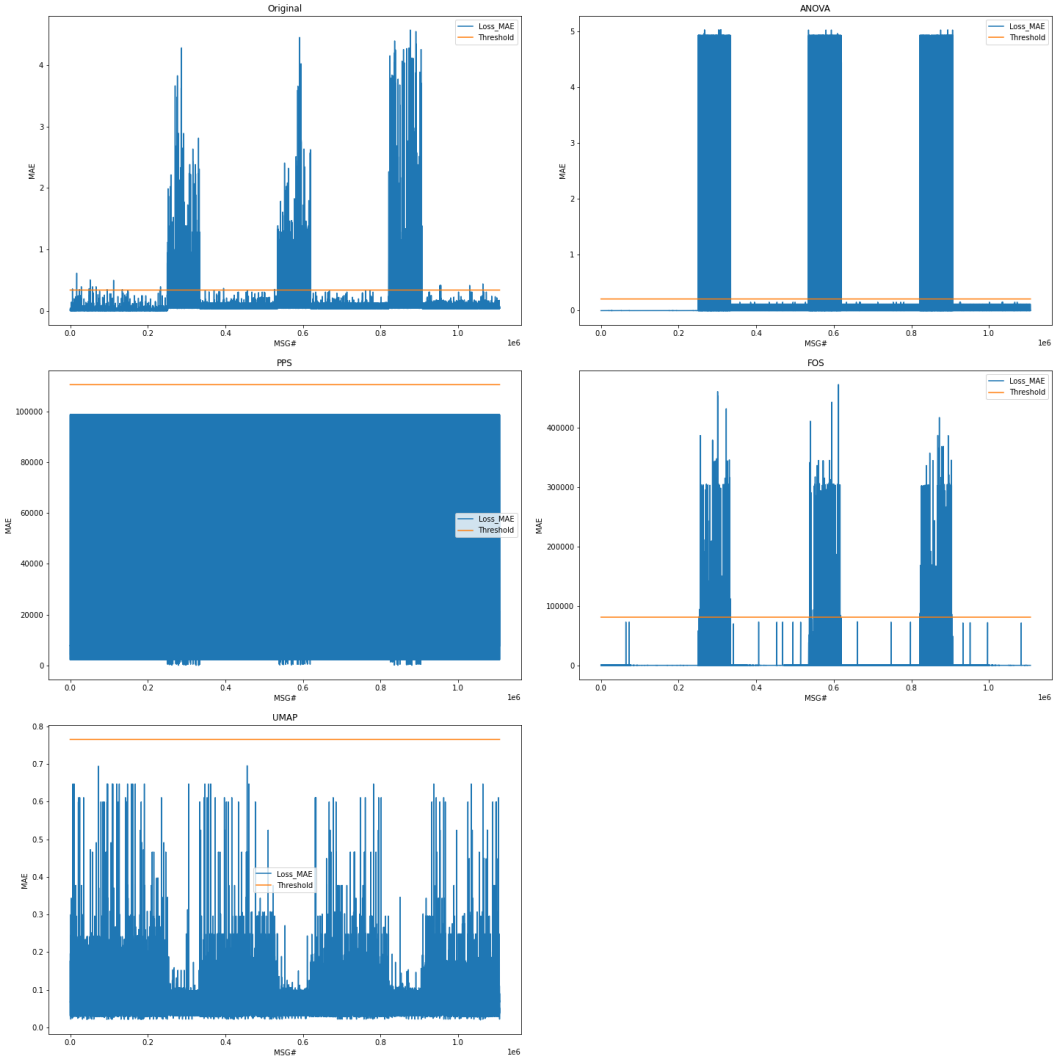

Figure E.1: MSE for Net Disrupt statusword 250820

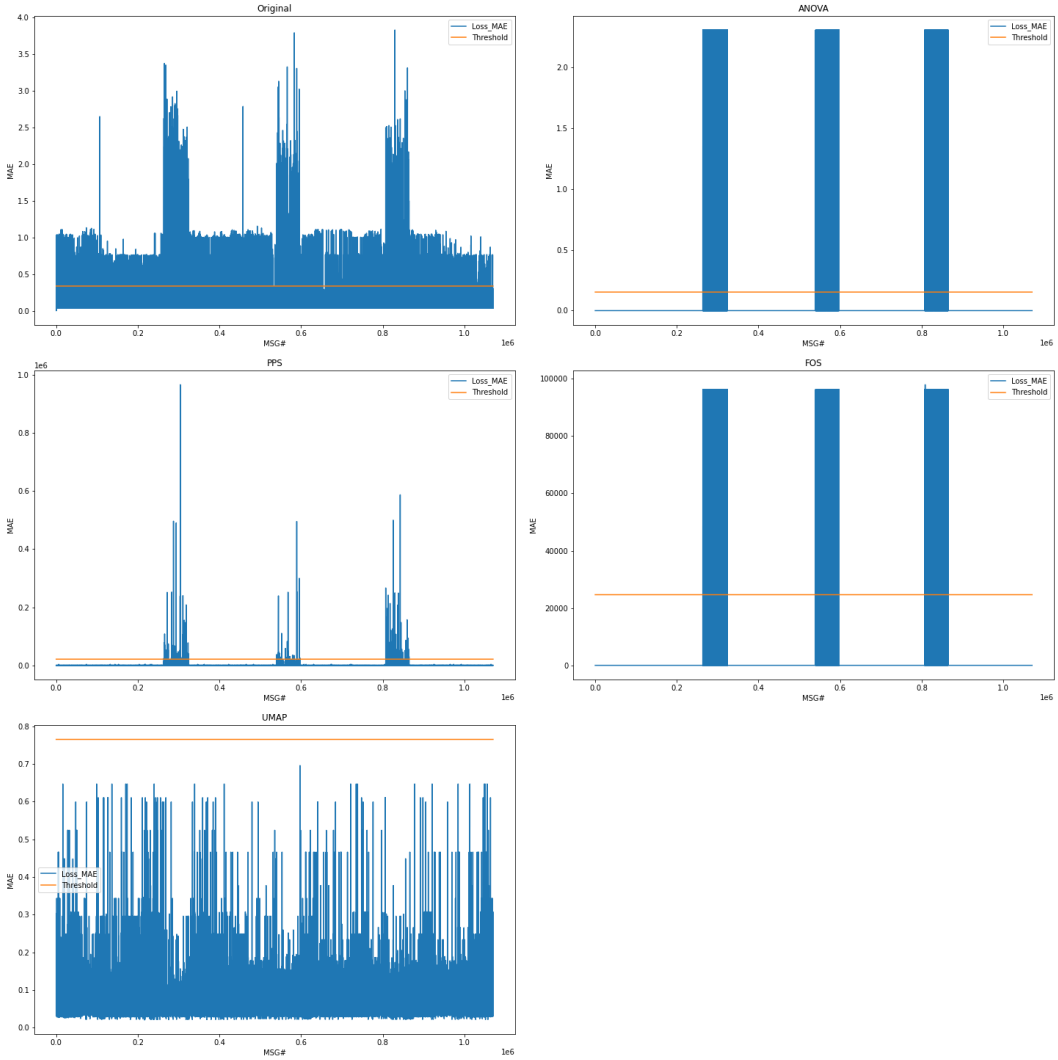

Figure E.2: MSE for Hijack rt18 sa6 w56 250820

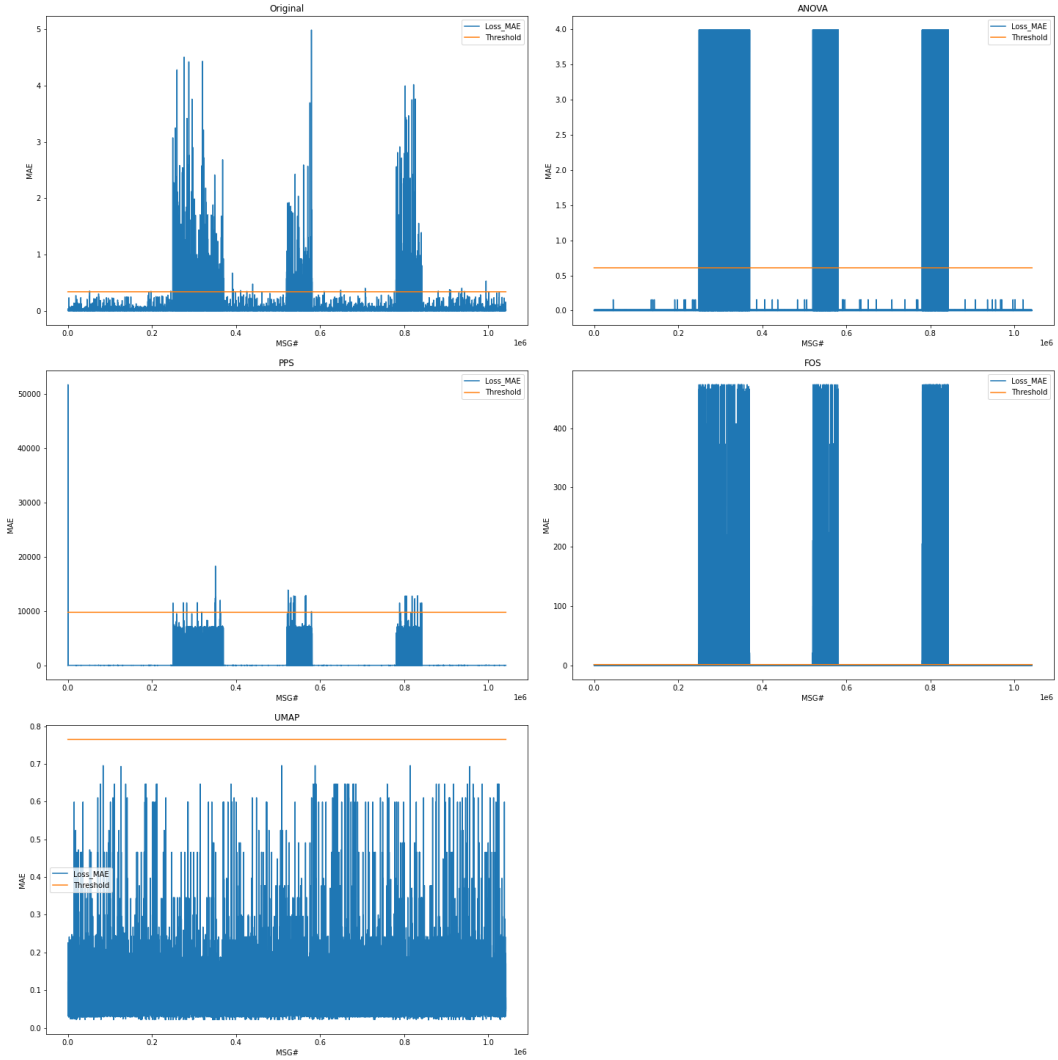

Figure E.3: MSE for RT-SA deny statusword rt18 sa1 250820

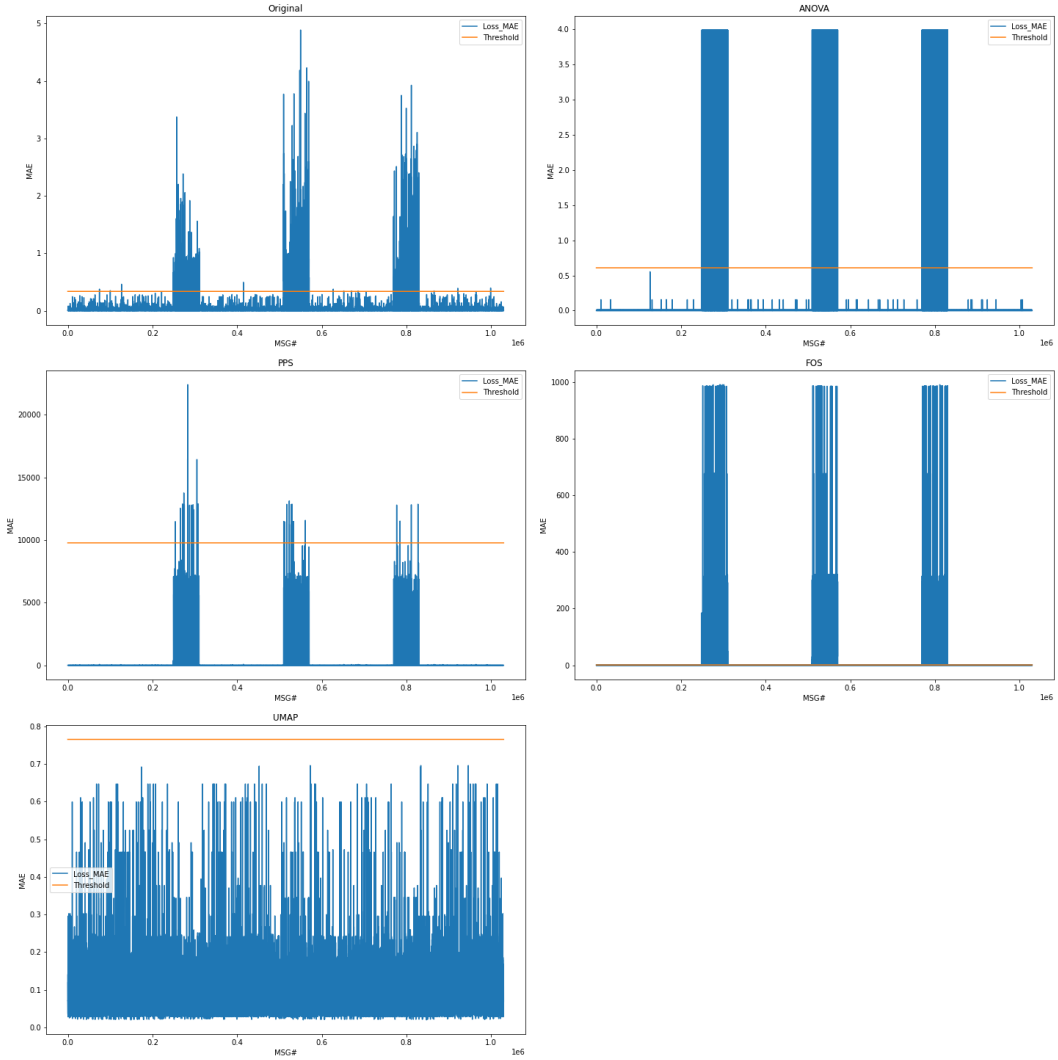

Figure E.4: MSE for RT-SA deny statusword rt18 sa1 rec2 250820

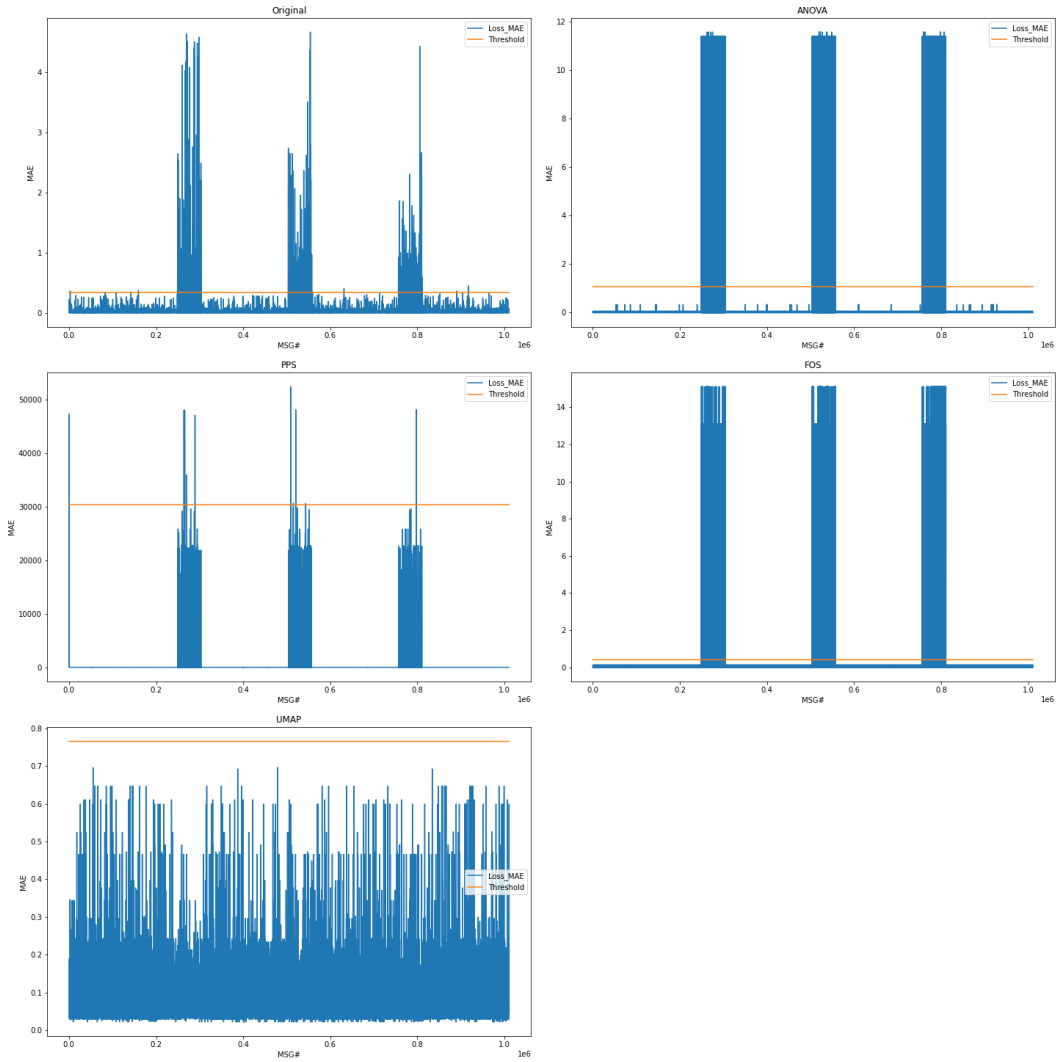

Figure E.5: MSE for RT-SA deny statusword rt18 sa32 250820