College of Engineering, Design and Physical Sciences
Department of Electronic and Computer Engineering
Brunel University London

Green AI for Industry 4.0: Energy-Efficient Generalised Deep Learning Approaches to Induction Motor Condition Monitoring

A thesis submitted for the degree of Doctor of Philosophy by

Ayman Elhalwagy

Brunel University London
Department of Computer Science
Uxbridge
Middlesex
UB8 3PH
United Kingdom
T: +44 1895 203397
F: +44 (0) 1895 251686

Abstract

The field of Industry 4.0 has seen a significant increase in demand for efficient and effective methods of data-driven analysis, particularly in the domain of condition monitoring for machinery. This thesis explores the use of deep learning techniques to address the challenges faced in this field, focusing on the development of energy-efficient and generalised approaches for Induction Motor fault detection and classification.

The first part of the thesis introduces the Multi-Channel LSTM-Capsule Autoencoder, a novel Neural Network (NN) architecture designed to tackle issues such as generalisation ability, the need for large volumes of labelled data, and understanding spatial context in multivariate time series data from a single data source. Experimental results demonstrate the architecture's resilience to overfitting, improved training efficiency, and state-of-the-art performance in outlier detection.

Building upon the LSTM-Capsule Autoencoder, the second part presents the Dataset Fusion algorithm, a novel dataset composition method for fusing periodic signals from multiple homogeneous datasets into a single dataset while retaining unique features. The proposed approach, tested on a case study of 3-phase current data from Induction Motor fault datasets, significantly outperforms conventional training approaches and effectively generalises across all datasets. The algorithm's effectiveness under non-ideal conditions and its computational efficiency, in line with the principles of Green AI, highlight its potential for practical use in real-world applications.

The final part introduces the Order Domain Transformer (ODT), a pre-processing algorithm designed to standardise and align the frequency components of signals from different motors, enabling the fusion of multiple heterogeneous datasets in the frequency domain. Experimental results indicate that using ODT maintains performance on data from the same motors but results in a substantial improvement in cross-motor generalisation and model performance. The ODT approach demonstrates the potential to train a single model for multiple motors, optimising the utilisation of available labelled data and reducing the computational resources required for training.

The proposed methods in this thesis progressively address the challenges of working with single data sources, multiple homogeneous data sources, and multiple heterogeneous datasets, providing a comprehensive framework for data-driven fault detection and classification in industrial settings.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisory team and the academics I worked with throughout my studies. In particular, I am indebted to Prof Tatiana Kalganova, who took a chance on me nearly 4 years ago. Her guidance, support, and the opportunities she provided have been instrumental in helping me excel as a researcher.

I had an incredible time working with our AI research team. We shared ideas, celebrated successes and commiserated over failures. We challenged and motivated each other, creating an environment that encouraged growth and innovation. That made the whole journey so much more enjoyable.

I would also like to thank my friends for their unwavering support. They patiently listened to my endless complaints and rants about experiments going wrong and new crazy ideas (even though they never understood the technical details!), provided moral support, and celebrated my small wins with me. Their friendship has been a source of strength throughout this journey.

Last but certainly not least, I owe a debt of gratitude to my family, particularly my parents. Their belief in me, their constant motivation, and the incredible example they've set have been the foundation of my success. I hope to one day provide my own children with the same level of opportunities and support that you have afforded me.

To all who have been part of this journey, thank you. Your contributions, big and small, have made this achievement possible.

Declaration of Authorship

I, Ayman Elhalwagy, declare that the work completed in this thesis is my original work conducted in accordance with the Code of Practice for Research Degrees. The work presented in this thesis has not been used in any other submission for an academic award.

Signature: [illegible]

Date: 23/06/2024

Contents

List of Figures

List of Tables

List of Acronyms

AE	Autoencoder
ANOVA	Analysis Of Variance
CapsNet	Capsule Network
CNN	Convolutional Neural Network
DF	Dataset Fusion
DL	Deep Learning
EMF	Electromagnetic field
FFT	Fast Fourier Transform
IM	Induction Motor
KW	Kruskal-Wallis
LSTM	Long Short-Term Memory
MCSA	Motor Current Signature Analysis
ML	Machine Learning
MAE	Mean Absolute Error
MSE	Mean Squared Error
NN	Neural Network
ODT	Order Domain Transformer
OT	Order Tracking
RNN	Recurrent Neural Network
SCADA	Supervisory Control And Data Acquisition
SKAB	Skoltech Anomaly Benchmark
TS	Time Series

Chapter 1

Introduction

The transition towards Industry 4.0 has exposed a critical gap in current approaches to Induction Motor (IM) fault detection. Although numerous methods have been proposed for anomaly detection and fault classification in IMs, these techniques often fail to generalise across different motors and operating conditions. This limitation severely hinders their real-world applicability, as industrial environments typically feature a wide variety of motor types and configurations.

Deploying fault detection systems at scale requires flexible, adaptable solutions that can be easily transferred to new motors without extensive retraining or recalibration. However, existing research has largely overlooked this crucial challenge, focusing instead on optimising performance for specific, narrow datasets.

This ties directly into the concept of Green AI, which was first formalised by Schwartz et al. [6], who advocated for AI research that values computational efficiency alongside accuracy and performance metrics. Although the original definition primarily focused on reducing the carbon footprint associated with training large neural networks, this thesis proposes a broader interpretation that encompasses not only computational efficiency but also data efficiency.

In industrial contexts such as IM condition monitoring, data collection itself carries significant costs: motors must be operated for extended periods to capture degradation signals, and additional instrumentation increases both financial and energy expenditures. Furthermore, training inflexible fault detection systems that need to be adjusted for each data source will invariably result in increased computational power requirements when deployed at scale in industry.

This thesis aims to bridge this gap by proposing novel Machine Learning (ML) architectures and signal processing techniques specifically designed to enable robust, generalisable, and green IM fault detection that can effectively adapt to diverse motor types and operating conditions, thus reducing computational power requirements, increasing energy efficiency, and enabling efficient use of data.

This work represents a significant step towards truly scalable, industry-ready fault detection systems. The techniques presented in this thesis have the potential to greatly streamline the deployment and maintenance of condition monitoring solutions, thus accelerating the adoption of predictive maintenance strategies across various sectors. Ultimately, this research contributes to the broader goal of improving the efficiency, reliability, and sustainability of industrial processes in the era of Industry 4.0.

1.1 Background

The ongoing transition from Industry 3.0 to Industry 4.0 has catalysed a profound shift in the way industrial data is collected and used [7]. As companies seek to optimise their processes and enhance decision-making capabilities, technologies such as Digital Twin have emerged as powerful tools for modeling and analysing machine performance [8]. One of the most promising applications of this data-driven approach is in the development of smart condition monitoring systems.

By enabling early fault detection and predictive maintenance, these systems offer a range of benefits, from extending equipment lifespan to reducing the frequency and cost of repairs [9][10]. However, realising these advantages hinges on the ability to effectively monitor the health of industrial machinery, particularly IMs.

IMs, especially three-phase squirrel cage variants, are the workhorses of modern industry [11], accounting for over 80% of all motors used in industrial applications [12]. Their ubiquity, coupled with the growing demand for condition monitoring solutions, has fueled a surge of research into various fault detection methods. Traditional approaches, such as vibration and acoustic analysis, have been widely studied but suffer from significant drawbacks. These techniques are often invasive, difficult to install, and limited in their ability to detect electrical faults in the early stages [13] [14].

Motor Current Signature Analysis (MCSA) has emerged as a promising alternative to traditional vibration analysis. MCSA is a non-invasive technique that can detect both electrical and mechanical faults due to their effect on the Electromagnetic field (EMF) between the rotor and stator [13]. This technique has been successfully applied to detect specific types of faults, such as broken rotor bars [15], and has shown potential for multi-fault detection and application-specific scenarios such as wind turbines [16]. The non-invasive nature of MCSA and its ability to detect a wide range of faults make it a promising tool for industrial condition monitoring.

Traditionally, statistical modeling and manual analysis were used to perform fault detection in IMs. For example, time-domain and frequency-domain analysis of vibration signals were commonly employed to identify fault signatures [17]. However, these approaches often require extensive domain expertise and human involvement, which hinders their effectiveness in real-world industrial environments, where large quantities of motors may operate under varying loads and speeds.

To address these shortcomings, ML based techniques have been increasingly proposed in literature. ML enables more powerful non-linear modeling and can adapt to environment-specific conditions and data distributions [18]. However, despite the potential of ML-based fault detection techniques, several limitations hinder their real-world application. The cold start problem, where a dataset must be collected to train a model, poses a significant challenge in industrial settings where labeled fault data may be scarce. Although this problem is rarely addressed in literature, it is a crucial consideration for the successful deployment of these techniques in real-world applications. Additionally, the large data requirements from each source due to the inability to adapt to environmental context and data heterogeneity can be a barrier to deployment in diverse industrial environments. The computational power needed to train new models for each source also raises concerns about the environmental impact of these techniques [19]. Furthermore, the lack of adaptability to new fault types limits the reliability and practical applicability of fault classification models in real-world scenarios, where unexpected faults may occur.

Addressing these limitations is crucial for enabling the real-world application of the techniques proposed in the literature. Given the critical role of IMs in industrial processes, developing robust, adaptive, and scalable fault detection techniques is essential for realising the full potential of Industry 4.0 and ensuring the reliability and efficiency of industrial systems. This thesis aims to tackle these challenges by proposing novel ML architectures and signal processing techniques that enable the generalisation of IM fault detection models across different motors and operating conditions.

1.2 Justification

Building upon the challenges and limitations outlined in the background, this thesis systematically addresses the key obstacles hindering the real-world application of fault detection and classification techniques for IMs. The presented work proposes novel approaches that not only overcome these limitations but also advance the state-of-the-art in IM condition monitoring.

The research begins with the introduction of a novel architecture for the detection of anomalies in multivariate data, combining the complementary strengths of capsule networks [20] and LSTM [1]. This Capsule LSTM hybrid model effectively captures complex temporal dependencies and spatial relationships, enabling more accurate anomaly detection compared to existing methods.

Next, the Capsule LSTM architecture is applied to tackle the challenge of multiple homogeneous motor data sources. A novel Dataset Fusion signal processing technique is proposed, which intelligently combines data from various sources to build a generalised training dataset. This approach significantly reduces the reliance on large volumes of data from individual sources, highlighting the importance of data variety over sheer volume, and ultimately resulting in higher energy efficiency through reduced computational power requirements.

Finally, the issue of heterogeneous data sources in fault classification is addressed by introducing an order domain transformer. Inspired by traditional order tracking techniques [21], the proposed method normalises IM signals with different sampling and operating frequencies into a common order domain. This enables the training of fault classification models that can achieve state-of-the-art accuracy on completely unseen IMs, a key step towards truly generalisable condition monitoring solutions.

Throughout this thesis, the proposed techniques are rigorously evaluated using extensive empirical studies, demonstrating their superior performance compared to existing state-of-the-art methods. The results of these studies validate the effectiveness of the presented approaches in addressing the critical limitations of current fault detection and classification techniques, paving the way for more robust, adaptable and industry-ready IM condition monitoring solutions that are also energy efficient.

Throughout this thesis, the term ”features” when discussing input data refers specifically to individual sensor channels or inputs in a dataset. For example, in the context of motor current data, a ”feature” might represent one of the three-phase current signals measured from the motor. This definition helps distinguish between input features (sensor channels) and the derived features that might be extracted through signal processing or neural network operations.

The research presented in this thesis was motivated by practical challenges encountered during an industrial collaboration with Voltvision, a power network analysis start-up. Although extensive testing was conducted on real-world data from Voltvision's industrial clients, these datasets cannot be published because of confidentiality agreements. This industrial background ensured that the methodologies developed were designed with real-world deployment considerations at the forefront. To rigorously validate the approaches while respecting confidentiality constraints, carefully selected open-source datasets that closely resemble the characteristics of the industrial problems addressed were used for the experiments presented in this thesis, which allowed for transparent scientific validation while maintaining the practical relevance of the solutions to industrial applications. These datasets will be introduced later in this chapter.

1.3 Thesis Contributions

The contributions to science presented in this thesis, which are visualised in Figure 1.1, can be summarised as follows:

1. **A novel Multi-Channel LSTM-Capsule Autoencoder Network for Anomaly Detection on Multivariate Data**: This contribution introduces a novel NN architecture which utilises a Long-Short-Term-Memory (LSTM) encoder and Capsule decoder in a multi-channel input Autoencoder architecture for use on multivariate TS data. Experimental results show that using Capsule decoders increases the resilience of the model to overfitting and improves training efficiency. Additionally, results also show that the proposed model can learn multivariate data more consistently, and was not affected by outliers in the training data. The proposed architecture was also tested on an open-source benchmark, where it achieved state-of-the-art performance in outlier detection, and performs best overall over the metrics tested. This contribution has the potential to enhance anomaly detection in various industrial applications, leading to more accurate and reliable fault detection systems. This work is covered in Chapter 3, the contents of which has been published in *MDPI Applied Sciences journal, number 22: 11393* [2].

2. **A Dataset Fusion Algorithm for Generalised Anomaly Detection in Homogeneous Periodic Time Series Datasets**: This contribution introduces "Dataset Fusion," a novel dataset composition algorithm for fusing periodic signals from multiple homogeneous datasets whilst retaining unique patterns for generalised anomaly detection. The proposed approach, which was tested on a case study of three-phase current data from two different homogeneous IM fault datasets on anomaly detection, outperforms conventional training approaches and effectively generalises across all datasets. Furthermore, when tested with varying percentages of the training data, results show that using only a small proportion of the training data, translating to a significant reduction in computational power, results in only a marginal decrease in performance, demonstrating the advantages of the proposed approach in terms of both performance and computational efficiency, and aligning with Green AI principles. This contribution has the potential to significantly reduce the computational resources required for training anomaly detection models, making

them more environmentally friendly and suitable for real-world deployment. This work is presented in Chapter 4, the contents of which has been published in *IEEE Access, pages 121212-121230* [22].

3. **Heterogeneous Induction Motor Current Dataset Fusion for Efficient Generalised MCSA-Based Fault Classification**: This contribution introduces the ODT, a pre-processing algorithm designed to standardise and align the frequency components of signals from different motors, thereby enhancing the learning capability of Neural Networks. The study explores the computational efficiency of the algorithm, highlighting its suitability for real-time applications. Experimental results indicate that using ODT maintains performance on data from the same motors trained on but results in a substantial improvement in cross-motor generalisation and model performance. This indicates a future potential to train a single model for multiple motors, thereby optimising the utilisation of available labelled data, and reinforcing the principles of Green AI by significantly reducing the computational resources required for training. This work is presented in Chapter 5, the contents of which is published in the conference publication *IEEE Intl Conf on Dependable, Autonomic and Secure Computing, pages 576-581* [23].

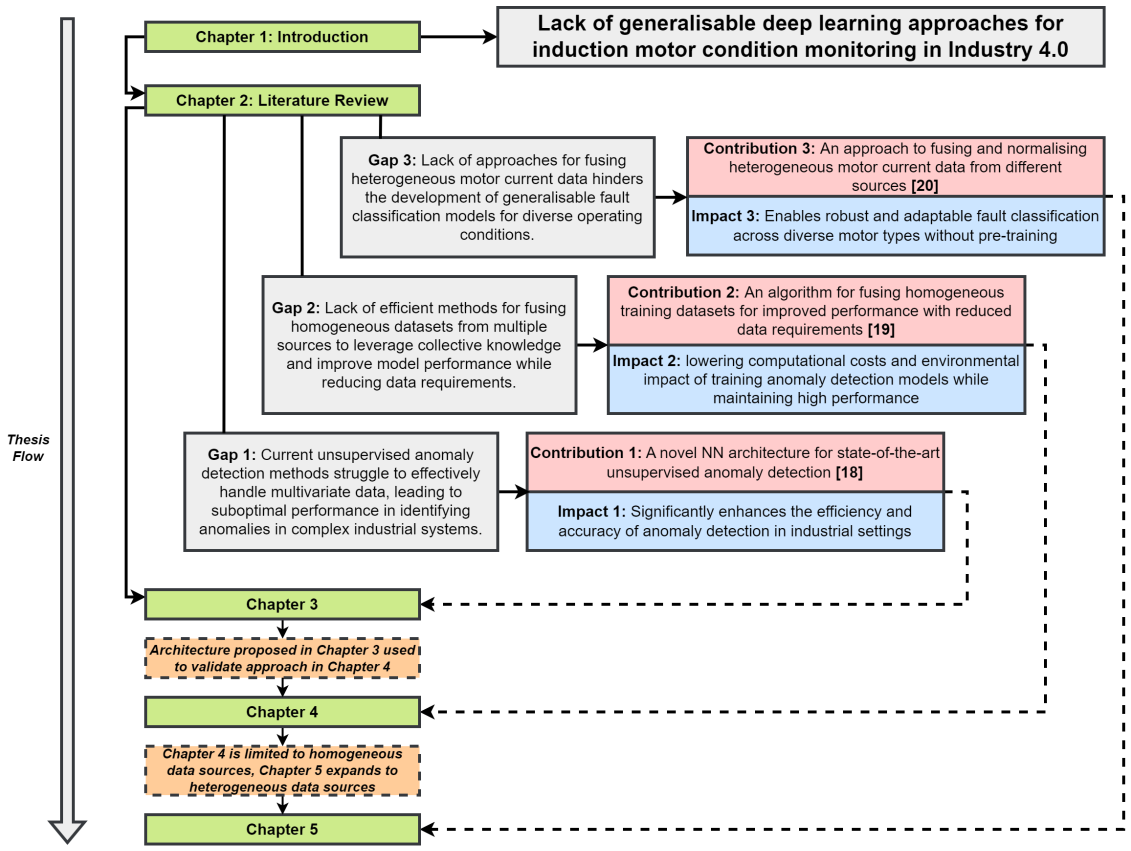

Thesis Flow
Chapter 1: Introduction
Lack of generalisable deep learning approaches for induction motor condition monitoring in Industry 4.0
Chapter 2: Literature Review
Gap 3: Lack of approaches for fusing heterogeneous motor current data hinders the development of generalisable fault classification models for diverse operating conditions.
Contribution 3: An approach to fusing and normalising heterogeneous motor current data from different sources [20]
Impact 3: Enables robust and adaptable fault classification across diverse motor types without pre-training
Gap 2: Lack of efficient methods for fusing homogeneous datasets from multiple sources to leverage collective knowledge and improve model performance while reducing data requirements.
Contribution 2: An algorithm for fusing homogeneous training datasets for improved performance with reduced data requirements [19]
Impact 2: lowering computational costs and environmental impact of training anomaly detection models while maintaining high performance
Gap 1: Current unsupervised anomaly detection methods struggle to effectively handle multivariate data, leading to suboptimal performance in identifying anomalies in complex industrial systems.
Contribution 1: A novel NN architecture for state-of-the-art unsupervised anomaly detection [18]
Impact 1: Significantly enhances the efficiency and accuracy of anomaly detection in industrial settings
Chapter 3
Architecture proposed in Chapter 3 used to validate approach in Chapter 4
Chapter 4
Chapter 4 is limited to homogeneous data sources, Chapter 5 expands to heterogeneous data sources
Chapter 5

Figure 1.1: Thesis structure

1.4 Datasets Overview

To ensure a rigorous validation of the methodologies proposed in this thesis, several datasets were carefully selected. Each dataset was chosen to represent specific aspects of industrial motor condition monitoring problems, providing suitable test beds for the different techniques developed. This section provides an overview of these datasets, their characteristics, and their relevance to each chapter's contributions.

1.4.1 Summary of Datasets

Table 1.1 provides a comprehensive summary of all datasets used throughout this thesis, including their sources, key characteristics, and the chapters in which they are used.

Table 1.1: Summary of datasets used throughout this thesis

Dataset Name	Source	Data Type	Features	Sampling Rate	Size	Used in
Drone Control Dataset	Sternharz et al. [24]	Control signals	3 (pitch, roll, throttle)	150 Hz	90,000 samples (training) 4,500 samples (validation) 4,500 samples (testing)	Chapter 3
SKAB Benchmark	Katser and Kozitsin [25]	Water circulation system	8 (various sensors)	Varies	35 subsets	Chapter 3
Inter-turn Short Circuit Dataset	Cunha et al. [26]	Three-phase motor current signals	3 (current phases)	10,000 Hz	353 healthy files 2,264 fault files	Chapter 4
Broken Rotor Bar Dataset	Maciejewski et al. [27]	Three-phase motor current signals	3 (current phases)	55,611 Hz	80 healthy files 320 fault files	Chapter 4
Low-Frequency Broken Rotor Bar Dataset	Luong [28]	Three-phase motor current signals	3 (current phases)	1,000 Hz	134 normal samples 221 fault samples	Chapter 5
Bearing Fault Dataset	Jung et al. [29]	Motor vibration and current signals	Multiple	26,500 Hz	3 normal samples 33 fault samples	Chapter 5
High-Frequency Broken Rotor Bar Dataset	Treml et al. [30]	Three-phase motor current signals	3 (current phases)	55,611 Hz	80 normal samples 320 fault samples	Chapter 5

1.4.2 Dataset Characteristics and Relevance

Each dataset was selected to address specific research questions related to the overall goal of developing energy-efficient and generalisable approaches to IM condition monitoring. This section discusses the characteristics of each dataset and explains its relevance to the corresponding chapter's research focus.

Drone Control Dataset (Chapter 3)

The drone control dataset [24] consists of control signals (pitch, roll, and throttle) from both healthy and faulty drone operations. While not directly related to IMs, this dataset was strategically chosen to validate the fundamental capabilities of the Multi-Channel LSTM-Capsule Autoencoder for multivariate time series anomaly detection. The dataset's multivariate structure, with three control channels representing interconnected signals with complex spatial relationships, mirrors the three-phase currents in motor monitoring systems. Furthermore, the dataset contains well-defined anomalies that allow for unambiguous evaluation of detection performance. With 90,000 training samples, the dataset is substantial enough to train neural networks effectively while remaining computationally tractable, aligning with Green AI principles. The drone dataset serves as an excellent initial validation platform because it presents a clean test case for anomaly detection in multivariate time series data before proceeding to the more complex domain of motor fault detection.

SKAB Benchmark (Chapter 3)

The Skoltech Anomaly Benchmark (SKAB) [25] is a public benchmark specifically designed for evaluating anomaly detection algorithms. It consists of 35 data subsets from a water circulation system, with each subset containing measurements from 8 different sensors. This benchmark was selected because it enables standardised evaluation, allowing for direct comparison with state-of-the-art anomaly detection methods. The water circulation system exhibits complex behaviors and interdependencies among sensors, providing a challenging test for the proposed algorithm. Additionally, SKAB includes both outlier anomalies and changepoint anomalies, enabling comprehensive evaluation of the detection capabilities. Testing on SKAB helps establish the broader applicability of the proposed LSTM-Capsule architecture beyond specific industrial domains, demonstrating its effectiveness on standard anomaly detection tasks.

Motor Current Datasets (Chapter 4)

Chapter 4 focuses on the Dataset Fusion algorithm for homogeneous time series data. Two complementary motor current datasets were selected for this purpose. The Inter-turn Short Circuit Dataset [26] provides three-phase current data from a motor with inter-turn short circuit faults of varying severity, operating at 60Hz with 4 poles, 1HP mechanical power, 220V supply, and 3A rated current. The Broken Rotor Bar Dataset [27] contains three-phase current data from a squirrel cage AC motor with broken rotor bar faults of different severity levels, also operating at 60Hz with specifications similar to the first dataset.

These datasets were specifically chosen to test the Dataset Fusion algorithm because they are homogeneous in structure while representing different fault types, thereby creating an ideal testing scenario. The different sampling rates (10,000 Hz for the Inter-turn Short Circuit Dataset and 55,611 Hz for the Broken Rotor Bar Dataset) test the resampling capabilities of the fusion algorithm. Furthermore, the complementary nature of the fault information—one focusing on stator faults and the other on rotor faults—allows a combined model to potentially detect a wider range of fault conditions. Together, these datasets provide an ideal test case for evaluating whether the Dataset Fusion algorithm can effectively combine information from multiple homogeneous sources while preserving distinctive fault signatures from each source.

Multiple Motor Current Datasets (Chapter 5)

Chapter 5 addresses the challenge of generalising across heterogeneous motor datasets using the Order Domain Transformer (ODT). Three different motor current datasets were selected for this purpose. The Low-Frequency Broken Rotor Bar Dataset [28] provides data at relatively low sampling rates (1,000 Hz) with motors operating at both 50 Hz and 60 Hz. The Bearing Fault Dataset [29] contains data on various fault types including bearing faults, unbalance, and misalignment, sampled at a much higher rate (26,500 Hz). The High-Frequency Broken Rotor Bar Dataset [30] focuses on broken rotor bar faults with a high sampling rate (55,611 Hz) and different motor specifications.

This combination of datasets creates a challenging test environment for the ODT approach due to their heterogeneity. The datasets represent different motors with varying specifications, sampling rates, and operating frequencies, creating a demanding scenario for cross-motor generalisation. Together, they cover multiple fault types, enabling comprehensive testing of the ODT's ability to standardise fault signatures across different motors. The High-Frequency Broken Rotor Bar Dataset serves as an entirely unseen dataset for evaluating generalisation to new motors, which is a critical capability for practical industrial applications.

1.4.3 Dataset Size and Data Efficiency Considerations

A notable characteristic of the datasets employed in this thesis is their relatively modest size compared to typical deep learning applications. This characteristic aligns with the Green AI principles that emphasise data efficiency alongside computational efficiency. In real industrial settings, collecting large volumes of labeled fault data is often prohibitively expensive or impractical, as it may require deliberately damaging expensive equipment or waiting for rare fault occurrences. The datasets used in this research, while limited in size, contain high-quality, domain-specific information that is more valuable than larger volumes of less relevant data. This focus on quality over quantity is particularly important for specialised applications like motor fault detection.

The modest size of these datasets actually strengthens the evaluation of the thesis contributions, as it allows direct assessment of how the proposed methods perform under data-constrained conditions typical of real industrial environments. Working with these focused datasets enables more rapid experimentation cycles and reduces the environmental impact of the research, in accordance with Green AI principles. Chapter 4 specifically investigates how the Dataset Fusion algorithm performs with varying amounts of training data, demonstrating that good performance can be maintained even with significantly reduced data volumes. This finding has important implications for industrial applications, where minimising data collection requirements can lead to substantial cost savings.

The relatively small datasets used in this thesis thus serve not as a limitation but as an authentic representation of industrial conditions and an opportunity to demonstrate the data efficiency of the proposed approaches. This practical focus is essential for developing methods that can be readily deployed in real-world industrial settings.

1.5 Statistical Methodology for Experimental Evaluation

All experiments conducted in this thesis were initially analysed using statistical methods to determine the significance of the results. This is crucial for substantiating conclusions from the experimental data presented in this work.

1.5.1 Statistical Tests for Group Comparisons

When comparing performance metrics across multiple model variants, it is essential to determine whether observed differences are statistically significant or merely due to random variation. Two primary statistical approaches were considered for this analysis:

One-way Analysis of Variance (ANOVA)

ANOVA is a parametric statistical test commonly used to determine whether the means of three or more independent groups are significantly different [5]. The test evaluates the null hypothesis that all population means are equal against the alternative hypothesis that at least one mean is different. The ANOVA test is based on several key assumptions:

- The data follow a normal distribution within each group
- The groups have approximately equal variances (homoscedasticity)
- The observations are independent

The test calculates an F-statistic, which is the ratio of between-group variance to within-group variance. If this ratio is sufficiently large, the null hypothesis is rejected. The ANOVA source table, shown in Table 1.2, provides a structured way to present the degrees of freedom, sum of squares, mean squares, F-ratio, and associated *p*-value.

Table 1.2: One-way ANOVA source table, courtesy of [5], where: X = individual observation, $\bar{X}_j$ = sample mean of jth group, $\bar{X}$ = overall sample mean, k = number of groups, N = number of observations per group.

Source	**Degrees of Freedom (DF)**	**Sum of Squares (SS)**	**Mean Square (MS)**	**F-Ratio**	***p*-Value**
Between Groups	$DF_b = k - 1$	$SSB = \sum n_j(\bar{X}_j - \bar{X})^2$	$MSB = \frac{SSB}{DF_b}$	$F = \frac{MSB}{MSE}$	Right tail $F(DF_b, DF_e)$
Error	$DF_e = N - k$	$SSE = \sum\sum(X - \bar{X}_j)^2$	$MSE = \frac{SSE}{DF_e}$		
Total	$DF_t = N - 1$	$SST = \sum\sum(X - \bar{X})^2$			

Kruskal-Wallis H Test

Upon examination of the experimental data through boxplots, it became evident that the performance metrics did not consistently follow normal distributions when performing repetitions, which violates a key assumption of ANOVA. Therefore, the KW H test [31] was adopted as a more appropriate statistical method for analysis.

The KW H test is a non-parametric alternative to one-way ANOVA that does not assume normality in the data distributions. Instead, it compares the medians of multiple independent groups by first converting all values to ranks before analysis, making it robust against violations of normality and particularly suitable for analysing machine learning model performance metrics. These often exhibit non-normal distributions and heteroscedasticity due to the stochastic nature of model training and convergence characteristics.

The test evaluates whether samples originate from the same distribution (null hypothesis) or from different distributions (alternative hypothesis). The test statistic H approximates a chi-square distribution with k-1 degrees of freedom (where k is the number of groups).

As with ANOVA, the *p*-value of the scores is evaluated based on a 95% confidence interval, which is the conventionally accepted value in scientific research.

Therefore, if the p-value of an experiment is calculated to be <0.05, the null hypothesis is rejected, and it can be concluded that statistically significant differences exist between at least two groups.

The formulation and interpretation of the KW H test are summarised in Table 1.3.

Table 1.3: KW H test summary, where: R_i = sum of ranks in group i, n_i = sample size of group i, N = total number of observations, k = number of groups.

Test Statistic	Degrees of Freedom	Distribution	Significant if
$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1)$	$df = k - 1$	Chi-square distribution	p-value $< \alpha$ (typically 0.05)

1.5.2 Post-hoc Analysis

For experiments where the KW test indicates significant differences, post-hoc Mann-Whitney U tests with Bonferroni correction were performed to identify specific between-group differences. The Mann-Whitney U test (also known as the Wilcoxon rank-sum test) [32] compares the rank distributions of two independent samples without requiring parametric assumptions.

The Bonferroni correction adjusts the significance threshold (α/m, where m is the number of comparisons) to control the family-wise error rate during multiple comparisons, thereby mitigating Type I error inflation. This approach ensures that when multiple pairwise comparisons are made, the overall probability of incorrectly rejecting the null hypothesis remains at the desired level.

1.5.3 Presentation of Results

Throughout this thesis, the KW test results are presented in tables showing descriptive statistics for each group (sample size, mean, standard deviation, and median), along with the chi-square statistic (H) and its associated p-value, as shown in the example format in Table 1.4.

Table 1.4: Example format for KW test results.

Model	n	Mean	SD	Median	Chi-square	p-Value
Model A	5	0.XX	0.XX	0.XX		
Model B	5	0.XX	0.XX	0.XX	X.XX	0.0XX*
Model C	5	0.XX	0.XX	0.XX		
Model D	5	0.XX	0.XX	0.XX		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis reveals significant differences between [specific pairs].

Chapter 2

Literature Review

2.1 Fault Detection

This section investigates the development of fault detection techniques, from traditional signal processing and statistical analysis methods to more advanced machine learning-based approaches, particularly focusing on NN and their variants.

2.1.1 Traditional Fault Detection Methods

Fault detection systems have been researched and improved extensively over the last two decades due to the intense demand to automate industrial processes for Industry 4.0 [7]. There have been numerous approaches that aim to be effective in detecting different types of faults in different systems, due to the nature of the usage of the system or other reasons relating to the susceptibility of the system to certain faults. Some approaches for fault detection have involved using methods and techniques such as redundancy for sensors, sometimes paired with analytical redundancy methods [33] [34].

The common issue with these proposed solutions is that they involve the installation and maintenance of physical hardware to monitor the sensors or the system, which naturally means that they may require redundancy in more sensitive use cases: for example, that require the monitoring of life-threatening substances with very sensitive sensors. Furthermore, this guarantees an increase in the operating costs of these solutions due to increased energy usage and maintenance costs and provides another barrier to the goal of achieving automation. This has encouraged the development of soft sensing systems, which use the existing sensors in a system to infer further information regarding the system. This approach provides an economical and cost-effective alternative to physical systems by not needing to implement any additional physical hardware that could be expensive to buy or maintain whilst achieving robust fault detection scores that are comparable to and even better than physical systems.

Statistical analysis and signal processing are frequently used soft-sensing methods in fault detection [35][36][37][38], since they are able to overcome the drawbacks of physical hardware monitoring and provide a robust method of data inference. For instance, in [39] a dynamic model is proposed that is able to utilise the existing Supervisory Control And Data Acquisition (SCADA) system in wind turbines to dynamically model the relationship between the sensor readings by a parameter estimation process for the purpose of fault detection. A frequency domain analysis is used to determine damage sensitive indices, which are then compared to the model sensor. The technique is tested on a 5-year wind turbine dataset where the system was able to detect faults as well as perform fault prognosis. Whilst the method is clearly effective in the specified use case, the flexibility of the method for other use cases comes into question as in-depth knowledge about the system and the relationships between the variables being analysed was utilised to be able to create the initial model. This issue is also mentioned in [40], where the authors concluded from their survey of outlier detection techniques that model-driven methods are heavily dependent on the understanding of the data being analysed. The lack of flexibility of such techniques is also mentioned, due to the heavy tailoring that must be made to the models for each dataset. This is a trend across many signal processing techniques, including for motor condition monitoring, where Gangsar [41] noted, in their review of state-of-the art outlier detection techniques, the lack of flexibility of data analysis techniques such as acoustic analysis and MCSA in detecting a wide range of faults that could occur within the system.

2.1.2 Machine Learning for Fault Detection

More recently, ML has been heavily utilised in literature for the modelling of such systems. As well as being a soft sensing technique, ML is able to provide a higher degree of flexibility in terms of application as well as being generally easier to implement than the aforementioned techniques. Specifically, NNs have been identified as an effective tool in data analysis and fault detection due to their unique ability to be trained to identify numerical relationships in different forms of data [42]. They provide an advantage over traditional signal processing and statistical techniques because of the level of complexity with which they can model the data, as well as being generalisable to similar types of data.

ML techniques are being applied to a wide range of fault detection tasks across different domains. In the context of IMs, Shubita et al. [43] proposed a fault diagnosis system based on acoustic emission using ML techniques. The authors aimed to achieve stable accuracy and real-time performance with a simple model for edge ML by using a minimal amount of data and extracting relevant features from the raw data. However, they noted that before feature extraction at the preprocessing stage, many new methods can enhance useful data embedded in the noise signal to improve the model's overall accuracy. Benninger et al. [44] developed a fault detection method for industrial applications that requires little prior knowledge of the motor and low measurement effort. Their approach combines analytical modeling using a multiple coupled circuit model and a feed forward NN, with parameter identification based on easily obtained data. While the method can effectively represent the behavior of the monitored motor, it has limitations such as the use of synthetic data, which may not accurately represent the dymanics of real-world data, and the inability to detect bearing faults.

In the domain of photovoltaic systems, Voutsinas et al. [45] employed logistic regression enhanced with Cross-Validation for fault detection. Their approach offers low computational cost, but the authors grouped unknown faults into a separate category, limiting the specificity of the method. Sahoo et al. [46] proposed an Online DNN algorithm to detect, classify, and predict both symmetrical and unsymmetrical faults occurring on transmission lines using the RMS values of voltages and currents during fault. Their approach utilises raw TS data, which is advantageous for fault detection as it requires significantly less domain expertise to leverage.

As shown, several studies in the literature use NN models for fault detection across various use cases and datasets [47] [48]. However, generalisation remains a challenge for NNs. The main issue found in literature with NNs is the importance of data volume and representation in being able to train NNs effectively. Most NN architectures such as the Convolutional Neural Network (CNN) and the LSTM require large quantities of data to effectively learn the shape and features of the data, and some methods even require the labelling of the data before training, known as supervised learning [49], which is very time-consuming and costly as this is usually a manual process. Furthermore, with some types of data it is very difficult to distinguish faults and anomalies in raw sequential format. The challenges associated with data volume, representation, and the difficulty in distinguishing faults and anomalies in raw sequential data highlight the need for advanced techniques that can effectively learn from limited data and capture complex relationships.

2.1.3 Combining Signal Processing and Machine Learning

To address the challenges associated with NNs, researchers have explored the combination of signal processing techniques with NNs to leverage the advantages of both approaches. Signal processing techniques can enhance data representation, while NNs provide flexibility and ease of use. This hybrid approach has demonstrated significant success in various fault detection tasks, particularly in the domain of IM fault detection [50][51]. In these cases, frequency domain transforms were employed to improve the representation of data for use with LSTM networks.

Several recent studies have investigated the integration of signal processing techniques with machine learning for fault detection across different domains. Cano et al. [52] proposed combining the Discrete Wavelet Transform (DWT) with radial basis function neural networks (RBFNN) for fault detection in microgrids. The authors found that this integration enhances accuracy in fault detection, especially in scenarios involving nonlinear elements such as photovoltaic, hydrokinetic, and variable electric load systems. However, their approach does not address high-impedance faults. Luo et al. [53] introduced a transformer framework based on the Fast Fourier Transform (FFT), called FFT-Trans, for mechanical fault diagnosis. The proposed framework uses a learnable filter in the form of CNN layers and excels in high noise conditions, but it focuses on a single dataset and the mechanical fault category.

In the context of IMs, Das et al. [54] proposed a custom CNN for stator winding interturn fault severity classification under variable load conditions. The authors use Park's vector modulus representation for the 3-phase current data. While their approach achieved promising results, the computational cost during the training phase is slightly higher compared to existing approaches. Fu et al. [55] developed a bearing fault diagnosis method based on wavelet denoising and machine learning, testing both unsupervised and supervised learning methods. They found that decision trees performed best among supervised methods, while K-means clustering was the best unsupervised approach. However, their study focused on specific types of bearing faults, limiting the generalisation ability of the model.

In addition to combining signal processing with ML, researchers have also explored the hybridisation of different NN architectures to capitalise on their strengths for various data types. The use of Capsule Network (CapsNet) has been found to improve training and classification performance on smaller datasets [56] [57]. Furthermore, the hybridisation of Recurrent Neural Network (RNN)s and CNNs has gained popularity for TS data analysis [49].

The combination of signal processing techniques with machine learning, as well as the hybridisation of different NN architectures, has shown promising results in addressing the limitations of traditional approaches and improving fault detection performance. These hybrid methods have the potential to enhance data representation, capture complex temporal dependencies, and achieve better generalisation on smaller datasets. The following sections will further explore specific NN architectures and their applications in fault detection.

2.1.4 Recurrent Neural Networks for Time Series Analysis

Recurrent Neural Networks (RNNs) have emerged as a popular choice for TS analysis due to their ability to identify dependencies in sequential data using their internal memory [58, 47]. Among RNN variants, the LSTM network [1] has gained significant attention for its ability to overcome the vanishing gradient problem [59] encountered by traditional RNN architectures. This capability allows LSTMs to learn long-term dependencies in TS data more effectively, making them particularly suitable for various applications in commercial and industrial environments.

Recent research has explored the application of LSTM networks in diverse fault detection scenarios. Lee et al. [60] focused on gearbox fault detection in plastic extruder machines, where traditional vibration measurements can be hindered by excessive noise from defective gearboxes. They proposed an AE-LSTM outlier fault detection technique using vibration and thermal data, employing Fast Independent Component Analysis (FastICA) to blend selected features into a single-dimensional representation. While their approach showed promise, it was limited to clear and obvious faults and relied on invasive measurement techniques. For petroleum production forecasting, Kumar et al. [61] demonstrated the advantages of combining attention mechanisms with LSTM networks. Their A-LSTM network showed improved accuracy and flexibility in handling data with or without noise. However, the study was limited to univariate time-series forecasting, which is a simpler task compared to multivariate analysis. Wen et al. [62] also explored the combination of LSTM with attention mechanisms, using two LSTM models as encoder and decoder with an attention mechanism between them. This approach demonstrated advantages in predicting TS with larger time steps.

Alikhani et al. [63] proposed a long short-term memory regulated deep residual network for data-driven fault diagnosis in electric machines. While their approach showed promising results, it did not account for real-world data dynamics such as noise, which could limit its practical applicability. An interesting application of LSTM networks in anomaly detection was explored by Provotar et al. [64], who utilised LSTM layers in an Autoencoder architecture. The authors recognised the limitations of many current machine learning methods that require labeled data, which is often impractical for large volumes of TS data. They also highlighted the shortcomings of classical anomaly detection methods like Support Vector Machines and Isolation Forests in accounting for the temporal aspects of TS data. While their approach showed promise in terms of detection accuracy and wide applicability, it had some limitations. The authors selectively used data that was easily detectable by the autoencoder and did not explore the sensitivity of detection in depth. Additionally, they assumed that the training data was free of anomalies, which may not hold true in real-life scenarios.

Despite the advantages of LSTM networks, they face challenges such as overfitting when used with gradient descent learning optimisation algorithms. Addressing this issue often requires careful hyperparameter tuning, which can be time-consuming and inefficient, especially for complex datasets.

2.1.5 Convolutional Neural Networks and Capsule Networks

CNNs have demonstrated significant success in image classification tasks [65, 66], and recent studies have extended their application to TS forecasting and outlier detection [67, 68]. The hybridisation of CNN and LSTM layers has shown empirical evidence of improving outlier detection performance [69, 49]. Several recent works have explored this hybrid approach in various contexts.

Borre et al. [70] proposed a hybrid CNN-LSTM attention model using quantile regression for fault detection. Their approach used vibration sensor data measured in three axes (axial, radial, and radial X). However, the study was limited to a single modality of data, and the use of vibration sensors can be invasive and challenging to implement at scale. Zhang et al. [71] developed a vibration data anomaly detection method based on combined CNN and LSTM, incorporating multi-feature extraction techniques such as residual analysis and power spectral density. While their approach showed promise, it was also limited to vibration data. Kim et al. [72] introduced an unsupervised prediction-based time-series anomaly detection method, using a prediction model consisting of an encoder with multiple Transformer encoder layers and a decoder including a 1D convolution layer. However, their study was limited to univariate data.

Despite the success of CNNs, they have a well-documented fundamental flaw in understanding spatial context in data and suffer from information loss due to the pooling layer, which is particularly evident in image classification tasks. To address these limitations, Sabour et al. [20] proposed the Capsule Network (CapsNet). CapsNets "encapsulate" the entity being described in a vector format, where the vector length represents the probability of existence, and its orientation describes the entity's characteristics, such as spatial context. CapsNets have shown state-of-the-art performance in various image classification tasks, including brain tumor classification using MRI images [73] and hyperspectral image classification [74]. Variants of CapsNets have been developed to use gradient descent instead of the dynamic routing algorithm [57], further improving their performance and efficiency.

The combination of CapsNet models with LSTM models has also shown effectiveness in applications such as transportation network forecasting [75]. However, the application of CapsNets to TS data, particularly in its raw format, remains limited [76]. Most approaches using CapsNets for TS data convert the data into an image representation, as CapsNets have demonstrated improved performance in this context. For instance, Fahim et al. [77] proposed an approach using capsule networks to detect short circuit faults in power network transmission lines, using a Gramian Angular Field (GAF) representation [78].

The limitations of CNNs and the promising results of CapsNets in various domains highlight the potential for developing novel architectures that can effectively capture spatial and temporal relationships in multivariate TS data.

2.1.6 Autoencoder-based Anomaly Detection

The field of anomaly detection has witnessed a growing trend towards the use of Autoencoders [3], which offer the flexibility of NNs while maintaining simplicity of application and reducing the need for large volumes of labeled data. This shift has led to several innovative approaches that address various challenges in anomaly detection across different domains.

Shen et al. [79] proposed a recurrent autoencoder with multi-resolution ensemble decoding to overcome overfitting and capture patterns at different resolutions. While their model outperformed competing approaches on diverse datasets, including EEG, power consumption, and website traffic data, it showed room for improvement in certain scenarios, particularly in terms of model sensitivity and performance metrics.

Kieu et al. [80] introduced a variational quasi-recurrent autoencoder and its bi-directional variant, aiming to enhance computational efficiency and generalisation ability through improved latent space learning. Their bi-directional model demonstrated superior performance across five different domains, although it did not address the detection of non-statistically separate anomalies.

In the industrial context, Radaideh et al. [81] developed a hybrid LSTM and Convolutional Autoencoder for anomaly detection in high voltage converter modulators, achieving high precision and recall scores. However, the known limitations of Convolutional Neural Networks suggest that incorporating Capsule Networks could potentially enhance these results further.

Recent works have explored autoencoder-based approaches in various applications. Torabi et al. [82] investigated anomaly detection in cloud network data, introducing a novel method for calculating residual error and determining threshold values. While promising in experimental settings, this approach still faces challenges in practical applications. Yan et al. [83] proposed an autoencoder for unsupervised anomaly detection in machine tools under noisy conditions, validating their approach with real CNC machine tool data. However, the generalisability of this method remains to be proven across different types of machinery and operating conditions.

For video surveillance, Mishra et al. [84] developed an unsupervised mechanism for anomaly detection using a novel combined regularity score-based thresholding mechanism. Their use of convolutional networks for video processing suggests potential for improvement through the incorporation of Capsule Networks, albeit with potential trade-offs in real-time performance.

Khalid et al. [85] addressed the challenge of detecting subtle anomalies in turbine shaft behavior of power plants using deep LSTM autoencoders. Their work highlights the importance of handling anomalies that fall outside the input space of the training data, a common challenge in real-world applications.

Xie et al. [86] proposed a hybrid model combining convolutional autoencoders, convolutional LSTM, and variational autoencoders for anomaly detection in multivariate sensor data. While showing strong performance, this approach requires further validation on real-world datasets to establish its practical efficacy.

Wei et al. [87] developed an LSTM-AE-based hybrid deep-learning technique for detecting contextual anomalies in indoor air quality datasets, applying their model to real-world CO2 time-series data. However, their focus on univariate data limits the applicability to more complex, multivariate scenarios common in industrial settings.

Chen et al. [88] introduced an LSTM-based semi-supervised variational autoencoder for anomaly detection in cloud environments, highlighting the need for auto-calculating thresholds based on data trends to enhance adaptability.

The diverse applications and approaches in autoencoder-based anomaly detection underscore both the potential and the challenges in this field. While these methods have shown promise in various domains, they often face limitations in terms of generalisability, handling multivariate data, and adapting to real-world conditions.

2.2 Generalisation techniques

This section investigates the current literature on different methods of addressing NN generalisation performance, as well as recent works using multiple datasets in different domains and tasks.

2.2.1 NN-based generalisation techniques

NN based generalisation techniques are methods designed to improve a model's ability to perform well on unseen data, beyond the specific examples it was trained on. These techniques primarily focus on modifying the network architecture, training process, or loss function to enhance the model's capacity to learn generalisable features and relationships. The following sections outline some existing strategies proposed in literature that have empirically proven to be effective in tackling the generalisation problem.

Dropout

Dropout [89] is a regularisation technique that can be used to prevent the neural network from overfitting on the training data. This is done by ignoring several randomly selected neurons, with the number of neurons dropped dependent on the "dropout rate" parameter, so they do not affect the output of the neural network on the forward pass. The idea behind dropout is to effectively train many subnets in a given network so that the network acts as a sum of many smaller networks that can learn the representation of the data without the presence of the dropped-out neurons. This was found to improve the generalisation performance of the network by reducing data overfitting.

Pruning

Pruning is a process whereby a NN selectively removes trainable parameters based on an established criterion with the aim of maintaining the performance of the NN [90]. There are two main categories of pruning: Structured and Unstructured pruning. Unstructured pruning directly removes trainable parameters from the network, such as connections to neurons (weights). Structured pruning involves removing entire structures from the network such as neurons and filters. Structured pruning allows for a faster computational time in relation to unstructured pruning, as most frameworks existing for ML do not allow for the acceleration of sparse matrix calculations, therefore the NN will be able to reduce the number of calculations for the former but not the latter.

Various criterion has been established in literature for the pruning of NNs. One primitive but popular method is known as the weight magnitude criterion. The criterion dictates that the weights of the smallest magnitude are removed. The idea behind this is to remove all the weights that contribute the least to the function as they are less likely to impact the final prediction.

The most established framework for pruning is known as the train, prune and fine-tune method [90]. As the name suggests, the model is first trained, then iteratively pruned and fine-tuned based on the weight magnitude criterion. More recently however, an increasing number of works [91] [92] [93] have evolved this framework with novel methodology that has allowed for further reduction of NN parameters and hence more efficient training and computation whilst maintaining similar performance.

L1 and L2 regularisation

L1 and L2 regularisation are widely used techniques to improve the generalisation performance of neural networks. These methods add a penalty term to the loss function, discouraging the model from relying too heavily on any individual features or weights. Drucker and Le Cun introduced the concept of double backpropagation, which involves adding a term to the objective function that penalises large derivatives of the output with respect to the input. This approach was shown to improve generalisation performance by making the network less sensitive to small perturbations in the input [94].

L1 regularisation, also known as Lasso regularisation, adds the absolute value of the weights to the loss function. L2 regularisation, or Ridge regularisation, adds the squared magnitude of the weights. The choice between L1 and L2 can have significant implications for model behavior and performance.

Ng demonstrated that L1 regularisation can be particularly effective for feature selection. His work showed that when using L1 regularisation, the sample complexity grows only logarithmically with the number of irrelevant features. This makes L1 regularisation especially useful in high-dimensional spaces with many irrelevant features [95]. In contrast, L2 regularisation tends to spread out the weight values more evenly and is less likely to result in sparse models. It's often the default choice for many applications due to its ability to prevent any single feature from dominating the model's decisions.

Ensemble Networks

Ensemble networks represent a powerful approach to improving the generalisation ability of neural networks. This technique, which involves combining the predictions of multiple classifiers into a single, unified classifier [96], typically demonstrates superior performance compared to any of its individual constituent models.

The enhanced generalisation capability of ensemble networks stems from several key factors. Primarily, ensembles reduce variance in output through averaging, which relates to the bias-variance tradeoff that will be explored in more depth in the subsequent section. Furthermore, ensemble methods contribute to error reduction and increased robustness. The diversity in models and their training processes often leads to convergence at different local minima, thereby capturing various aspects of the data distribution. This diversity helps to mitigate overfitting by individual models, as their potential biases tend to average out across the ensemble [97].

Several types of ensemble methods have been proposed and widely adopted in literature. One such method is bagging, or Bootstrap Aggregating, introduced by Breiman [98]. Bagging techniques combine multiple models trained on different subsets of a given dataset, helping to reduce variance and overfitting by aggregating predictions from models trained on various data samples.

Another prominent ensemble method is boosting, proposed by Freund and Schapire [99]. This approach involves training models sequentially, with each iteration focusing on improving the model based on the errors made in its previous form. Boosting algorithms are particularly effective at reducing bias and can create strong predictors from relatively weak learners.

Building upon the concept of boosting, Chen and Guestrin developed XGBoost [100], a modern and highly efficient implementation of gradient boosting. XGBoost has gained significant popularity in machine learning competitions and practical applications, renowned for its computational efficiency and achieving state-of-the-art performance on a wide range of tasks [101][102][103]. However, dataset noise significantly impacts XGBoost's performance, as well as boosting algorithms in general.

2.2.2 Data-based generalisation techniques

Data-based generalisation techniques are largely overlooked in deep learning in comparison to NN-based techniques. This is because the NN structure and learning optimisation algorithms are usually the reason for such weak performance, so improvements can largely be made by improving and optimising how the NN learns as opposed to what the NN is learning. However, it is still important to consider the training data as it can be a bottleneck for learning ability if not composed in the correct manner [104].

An important aspect to consider is the difference in the data distribution between the data used to train the NN and the data that the NN will eventually be applied to. Often the data picked for training is not fully representative of the true distribution of the dataset, which creates a bottleneck to generalisation as the NN is not prepared for the distribution found in the overall population since it has been tuned to the distribution of the training sample. Shuffling the data before taking the training sample often helps with this to increase the likelihood of the sample representing the true distribution. Furthermore, shuffling during training also helps the NN weights to escape local minima and converge towards the global minimum of the function. Many works in deep learning falsely assume that the data distribution is static, which is largely incorrect as in practice data distributions are generally dynamic and tend to shift away from the test data distribution with real-world data; this phenomenon is known as distribution shift. This shift has been detected and quantified in recent works [105] [106], and accounted for with online adaptation to the shift [107], which allows the NN weights to adjust themselves to account for this distribution shift.

Data with excessive noise can inhibit the NNs ability to learn the true input-output function required by a specific application. Mathematically, this can be explained by the bias-variance decomposition [108], specifically the variance component. The variance value refers to the change in prediction accuracy over different subsets of the data. A high variance value indicates that the NN has learnt the noise in the data, or in other words overfit on the training data. The most common method of reducing the variance is increasing the volume of the training data, which will naturally bring the distribution of the data closer to the required distribution that represents the overall dataset as opposed to just the subset of training data. As well as using real-world data, data augmentation has also proved to be an effective method of increasing data volume using data that is already accessible [109] [110]. Another effective method that is commonly used is the denoising of data [111]. This also contributes to the reduction of the variance: By removing the noise in the data, the training data subset will be cleaner and will more accurately represent the true distribution of the dataset. Whilst both methods are generally proven to work, there are still major issues with both methods that are still being addressed in literature, such as effectiveness in a real-world situation.

The financial and computational costs associated with increasing the volume of training data are not just substantial but also rapidly increasing [112]. This makes it increasingly difficult to train models without significant resources. Consequently, only well-funded entities with considerable resources can achieve a performance boost using this method, as their computational power and finances typically surpass those of other research entities and companies. This disparity creates a bottleneck for smaller entities, hindering their competitiveness in the field [113].

Denoising is still a very active field of research due to the many limitations of the currently proposed techniques. Since it is very difficult to determine which aspects of the data features are representative of the true distribution [111], it is very difficult to use denoising techniques such as filtering since features that are potentially important to NNs could be filtered out, limiting the potential performance of the NN. Furthermore, filtering techniques are largely contextual, so it is very difficult to develop filtering methods that work across multiple contexts to the same level of effectiveness.

Transfer learning [114] is also widely regarded in literature as a robust method of improving performance by utilising data from a different set but in the same domain as the target data. Whilst there have been many recent works regarding transfer learning [115], there is a gap in research concerning dataset fusion methods, an alternative to transfer learning that utilises multiple datasets in the same training phase, as opposed to multiple training phases to train a model more robust to the difference in data distributions between the training data and the data encountered during operation. This will be further explored in the present work.

2.2.3 Multi-Dataset training

Recently, researchers have identified the potential benefits of training generalisable NNs using multiple datasets to enhance performance and expand the capabilities of the NN models. Many of these approaches primarily focus on image-based applications [116] [117]. For example, Yao et al. [118] introduced a novel framework for cross-dataset training, leveraging pre-existing labels from multiple datasets to create a single model capable of detecting the union of labelled features from all contributing datasets. This approach aims to maximise the utility of available labels for distinct classes in each dataset, thereby circumventing the time-consuming and resource-intensive process of labelling a single dataset with new classes that are already present in another dataset. Empirical evidence provided by the authors demonstrates the effectiveness of this approach when applied across multiple datasets, achieving comparable performance levels without sacrificing accuracy. In a related study, Zhou et al. [119] introduce a technique for training a unified object detection model on several extensive datasets by using dataset-specific training methods and losses, but maintaining a shared detection architecture with outputs specific to each dataset. This approach circumvents the necessity for manual taxonomy alignment, as it automatically combines the outputs of different datasets into a unified semantic taxonomy. The authors show that this multi-dataset detector achieves comparable performance to dataset-specific models in their respective domains while effectively generalising to unseen datasets without the need for fine-tuning. By utilising multiple training datasets, these approaches can lead to reduced resource requirements in terms of training data and improved performance. However, it is worth noting that they do not specifically address the aspect of reducing computational power during the training process.

2.2.4 Domain Adaptation

Domain adaptation techniques are designed to enable models to leverage common features between different but related datasets, while also accommodating the unique characteristics of each domain [120]. Techniques such as Domain-Adversarial Neural Networks (DANN) [121] and Domain Adaptive Faster R-CNN [122] have shown success in allowing models to generalise across domains with slight non-homogeneity. These models identify both domain-specific and domain-invariant features, facilitating training on both source and target domains. Furthermore, they show considerable strength in dealing with the domain shift from source to target domains, which is a common issue when integrating datasets from different contexts or applications. However, while these techniques offer significant advancements in handling domain shifts, they do not explicitly address the distribution characteristics of the initial source domain, an aspect which the present study aims to explore and address using the proposed method.

2.3 Cross-Domain Fault Diagnosis

The field of motor fault diagnosis has witnessed significant advancements with the development of various computational techniques, from traditional signal processing to modern machine learning algorithms. This section explores the most recent and relevant methods and dissects the limitations which Chapter 5 aims to address.

2.3.1 Order Tracking for Fault Diagnosis

Order Tracking (OT) techniques have long been employed in rotating machine fault diagnosis to analyse signals in relation to the rotating speed of the machine. OT is defined as a signal processing technique that transforms a signal from the time domain to the angular, or order, domain. It uses resampling to match the response to the angular position of the rotating component.

Traditional OT methods often relied on tachometers for accurate speed measurements. Cheng et al. [123] proposed an approach targeting the modulation feature of gear fault vibration signals in run-ups and run-downs. They combined OT with Local Mean Decomposition (LMD), a self-adaptive time-frequency analysis method. LMD overcomes issues found with conventional demodulation approaches such as Hilbert transform, which struggle with multi-component AM-FM signals and suffer from window effects. The authors demonstrated that their approach could identify gear status - with or without fault - accurately and effectively.

Li et al. [124] presented a method for fault diagnosis of rolling bearings based on computed order tracking, empirical mode decomposition (EMD), and Teager Kaiser energy operator. Using computed order-tracking, they transformed non-stationary vibration signals during run-up of bearing faults from the time domain into stationary signals in the angle domain. The EMD method allowed them to decompose the resampled vibration signals into intrinsic modes, providing a better understanding of the fault information contained in the vibration signal.

Recent advancements have explored tacholess methods and complex signal transformations to adapt to varying operational conditions and fault types. Lu et al. [21] provide an extensive review of tacholess speed estimation methods for OT. The authors categorise these methods based on the signal source for speed estimation: vibration or sound signals, electrical motor current signals, and video signals. While these tacholess OT methods offer flexibility in fault diagnosis without a tachometer, they often rely on complex signal processing techniques.

Sapena-Bano et al. [125] introduced a novel approach that transforms complex 3-D spectrograms into simplified x-y graphs for fault diagnosis in IMs. They extended the harmonic OT approach to non-stationary conditions, providing unique fault patterns that are easy to identify. Akar [126] presented a new method for detecting static eccentricity faults in inverter-driven IMs using Angular Domain OT (AD-OT). The method proves to be efficient for varying operating conditions and offers the advantage of leveraging existing sensors in inverter systems. However, this work focuses solely on static eccentricity faults and does not address the generalisability across multiple motor types or varying speeds.

Wang et al. [127] proposed a new simple and effective vibration OT method for bearing fault diagnosis of variable-speed direct-drive wind turbines. Their method uses the generator stator current signal as a resource to provide information about the shaft speed for the order tracking of the vibration signal. By specifying a reference shaft rotating frequency for order one, they converted the order-domain spectrum of the resampled vibration envelope signal into a frequency-domain power spectrum. This approach resolved the smearing problem caused by shaft speed fluctuation, allowing for clear identification of bearing fault characteristic frequencies.

Advanced OT methodologies have evolved, with the Vold-Kalman Filter (VKF) being a notable technique [128]. VKF enables the extraction of non-stationary periodic components from signals using a predefined frequency vector. The algorithm is formulated as a least-squares problem, allowing its implementation as a sparse linear system. VKF shares conceptual similarities with the Kalman filter, as it operates by minimising a cost function [129].

Pan [130] implemented an adaptive VKF OT approach to address the computational challenges of previous VKF OT schemes, which were not suitable for real-time applications due to their time-consuming nature. While the proposed technique is not exactly accurate, the decrease in computation time to make it suitable for real-time processing can justify the reduced accuracy.

Zhao et al. [131] applied VKF OT to analyse the vibration mechanisms of healthy motors, as well as those with inner ring faults, eccentricity faults, and compound faults. Their work demonstrates the versatility of VKF OT in diagnosing various fault conditions.

It's worth noting that OT techniques face limitations in non-stationary conditions. As shown in the literature, determining the angular velocity is challenging without specialised hardware, but various approaches have covered different strategies to address these issues. Furthermore, whilst existing literature present significant advancements in OT techniques for fault diagnosis, there is a notable gap in the literature regarding the combination of these approaches with intelligent fault diagnosis methods, such as using NNs. This presents an opportunity for future research to leverage the strengths of both OT and machine learning techniques for more robust and adaptable fault diagnosis systems.

2.3.2 Cross-Domain Fault Diagnosis

In recent times, there has been an increased volume of research into cross-domain fault diagnosis methods in an effort to increase detection accuracy and computational efficiency. This section explores several key contributions to this field, highlighting their innovations and limitations.

Chen et al. [132] developed a novel Domain Adversarial Transfer Network (DATN) approach to tackle the large domain shift problem in fault diagnosis. Their method introduces asymmetric deep encoder networks and domain adversarial training to adaptively learn discriminative representations. While this approach demonstrated improved accuracy compared to CNN baselines, it faced limitations in terms of high computational cost during the training stage and a lack of adaptability to new classes in the target domain.

Li et al. [133] proposed a novel deep learning-based method to address the partial transfer learning problem in machinery fault diagnosis. They adopted a class-weighted domain adversarial neural network for partial domain adaptation. This work is particularly noteworthy as it challenges the common assumption of identical label spaces between source and target domains. The authors argue that in real industrial settings, testing data often contains only a subset of the source label space, motivating the need for transferring diagnosis knowledge from a comprehensive source domain to a target domain with limited machine conditions.

Zheng et al. [134] introduced a diagnosis method for rolling bearings that explores more challenging but practical cross-domain scenarios. Their approach considers multiple source domains with potential discrepancies and assumes that only normal samples are available in the target domain during model training. A key innovation in their work is the use of a priori diagnosis knowledge in the signal preprocessing steps, designed to eliminate potential discrepancies among different domains and construct consistent-meaning inputs for the domain generalisation network. This preprocessing includes normalisation, angular resampling based on known fault frequencies, and envelope computation with Hilbert transform. However, a limitation of this method is the requirement for existing "normal" data in the target domain.

Xiao [135] proposed a feature adaptive motor fault diagnosis method using transfer learning to address real-world conditions where signals are unlabelled and operating conditions are unstable. The technique employs a CNN base with maximum mean discrepancy (MMD) regularisation to reduce distribution mismatch between source and target domains. While this approach shows clear performance improvements on unlabelled target domain data, it suffers from lengthy training times and a complex architecture.

Altaf et al. [136] presented a method for monitoring the dynamic behaviour of individual motors through a powerline network, using a mathematical model to analyse signal attenuation for estimating fault origin and type. Their approach uses FFT for spectral analysis and successfully localises motor faults within the network. However, the fault diagnosis process is not fully automated, and the method was not tested on high-power motors, limiting its applicability to real-world scenarios.

Liu et al. [137] introduced a Deep Adversarial Domain Adaptation framework to address the domain shift problem in rolling bearing fault diagnosis. Their model, which uses a Stacked Autoencoder (SAE) and a label classifier, effectively extracts features from vibration data and identifies faults across different operational conditions. However, the study does not explore the method's generalisability to other types of faults.

Chao et al. [138] proposed a method for online domain adaptation in rolling bearing fault diagnosis, specifically addressing imbalanced cross-domain data. Their approach adapts to gradually increasing target domain data using deep transfer learning and an adaptive network-based fuzzy inference system (ANFIS). A limitation of this method is its assumption of well-labelled data from a single source for initial training, which may not be feasible in many real-world applications.

Zhang et al. [139] addressed the challenges of fusing heterogeneous data from multiple sensors with their low-pass pyramidal ratio-color symmetric dot pattern (RP-CSDP) framework. This approach demonstrates rapid adaptability to new working conditions and fault types, even with limited training samples, and offers reasonable computational efficiency. However, the scarcity of vibration data in target domains may limit its real-world applicability. Additionally, the method's cross-motor generalisation capability remains unexplored, as it was not tested on data from different motors.

These studies collectively highlight the ongoing challenges and advancements in cross-domain fault diagnosis. While significant progress has been made in addressing domain shifts, data scarcity, and computational efficiency, there remains a need for methods that can generalise across different types of faults and motor configurations without relying on extensive labelled data or specific domain knowledge.

2.4 Conclusion

2.4.1 Fault Detection

It is clear from the presented literature that soft sensing methods of anomaly detection are more efficient and effective methods than hardware redundancy. However, with traditional methods it is difficult to accurately model systems without in depth knowledge of their dynamics and parameters, creating a barrier to flexible and accurate system modelling, which is the basis of many anomaly detection systems.

However, NNs provide a solution for this issue, providing a method of easily modelling system behaviour based on previously encountered data. Using NNs such as LSTM NNs for TS data learning, researchers have been able to accurately account for long term dependency in temporal data and create robust anomaly detection systems. However, this creates another issue with requiring access to large amounts of labelled data, which is expensive and time consuming to produce. To avoid labelling data, some literature have proposed unsupervised learning techniques such as the autoencoder, which is able to learn data features by transforming it into a latent space representation. However, a large volume of data is still required to utilise this technique, and generalisation performance is weak in many these methods. The Capsule was proposed to address issues with training efficiency and the shortcomings of traditional NNs with learning spatial context of data. Chapter 3 further explores these qualities found in Capsules by hybridising them with LSTMs in a NN, and addresses issues found with learning multivariate data with single channel NNs.

2.4.2 Generalisation techniques

Previous works applying multi-dataset utilisation reveal clear benefits, such as reduced dataset labelling requirements and increased generalisation performance. However, there is a lack of exploration in time-series-based multi-dataset models due to challenges in fusing sequential data from different sensors and collection specifications. Chapter 4 aims to address these points by proposing a novel approach for effectively combining raw time-series data from multiple sources. Furthermore, Chapter 4 will consider computational efficiency, as training on multiple datasets may require additional computational resources.

2.4.3 Cross-Domain Fault Diagnosis

Recent advancements in cross-domain fault diagnosis have focused on improving the accuracy and efficiency of fault detection across different domains, with OT methods and deep learning-based approaches showing promise in addressing the domain shift problem. However, these methods often rely on complex signal processing techniques, assume the availability of well-labeled data from a single source, or may not effectively address the generalisability across multiple motor types or varying speeds. Chapter 5 differentiates itself from the current OT field through a novel application of cross-motor generalisation through deep learning, as opposed to time-varying signal tracking. Furthermore, the method proposed in Chapter 5 seeks to address the limitations of the existing methods in the literature presented, particularly in the context of computational and data efficiency.

Chapter 3

Multi-Channel LSTM-Capsule Autoencoder Network for Anomaly Detection on Multivariate Data

3.1 Introduction

3.1.1 Motivation and Incitement

TS data analysis is a prominent field of research due to the significant demand stemming from increasingly larger datasets being acquired in industrial and commercial environments. The automation of this analysis has been integral to the advancement of Industry 4.0 [7]. One important use case for TS analysis is outlier detection, which is an important part of the function of intelligent systems in the context of fault diagnosis. There have been numerous approaches that aim to be effective in detecting different types of faults in different systems, due to the nature of the usage of the system or other reasons relating to the susceptibility of the system to certain faults. Some approaches for fault detection have involved using methods and techniques such as hardware-based redundancy for sensors, sometimes paired with analytical redundancy methods [34][33][140].

However, with the rapid advancement of Industry 4.0, there is an increasing demand for more effective and efficient anomaly detection techniques to monitor industrial machinery and systems. These techniques are crucial for maintaining the reliability and safety of industrial processes, as they enable the early identification of potential faults and anomalies in machinery and systems. In recent years, NNs have emerged as a powerful tool for anomaly detection, particularly in the context of multivariate TS data.

Despite the success of NNs in various domains, their application to multivariate TS data poses several challenges. One of the main issues is the need for large amounts of labeled data for training, which can be time-consuming and costly to obtain in industrial settings. Additionally, existing NN architectures often struggle to capture the complex temporal dependencies and spatial relationships present in multivariate TS data, leading to sub optimal performance in anomaly detection tasks.

3.1.2 Major Contribution and Organisation

To address these challenges, this chapter proposes the hybridisation of the CapsNet and the LSTM network in a novel multi-channel input Autoencoder architecture for use on raw TS data. The contributions of this chapter are summarised as follows:

- A Novel Hybridisation of the LSTM and Capsule layers is proposed using LSTM layers as encoders and Capsules as decoders;
- The hybridisation is implemented in a novel multi-channel input, merged output model architecture for use on raw multivariate TS data;
- The model is tested on a real-world dataset and bench-marked on another real-world dataset against prominent detection methods in the field.

This chapter is organised as follows: Background theory relating to the proposed method will first be provided. Following this, the proposed method will be introduced. The experimental design on the first dataset used, a drone dataset, will be briefly explained, then the experimental work completed on the dataset will be presented and analysed. The second dataset, a benchmark motor dataset, will then be introduced, and results from the experiments on the latter will be provided. A discussion as well as the conclusion will follow the experimentation.

3.2 LSTMCaps Autoencoder Network

3.2.1 Long Short-Term Memory Network

The LSTM network, proposed initially in 1997 by Hochreiter and Schmidhuber [1] but popularised recently by its widespread usage in commercial environments, is a popular iteration of the RNN that can overcome the vanishing gradient issue and allows for the learning of long-term dependency. The vanishing/exploding gradient problem commonly occurs in RNN architectures when training, using the backpropagation through time algorithm due to the depth of the unrolled RNN as well as the shared weights across the RNN cells. The calculated derivatives during training are prone to exponentially increasing or decreasing as the calculation progresses through the network. This results in either a lack of change or extreme changes to the RNN weight values, which in the very worst cases means that the model stops training completely in the case of a vanishing gradient, or trains erratically and fails to converge to a minimal error.

The LSTM architecture addresses this by using a specialised architecture that integrates "gates" to the architecture to allow the cell state to forget values and replace values, then decide which values to output and send to the next cell. The forget gate uses a sigmoid layer on the input x_t and previous hidden cell state h_{t-1} to output a vector f_t that determines which irrelevant data from the previous cell state, C_{t-1}, to remove from the current cell state.

$$f_t = \sigma \left(W_f \times [h_{t-1}, x_t] + \ b_f \right) \tag{3.1}$$

The input gate determines the information from the candidate values that is stored in the cell state by using a sigmoid layer on the input x_t and previous hidden cell state h_{t-1} to decide which values to update, i_t, and a tanh layer to create a vector of candidate values, $\widetilde{C}_t$, to add to the cell state; the Hadamard product of which is used as the values to be added to the new cell state.

$$i_t = \sigma\left(W_i \times [h_{t-1}, x_t] + \ b_i\right) \tag{3.2}$$

$$\widetilde{C}_t = \tanh\left(W_C \times [h_{t-1}, x_t] + b_C\right) \tag{3.3}$$

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t \tag{3.4}$$

The output gate determines which part of the previous hidden cell state $ht-1$ to output to the next cell by applying a sigmoid function on the input x_t and previous hidden cell state h_{t-1}, then calculating the Hadamard product between the tanh of the current cell state C_t and the output of the sigmoid gate o_t.

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + \ b_o) \tag{3.5}$$

$$h_t = o_t \times \tanh\left(C_t\right) \tag{3.6}$$

A visualisation of this architecture is illustrated in Figure 3.1.

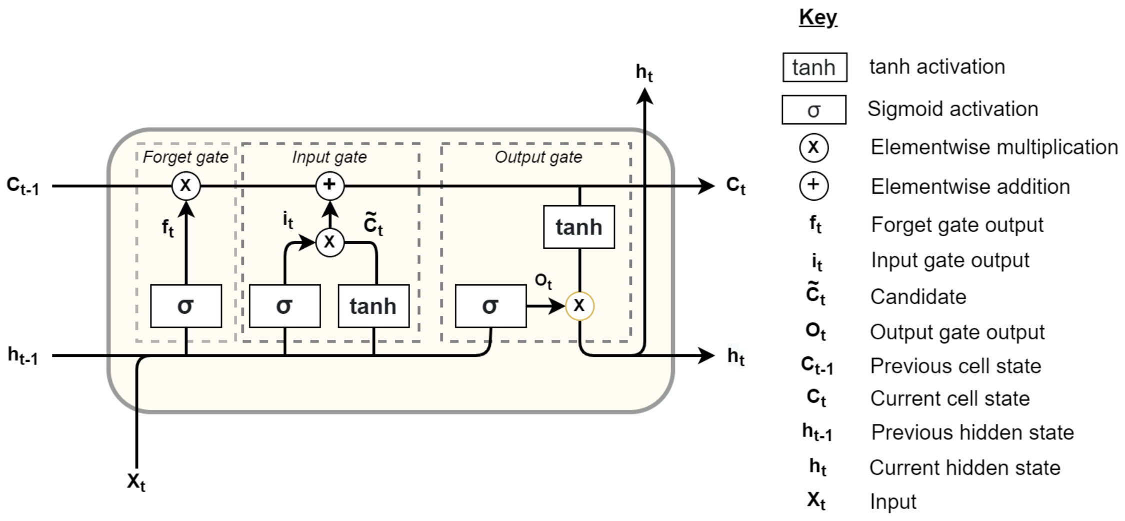


Figure 3.1: Visualisation of the LSTM cell [1].(Published in [2])

3.2.2 Capsule Network

The Capsule network (CapsNet) [20], is a novel neural network architecture designed to address the issues the CNN has with spatial context. A key limitation of traditional CNNs arises from pooling layers (e.g., max-pooling), which, while reducing dimensionality, often discard precise information about the spatial relationships and pose (position, orientation, scale) of detected features [141]. CapsNets aim to overcome this by ”encapsulating” the properties of features, including their spatial information, using vectors instead of scalars. This allows the neural network to learn not just the presence of features but also their relationships to one another. Figure 3.2 provides a conceptual illustration of this difference.

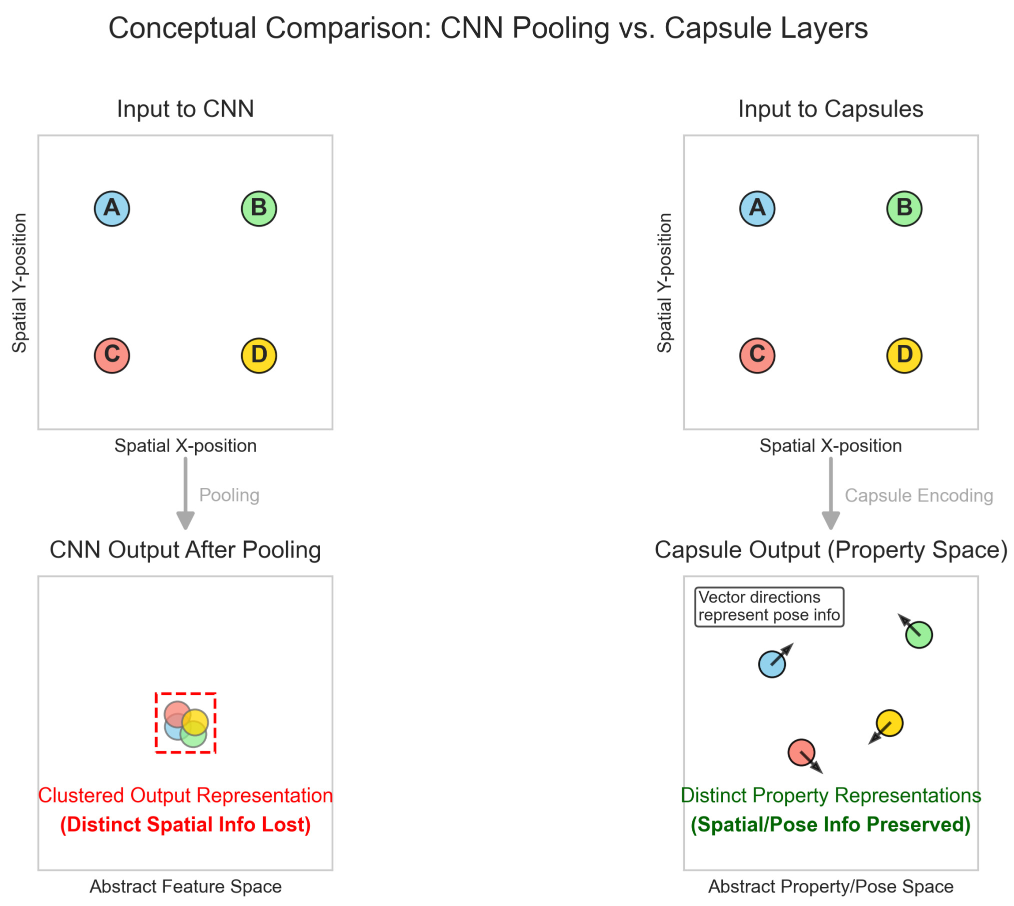

Conceptual Comparison: CNN Pooling vs. Capsule Layers
Input to CNN
Spatial Y-position
A
B
C
D
Spatial X-position
Pooling
CNN Output After Pooling
Clustered Output Representation
(Distinct Spatial Info Lost)
Abstract Feature Space
Input to Capsules
Spatial Y-position
A
B
C
D
Spatial X-position
Capsule Encoding
Capsule Output (Property Space)
Vector directions represent pose info
Distinct Property Representations
(Spatial/Pose Info Preserved)
Abstract Property/Pose Space

Figure 3.2: Conceptual comparison of information handling in CNNs vs. Capsule Networks. The same input containing spatially distributed features (A, B, C, D). After typical CNN pooling, the output representation often loses distinct spatial/pose information, shown here as clustered features in an abstract space. Capsule layers transform the input into a property/pose space where each feature is represented by a vector (arrow direction indicating pose/properties), preserving the distinct spatial relationships encoded in the vector orientations.

A capsule differs significantly from the traditional artificial neuron. A standard neuron receives scalar inputs, computes a weighted sum, applies a non-linear activation function, and outputs a single scalar value, indicating feature presence or intensity. In contrast, a capsule processes and outputs vectors. As shown conceptually in Figure 3.2, the vector output from a capsule (often called an 'activity vector') carries richer information: its length (magnitude) typically represents the probability that the entity detected by the capsule exists within the input, while its orientation encodes the entity's instantiation parameters or properties, such as its pose, deformation, or texture [20].

To achieve this, a capsule receives vector inputs from lower-level capsules (or transformed scalar features in the first capsule layer). It then applies an "affine transformation" using a weight matrix (W_{ij}) for each input vector (u_i), effectively predicting the pose of the higher-level feature based on the lower-level one. This replaces the traditional scalar weight multiplication and is formally defined in Equation 3.7 [20]:

$$\hat{u}_{j|i} = W_{ij} u_i \tag{3.7}$$

Here, $\hat{u}_{j|i}$ is the prediction vector from capsule i in the layer below for capsule j in the layer above, and W_{ij} is the learned transformation matrix. This transformation matrix allows the network to model spatial relationships (such as translation or rotation) between parts and wholes.

These prediction vectors are then combined through a weighted sum, typically determined by a 'routing-by-agreement' mechanism (though simpler variants exist [57]), to form the input s_j to the higher-level capsule j, as shown in Equation 3.8 [20]:

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{3.8}$$

where c_{ij} are coupling coefficients determined by the routing process.

Finally, to preserve the vector information (both magnitude and orientation) while normalising the output, a non-linear "squashing" activation function is applied. This function ensures that short vectors (low probability/agreement) are shrunk close to zero length, and long vectors (high probability/agreement) are shrunk to a length just below 1, effectively acting as a non-linearity while maintaining the directional information encoded in the vector. Equation 3.9 [20] formally defines this operation:

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \times \frac{s_j}{||s_j||} \tag{3.9}$$

where v_j is the final vector output of capsule j. This vector representation throughout the network enables CapsNets to be more robust to viewpoint changes and better model hierarchical relationships compared to traditional CNNs.

3.2.3 Autoencoder

An Autoencoder (AE) is a variant of neural network architecture that aims to learn a compressed representation of the input data and copy it to the output. A compressed representation is used so that the model does not learn the noise in a data representation but only the main shapes and features of the data. A visualisation of this can be seen in Figure 3.3.

An AE is composed of two sections: An encoder and a decoder. The encoder part is used to transform the input data into a latent space representation through dimensionality reduction, which the decoder part then learns and decodes back into the input data with reduced accuracy and hence noise. A formal definition of the AE operation is provided in Equations (3.10) and (3.11) (courtesy of Ref. [3]).

$$Z = e(X) \tag{3.10}$$

$$X' = d(Z) \tag{3.11}$$

There are various different configurations for an AE that are employed in different use cases. An undercomplete AE, which is the configuration used in the present study, encodes the input values to a compressed latent space representation. However, learning data that is too compressed would reduce the accuracy of the reconstruction, so when training the network, the aim is to balance the denoising ability with the accuracy of reconstruction. This is determined by the reconstruction loss, and the aim of training this type of network is to minimise this loss whilst maintaining a good generalisation performance. This type of neural network is typically used for unsupervised deep learning, as the inputs are being copied to the outputs with no labelling required, which is suitable for the use case that this study explores.

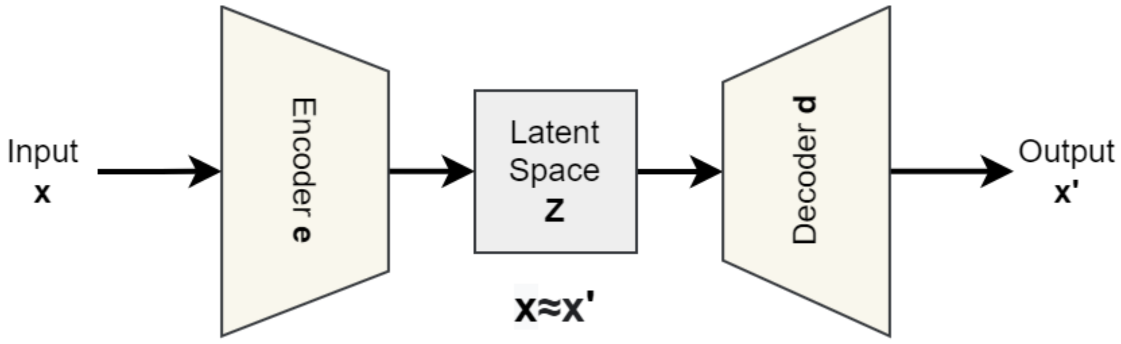


Figure 3.3: Autoencoder architecture visualised (courtesy of Ref. [3], published in [2]).

3.2.4 Proposed Model Architecture

The dataset is first checked for the number of features to be used in the model; this will determine the number of input branches as each signal is input in a single branch. The data is fed into the network using a sliding window, where the size of the window is referred to as the "time steps" or lookback of the network; how many datapoints in time the NN uses to make a prediction. This value is constant and determines the shape of the network, and is thus a hyperparameter that is optimised during model training.

Each individual feature is first encoded through an LSTM layer using dimensionality reduction and is thus output as a 1D vector. This dimensionality reduction is carried out to reduce the number of degrees of freedom in the model so that the risk of overfitting on data is reduced, and the most prominent features in each signal are highlighted. The strength of the LSTM in identifying long-term dependency of TS features is utilised here in order to capture the univariate temporal features in each input feature. Each feature is then separately decoded using a Capsule layer for univariate spatial feature learning, where the number of Capsules in the layer is dependent on the "time steps" of the input data. The 2D vector output of each Capsule layer is then concatenated as a 3D vector into a single channel, which is passed through a single Capsule layer so that multivariate spatial feature learning between the previously separated input features is undertaken. A fully-connected layer is then applied to each temporal slice in the data, where the number of temporal slices is the window size of the input or the time steps. The resulting output is the reconstruction of the separate 2D input vectors in a single 3D vector where the third dimension is the number of input features. The proposed model architecture is illustrated in Figure 3.4.

The number of input branches and therefore the trainable parameters scale with the number of features present in the dataset. Whilst this can be seen as a disadvantage of such an approach, the increased volume of data will generally result in the need for a larger model with more training parameters, otherwise the model may run the risk of underfitting on the data if there are too few trainable parameters to accurately model the data sufficiently. Furthermore, we show empirically during experimentation that the number of trainable parameters in the proposed model can be reduced to a value comparable to that of a single channel NN and still outperform the latter.

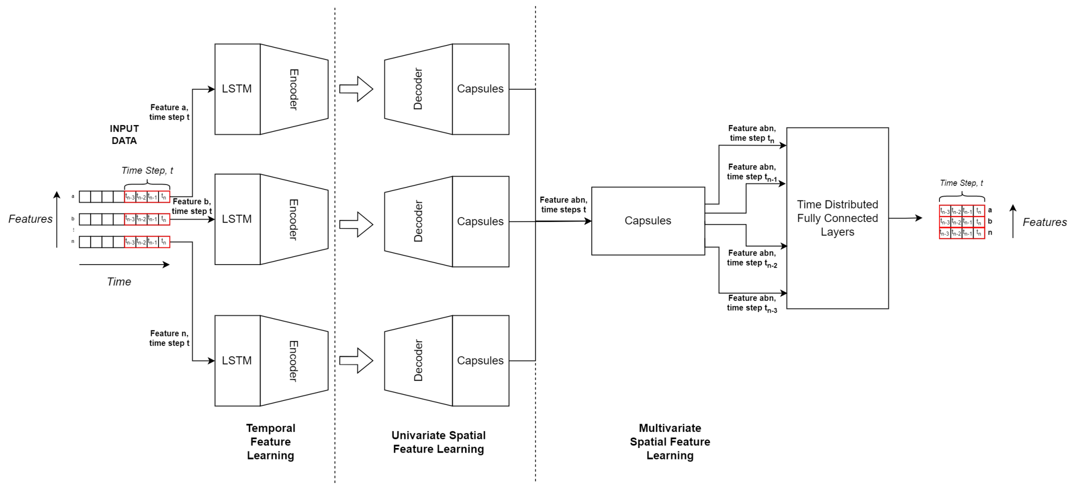

INPUT DATA
Time Step, t
Features
a
b
n
t_{n-3} t_{n-2} t_{n-1} t_n
Time
Feature a, time step t
Feature b, time step t
Feature n, time step t
LSTM
Encoder
Decoder
Capsules
Feature abn, time steps t
Feature abn, time step t_n
Feature abn, time step t_{n-1}
Feature abn, time step t_{n-2}
Feature abn, time step t_{n-3}
Time Distributed Fully Connected Layers
Temporal Feature Learning
Univariate Spatial Feature Learning
Multivariate Spatial Feature Learning

Figure 3.4: Proposed model architecture.

3.2.5 Model Training and Anomaly Detection Method

This section will explore the method of training the proposed model on a dataset for the purpose of anomaly detection.

Data Preprocessing

Data pre-processing is an integral part of any model application due to the effect it can have on model performance. One important pre-processing step is data scaling, which has shown to have a significant impact on performance across literature. However, the type of scaling used has been shown to be dependent on the dataset as well as the model type used [142] and therefore should only be decided during model testing, as there is no specific approach that yields better results than others. Therefore, the present study will address the scaling later during experimentation.

The proposed model requires a specific data shape to be input to the NN due to the layer types used. As the data is fed into the NN in "time steps", which refers to a window of time that advances by a datapoint for each sample, the data shape must reflect this. This can be represented mathematically with the following: A dataset with shape $(samples,\ features)$ will be reshaped to $(samples - time\ steps,\ time\ steps,\ features)$ where, for a sample t_n, each time step will contain the list of values $(t_{n-time\ steps}, \ldots, t_{n-1}, t_n)$. The number of samples will be reduced by the size of the time steps as the first sample will include the first n_t values that constitute a single time step.

Training

In order to be able to detect anomalies in a dataset, the proposed model must be trained to reconstruct data that is known to be "healthy". This requires some domain knowledge in order to verify what is defined as "healthy" but can be done relatively simply nonetheless by using a device or system that is known to be operating in a "healthy" condition as a reference point. In the case of anomalies appearing in the healthy data, the model will need to exhibit a resilience to learning these anomalies in order for the anomaly detection performance to not be affected. In order to investigate this, an experiment will be conducted to observe the effect of anomalies in the training data.

As the model takes in each feature separately, the 3D array ($samples{-}timesteps$, $time\ steps$, $features$) will then be split into multiple 2D arrays where the number of arrays is equal to the number of features in the data, and each array will be input into the relevant input channel. The model output will be a reconstruction of each 2D array input $(samples, time\,steps, feature_A), (samples, time\,steps, feature_B) \ldots$ $(samples, timesteps, feature_N)$ as a single 3D array $(samples{-}timesteps, timesteps$ $, features)$. This is visually represented in Figure 3.4.

Anomaly Detection

The reconstruction error can be found using the MAE of the training predictions. The maximum prediction error for the training set can be used as the reconstruction error threshold, which essentially means that the worst prediction case is being used as the threshold initially so that when applying the system to more data from the system being analysed, any predictions outside this value will be more likely to be an anomaly. The sensitivity of the anomaly detection can be adjusted by changing the threshold value; however, this value will not be adjusted in the present study. Furthermore, each data feature will have its own error threshold to maximise the accuracy of detection as the model may reconstruct less complex features more accurately than others. The main aim of the training process is to minimise the standard deviation of this plot so that the system is able to make more confident anomaly predictions.

3.2.6 Experimental Results

Experimental Design

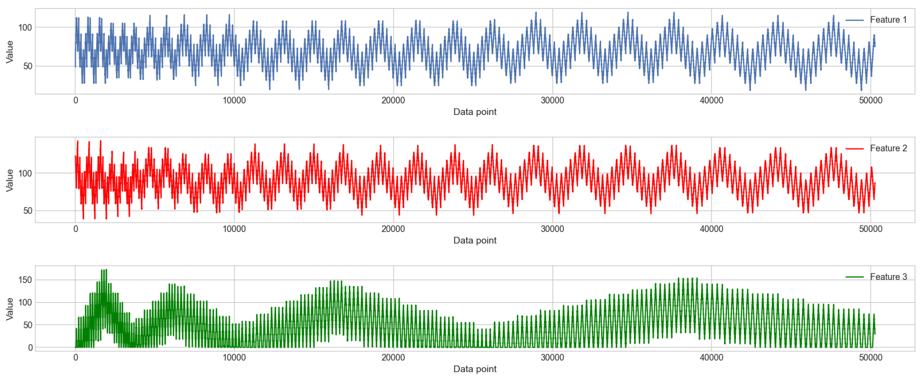
Each of the following experiments will aim to provide insight into different aspects of the proposed approach: Training performance, anomaly detection performance, resilience to noise in the dataset, and comparison with popular and state-of-the-art anomaly detection methods. Due to this, each experiment design will be different and will therefore be covered in the relevant section. However, each experiment will always be repeated five times for experimental rigour. This will also allow for an analysis of the stability of the approach, as well as the statistical significance of the results for each experiment. This is essential for the replicability and reliability of the results.

The statistical analysis of the experimental results was performed using the Kruskal-Wallis H test as described in Section 1.5.1 of this thesis, which is more appropriate than parametric tests like ANOVA for the non-normally distributed performance metrics observed in this study.

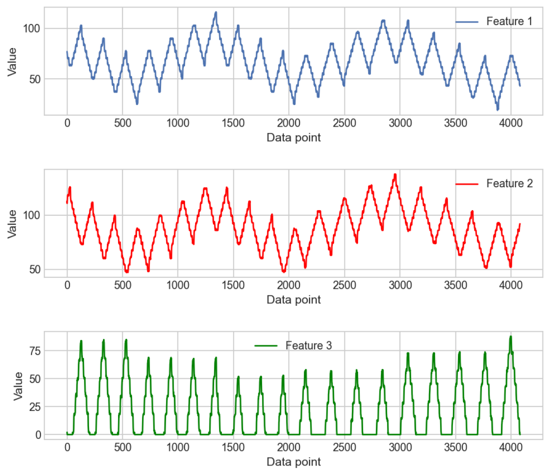
Drone Dataset Anomaly Detection

The drone dataset was acquired from previous studies conducted on Virtual Sensing fault detection [24]. The datasets consists of 3 subsets of data sampled at 150 Hz: 2 subsets of data are taken from a fully functional drone, where 1 subset is sampled for a duration of 600 s, giving 90,000 samples, and the other for 30 s, giving 4500 samples. These subsets will be used for the training and validation sets, respectively. The third subset is acquired from a drone with faulty controls and is recorded for a duration of 30 s, similarly giving 4500 samples. This subset will be used as the test data. From this dataset, 3 features are being used. Feature 1 represents the pitch, feature 2 represents the roll, and feature 3 represents the throttle of the drone. These features overall represent the input signals to the drone, so anomaly detection on these signals will serve the purpose of detecting any abnormal control issues on the drone. More details on the acquisition of this data can be found in Ref. [24]. A visualisation of the training data, the validation data, and the test data with various anomalies outlined is provided in Figure 3.5. It is important to note that the cyclical pattern observed in the initial segment of the training data (visualised in Figure 3.5a), while visually distinct, represents characteristic behaviour within the normal operational envelope of the drone control signals in this dataset, rather than a transient start-up or warm-up phase.

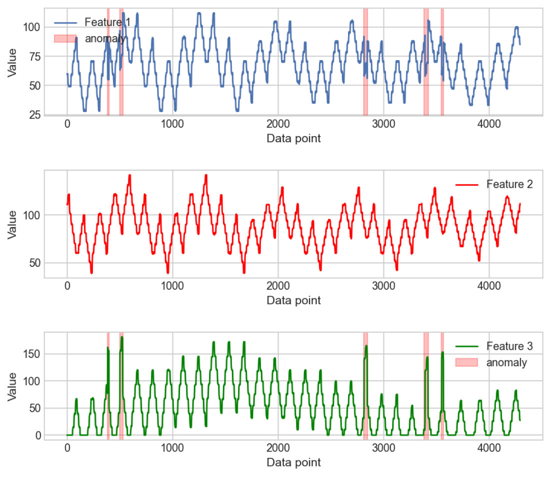
As the data is unlabelled, the anomalies were manually labelled so that a measure of the anomaly detection performance of each NN model could be attained. The metric used for this is the F_1 score, which is defined in Equation (3.12) [143]



(a) Training data



(b) Validation data



(c) Testing data (anomalies in red)

Figure 3.5: Visualisation of the drone dataset.

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.12}$$

The F_1 score can also be expressed in terms of Precision, Equation (3.13), and Recall, Equation (3.14). The F_1 score expressed in these terms is described in Equation (3.15).

$$Precision = \frac{TP}{TP + FP} \tag{3.13}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.14}$$

$$F_1 = 2 \cdot \frac{precision \times recall}{precision + recall} \tag{3.15}$$

where TN = True positive, TN = True negative, FP = False positive and FN = False negative.

For this dataset, it was found to be advantageous to normalise the data before using the Z-Score normalisation to improve the performance of the models. This operation transforms the mean of the sample to 0 and the standard deviation to 1. This is accomplished using Equation (3.16).

$$z = \frac{(X - \mu)}{\sigma} \tag{3.16}$$

This experiment aims to explore the training capability and anomaly detection performance of the proposed NN architecture (Design D) by comparing it against several variants. These comparisons help isolate the specific contributions of using a multi-channel input structure versus a single-channel one, and the impact of using Capsule networks within the decoder stage. The architectures are visualised across two figures: Figure 3.6 shows the single-channel approaches, and Figure 3.7 details the multi-channel approaches. The specific designs represent systematic variations:

- **Single-Channel Baselines (Figure 3.6):** These models, shown in Figure 3.6, process the multivariate input as a single sequence, lacking separate initial processing paths for each feature.
  - *Design A (Figure 3.6a):* A standard single-channel Autoencoder using LSTM layers for both encoding and basic decoding (reconstruction typically via Dense layers). This serves as the primary baseline representing conventional sequential autoencoder approaches.
  - *Design B (Figure 3.6b):* A single-channel hybrid LSTMCaps Autoencoder. It maintains the single-channel input structure and LSTM encoder but replaces the standard decoder with Capsule layers. Comparing Design A and B directly assesses the benefit of Capsule decoders over standard LSTM/Dense decoders when processing the multivariate data as one sequential entity.
- **Multi-Channel Architectures (Figure 3.7):** These models, detailed in Figure 3.7, use a branched input structure, processing each time series feature (channel) independently in the initial encoding layers before potentially merging information later.

- *Design C (Figure 3.7a):* A multi-channel Autoencoder employing parallel LSTM encoder/decoder branches for each input feature. The outputs are typically concatenated or merged for final reconstruction. This architecture explores the effect of parallel temporal processing specific to each feature, without leveraging Capsules. Comparing Design A and C highlights the impact of adopting a multi-channel input strategy.
- *Design D (Figure 3.7b):* The proposed multi-channel LSTMCaps Autoencoder. This architecture combines the branched structure, using separate LSTM encoders per channel, with individual Capsule decoders. Crucially, it includes a final Capsule layer designed to explicitly model and learn the spatial or correlational relationships between the encoded features derived from the different input channels before final reconstruction. Comparing Design C and D isolates the contribution of Capsules specifically within the multi-channel framework, while comparing Design D against all others demonstrates its overall proposed effectiveness.

Since the trainable parameters of the proposed model, Design D and the multi-channel LSTM, Design C, will scale with the number of features in the data, the size of each model was adjusted so that all networks being trained have a similar number of parameters. This is done to show the difference in performance of each NN model variant regardless of the number of parameters, which will result in a fairer comparison of the variants. The trainable parameters for each model variant are shown in Table 3.1. The models were optimised for the drone dataset using the F_1 score as the target variable to be optimised. The optimal hyperparameters found for the models are found in Table 3.2.

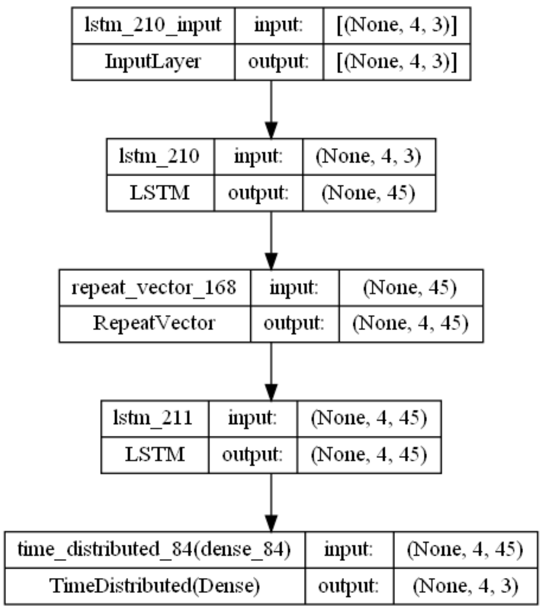
Each model was trained 5 times on the 5-minute subset from the reference device to observe consistency and determine statistical significance of the training and validation loss values. Table 3.3 depicts the average and best loss scores as well as the percentage difference in the training and validation scores, referred to as "% Overfitting" and the improvement in training performance with the inclusion of the Capsule Layer. For each individual training procedure, the overall prediction Mean Squared Error (MSE) for the validation set was also calculated. A training plot from one training run from each model is also illustrated in Figure 3.8.

Table 3.1: Trainable parameters for each NN model variant, which were kept as similar as possible for experimental rigour.

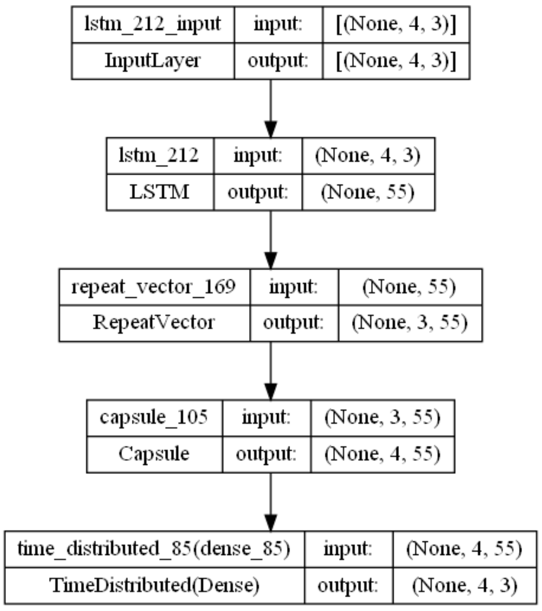
Model	**Trainable Parameters**
Design A: single channel LSTM	25,338
Design B: single channel LSTMCaps	25,248
Design C: multi-channel LSTM	25,473
Design D: multi-channel LSTMCaps (proposed)	24,663

Table 3.2: Hyperparameter values used across the models for the drone dataset experimentation.

Hyperparameter	Value	Hyperparameter	Value
Epochs	20	Loss Function	Huber
Batch size	1024	Optimiser	Adam [144]
Learning rate	0.0003	LSTM Activation	tanh
Time Steps	4	Capsule Activation	relu

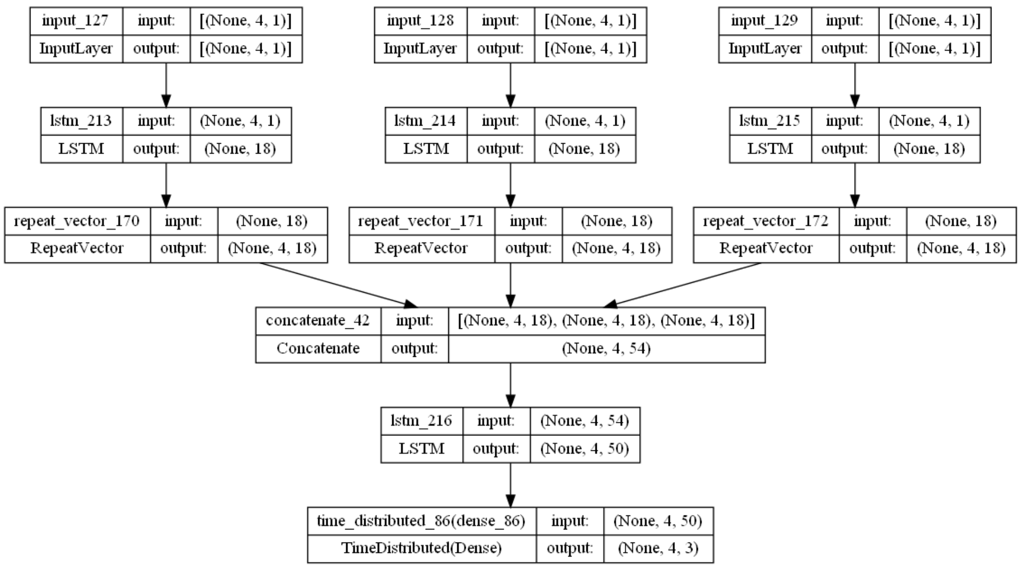


(a) Design A: Single-channel LSTM AE

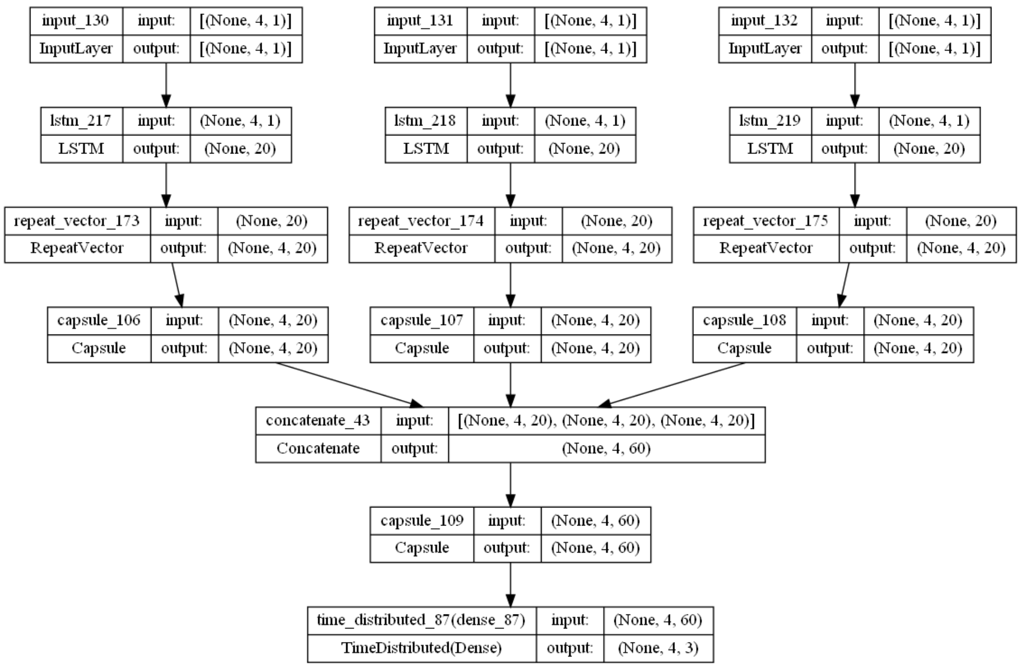


(b) Design B: Single-channel LSTMCaps AE

Figure 3.6: Visualisation of the single-channel Autoencoder (AE) architectures used for comparison. These models process all input features concatenated into a single sequence. (a) Design A employs standard LSTM layers for encoding and decoding. (b) Design B uses an LSTM encoder but incorporates Capsule layers for decoding.


input_127 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
input_128 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
input_129 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
lstm_213 LSTM input: (None, 4, 1) output: (None, 18)
lstm_214 LSTM input: (None, 4, 1) output: (None, 18)
lstm_215 LSTM input: (None, 4, 1) output: (None, 18)
repeat_vector_170 RepeatVector input: (None, 18) output: (None, 4, 18)
repeat_vector_171 RepeatVector input: (None, 18) output: (None, 4, 18)
repeat_vector_172 RepeatVector input: (None, 18) output: (None, 4, 18)
concatenate_42 Concatenate input: [(None, 4, 18), (None, 4, 18), (None, 4, 18)] output: (None, 4, 54)
lstm_216 LSTM input: (None, 4, 54) output: (None, 4, 50)
time_distributed_86(dense_86) TimeDistributed(Dense) input: (None, 4, 50) output: (None, 4, 3)

(a) Design C: Multi-channel LSTM AE

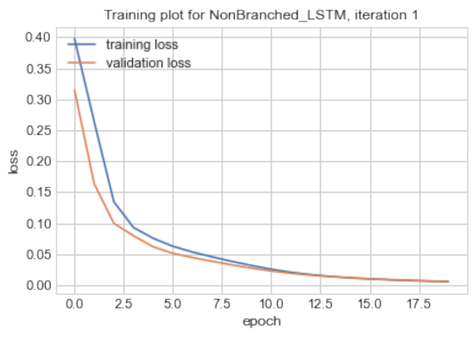

input_130 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
input_131 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
input_132 InputLayer input: [(None, 4, 1)] output: [(None, 4, 1)]
lstm_217 LSTM input: (None, 4, 1) output: (None, 20)
lstm_218 LSTM input: (None, 4, 1) output: (None, 20)
lstm_219 LSTM input: (None, 4, 1) output: (None, 20)
repeat_vector_173 RepeatVector input: (None, 20) output: (None, 4, 20)
repeat_vector_174 RepeatVector input: (None, 20) output: (None, 4, 20)
repeat_vector_175 RepeatVector input: (None, 20) output: (None, 4, 20)
capsule_106 Capsule input: (None, 4, 20) output: (None, 4, 20)
capsule_107 Capsule input: (None, 4, 20) output: (None, 4, 20)
capsule_108 Capsule input: (None, 4, 20) output: (None, 4, 20)
concatenate_43 Concatenate input: [(None, 4, 20), (None, 4, 20), (None, 4, 20)] output: (None, 4, 60)
capsule_109 Capsule input: (None, 4, 60) output: (None, 4, 60)
time_distributed_87(dense_87) TimeDistributed(Dense) input: (None, 4, 60) output: (None, 4, 3)

(b) Design D: Multi-channel LSTMCaps AE (Proposed)

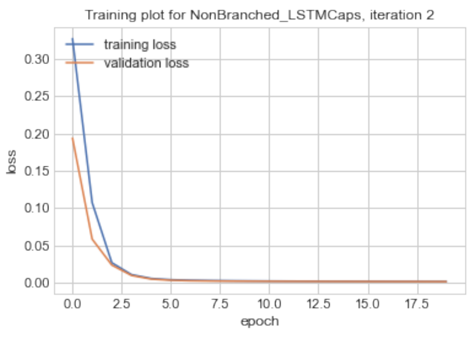
Figure 3.7: Visualisation of the multi-channel (branched) Autoencoder (AE) architectures. These models process each input feature stream independently in initial layers. (a) Design C uses parallel LSTM encoder/decoder branches. (b) Design D, the proposed model, uses parallel LSTM encoders combined with Capsule decoders and a final multivariate Capsule layer to learn inter-feature relationships.

Table 3.3: Results for training for each NN model in Figure 3.6 and Figure 3.7 for the drone experiment using the hyperparameters from Table 3.2.

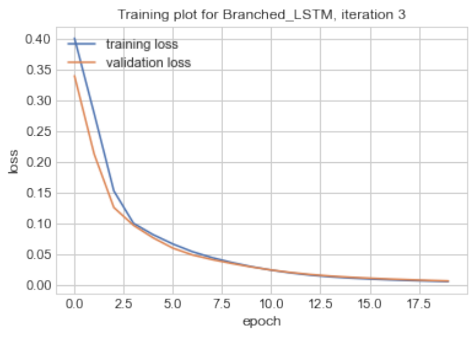
Model	Score	Final Training Loss	Final Validation Loss	Training Time (s)	MSE	% Overfitting	% Val Loss Improvement from Non-Caps
Design A	Average over 5 runs	0.00591	0.00639	37.75	10.37	7.45	N/A - Non-Caps
	Best over 5 runs	0.00589	0.00614	43.42	10.10	4.09	
	Standard Deviation	0.00034	0.00046	3.22	0.85	0.04953	
Design B	**Average over 5 runs**	**0.00158**	**0.00157**	**50.44**	**2.12**	**−0.27**	**3.07**
	Best over 5 runs	**0.00164**	**0.00159**	**50.71**	**2.15**	**−3.15**	**2.87**
	Standard Deviation	**0.00006**	**0.00008**	**0.58**	**0.10**	**0.00005**	
Design C	Average over 5 runs	0.00575	0.00620	56.52	10.08	7.28	N/A - Non-Caps
	Best over 5 runs	0.00610	0.00709	56.09	12.41	14.01	
	Standard Deviation	0.00034	0.00074	0.83	1.89	0.00034	
Design D (proposed)	Average over 5 runs	0.00162	0.00161	143.34	2.14	-0.69	2.84
	Best over 5 runs	0.00168	0.00172	142.59	2.32	2.52	3.12
	Standard Deviation	0.00005	0.00006	1.63	0.11	0.00005	



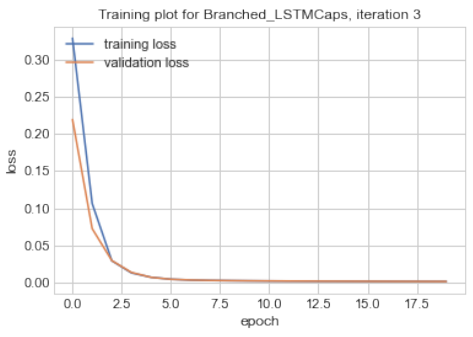
(**a**)



(**b**)



(**c**)



(**d**)

Figure 3.8: Training plot comparison for best models in Table 3.3. (**a**) Design A; (**b**) Design B; (**c**) Design C; (**d**) Design D (proposed).

The MAE thresholds were calculated using the maximum MAE values on the validation set prediction. The NN models then ran inference on the test data, and any predictions exceeding the thresholds set were outlined as anomalies. The predicted anomalies were then compared to the real anomalies labelled during data analysis, and the precision, recall, and F_1 scores for single data-point outliers were calculated for each NN. The best and average score attained by each NN model variant over five runs is shown in Table 3.4, and a visualisation of the anomalies detected in the best iteration for each model is shown in Figure 3.10. The point-wise detections in this figure should be visually compared against the true anomaly locations (indicated by shaded regions in Figure 3.10c) to assess performance. True

Positives are red dots within the shaded regions, False Positives are red dots outside, and shaded regions lacking red dots represent False Negatives, forming the basis for the point-wise F1 score calculations in Table 3.4. The KW results table for the F_1 scores is depicted in Table 3.5.

When interpreting anomaly detection performance metrics like the F1 score, it is crucial to consider the context of the task. Unsupervised anomaly detection in complex time series data, particularly with subtle anomalies or noisy signals as potentially present here, is inherently challenging. Unlike supervised classification where scores above 0.9 are common, F1 scores in unsupervised anomaly detection can vary widely. Achieving perfect scores is rare. Performance should be evaluated relative to the application's tolerance for false positives versus false negatives and, importantly, relative to other methods tested on the same data. In this study, while absolute F1 scores (e.g., in Table 3.4) might appear modest (around 0.5-0.6), the primary goal is to demonstrate the comparative advantage of the proposed architecture over baseline and variant models under identical conditions. Scores in this range can still indicate effective detection, especially if they represent a significant improvement over alternatives or meet specific operational requirements. Furthermore, it is critical to note the evaluation methodology: the F1 score here represents single, point-wise detections. This is a significantly stricter evaluation than event-based metrics (like NAB or range-based F1 scores), which often only require detecting some part of an anomalous event or range. Accurately classifying the majority of individual points within an anomaly is more demanding and naturally leads to lower point-wise F1 scores compared to event-based scores for the same underlying detection capability.

Table 3.4: Best and average test results out of five runs for anomaly detection using optimised hyperparameters in Table 3.2 for each NN Design, where MAE is Mean Absolute Error, F_1 is F_1 Score (Equation (3.15)).

Model	Score	MAE Threshold 1	MAE Threshold 2	MAE Threshold 3	Precision	Recall	F_1
Design A	Average over 5 runs	6.13	6.06	10.81	0.37	0.59	0.45
	Best over 5 runs	5.60	6.70	11.90	0.40	0.63	0.49
	Standard Deviation	0.63	0.49	0.75	0.03	0.04	0.03
Design B	Average over 5 runs	4.27	4.12	9.98	0.51	0.41	0.45
	Best over 5 runs	4.31	4.08	10.62	0.57	0.51	0.54
	Standard Deviation	0.15	0.06	0.59	0.06	0.11	0.08
Design C	Average over 5 runs	7.96	6.87	9.88	0.54	0.50	0.52
	Best over 5 runs	7.70	6.92	11.22	0.58	0.50	0.53
	Standard Deviation	0.71	0.64	1.01	0.05	0.01	0.02
Design D (proposed)	**Average over 5 runs**	**4.37**	**4.20**	**9.79**	**0.53**	**0.54**	**0.54**
	Best over 5 runs	**4.50**	**4.21**	**9.81**	**0.61**	**0.59**	**0.60**
	Standard Deviation	**0.15**	**0.07**	**0.61**	**0.06**	**0.06**	**0.05**

The results in Table 3.3 show that the proposed additions result in an overall better training performance. The addition of the Capsule layer to both the multi-channel and single channel variant of the model architecture shows a clear improvement of the training and validation losses over the same number of training epoch in comparison to the variants without Capsules. Furthermore, whilst all architectures did not significantly overfit on the training data, the architecture variants with Capsules exhibited no difference between the training and validation losses and in some cases, the validation loss was marginally below the training loss, which indicates strong generalisation ability when using Capsules. The training plots in Figure 3.8 also show that the architecture variants with Capsules converge to low loss values

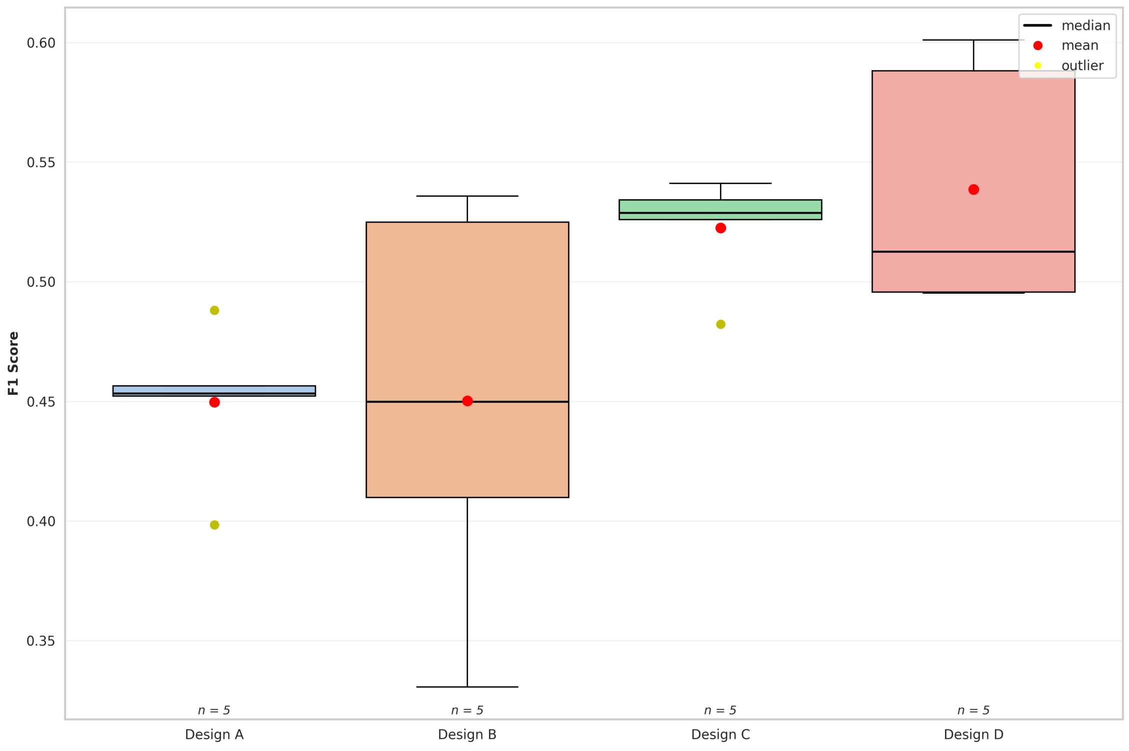
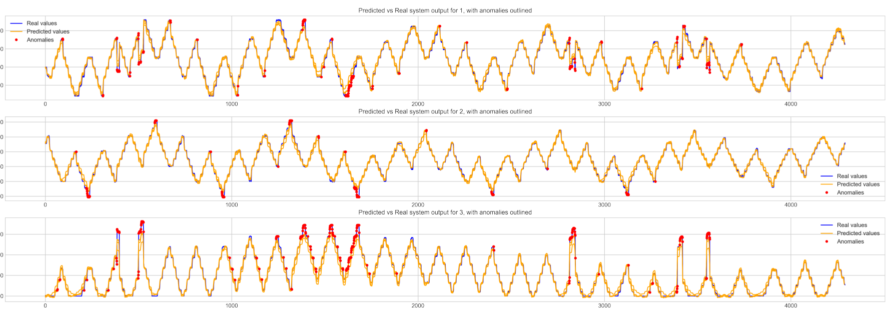


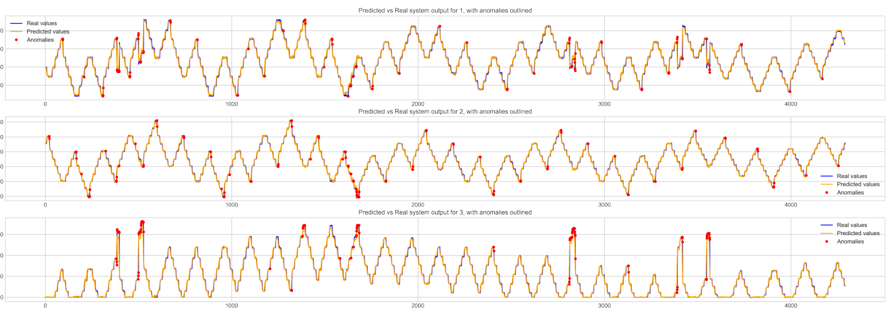
Figure 3.9: Box-plot of the F_1 scores for the drone experiment

significantly faster than the variants without Capsules, which means that without a fixed number of training epochs, the Capsule based models will require less training epochs to converge to the same loss values as the model variants without capsules. One drawback that can be observed however is the longer training times for the variants using Capsules, which is expected as Capsule-based Networks require more complex calculations than traditional NN architectures due to the dynamic routing algorithm. However, due to the faster convergence, reducing the number of epochs on the model variants with Capsules is feasible and will reduce the training time to a period comparable with the variants without capsules.

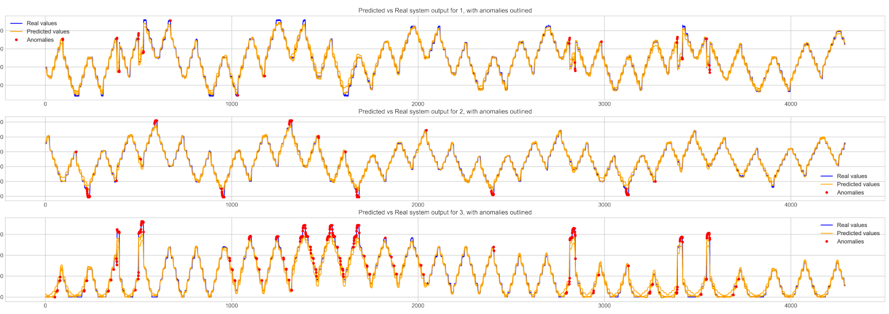
The anomaly detection results in Table 3.4 show that both the inclusion of Capsules and a multi-channel input architecture result in stronger anomaly detection performance. Both variants with multi-channel inputs, Designs B and D, are able to perform more confident predictions, as shown by the MAE thresholds for each feature in the data. Due to this, the model variants with Capsules also achieve better F_1 scores than the single channel variants. The proposed model, Design D, outperforms the other models tested with anomaly detection with an average F_1 score of 0.53, and a best F_1 score of 0.60, which supports the conclusion that both the inclusion Capsules and the multi-channel input architecture play an important role in improving the performance of the model.



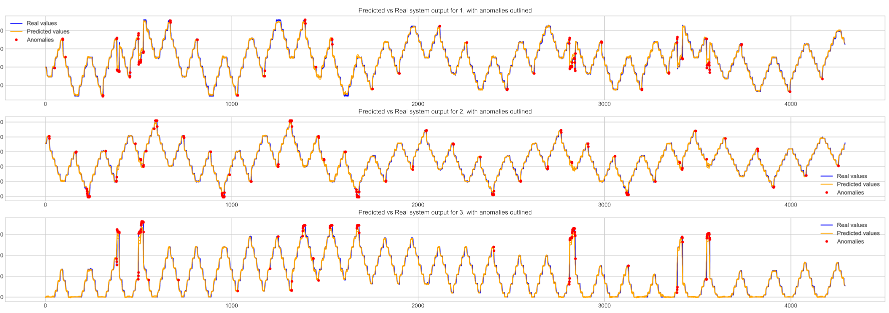
(a) Design A: single channel LSTM



(b) Design B: single channel LSTMCaps



(c) Design C: multi-channel LSTM



(d) Design D: multi-channel LSTMCaps (Proposed model)

Figure 3.10: Visualisation of anomaly detection results on the test dataset for the three features, generated by the best iteration of each design. **Red dots highlight individual data points where the reconstruction error (MAE) exceeded the calculated anomaly threshold.**

Table 3.5: Kruskal-Wallis test results for the F_1 score of the drone experiment.

Model	n	Mean	SD	Median	Chi-square	*p*-Value
Design D	5	0.54	0.05	0.51	8.49	0.037*
Design C	5	0.52	0.02	0.53		
Design B	5	0.45	0.08	0.45		
Design A	5	0.45	0.03	0.45		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis reveals significant difference between Design A and Design D.

The plots in Figure 3.10 show that each model was able to detect all the anomalies in the data. However, Figure 3.10b,d demonstrate that using by Capsules, less False Positives are flagged in the data in comparison to Figure 3.10a,c, the plots for the non-capsule model variants. Due to the method of calculation, the F_1 scores achieved by each model may not be fully representative of the abilities of each model. This aspect can be further explored and improved in future works.

The results attained give evidence to show that both the hybridisation of the Capsule and LSTM layers and the multi-channel input model structure are both effective methods for improving the performance of the neural network with multivariate data, especially when used in conjunction with each other. Furthermore, the results also show that the proposed model is superior during training in terms of convergence and resilience to overfitting. Additionally, the p-value of 0.037 calculated using the KW test in Table 3.5 suggests that the test groups are statistically different, and unlikely to be a result of randomness as is the case with some NN models due to their stochastic nature. The next section will explore the effect of "unclean" training data on the performance of the model variants.

Drone Dataset Outlier Resilient Anomaly Detection

This experiment aims to explore the resilience of each model variant presented in Figure 3.6 and 3.7 when trained on data containing artificial outliers. Such outliers might simulate transient sensor errors or noise sometimes present in real-world data acquisition. To prepare the training data for this test, a total of 200 anomalous data points were randomly generated and inserted into the original 'healthy' training set across its three features (pitch, roll, throttle). The magnitude of these artificial anomalies was designed to significantly deviate from the normal signal range. Subsequently, the same experimental procedure for training and evaluation, as outlined in Section 3.2.6, was conducted using this modified training data.

Figure 3.11 provides a visualisation of a segment of this modified training set, clearly illustrating the presence and nature of the inserted artificial outliers. These points were deliberately excluded from the validation and test sets, ensuring the evaluation measures the model's ability to detect **true** system anomalies after being trained on **imperfect** data. Table 3.6 shows the best and average training results for each model variant under these conditions, and Table 3.7 presents the corresponding anomaly detection scores on the original (unmodified) test set, with a supporting box-plot in Figure 3.12. The statistical significance of the differences in anomaly detection performance (F_1 scores) is assessed using the KW test, with results depicted in Table 3.8. Examining Figure 3.11, one can observe examples of these randomly inserted outliers - note the sharp, isolated spikes deviating significantly from the regular oscillating patterns, particularly visible in the throttle signal (green trace, bottom plot) around data points 5000, 25,000, and 42,000, among others scattered throughout the dataset.

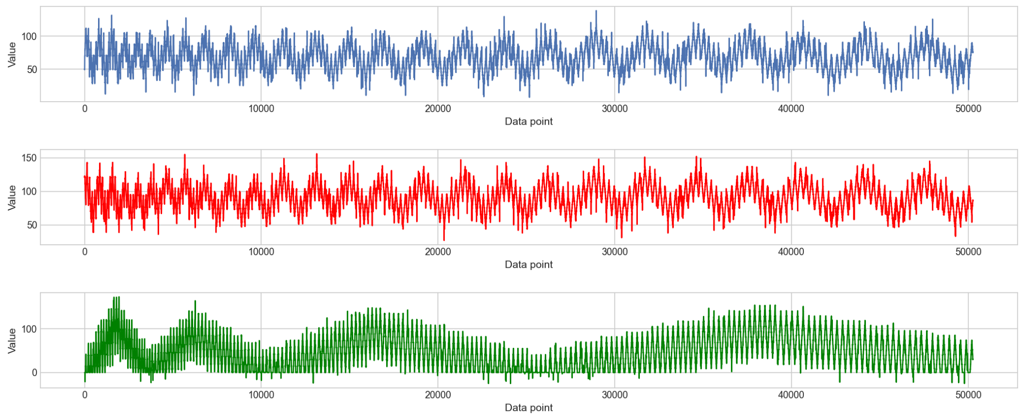


Figure 3.11: Visualisation of the drone training dataset (features 1-3 shown top to bottom) containing 200 randomly inserted artificial outlier points (examples visible as sharp spikes, e.g., in the green trace around points 5000, 25,000, 42,000) used for the outlier resilience experiment.

Table 3.6: Results for training for each NN model in Figure 3.6 and Figure 3.7 for the drone experiment with outliers using hyperparameters from Table 3.2.

Model	Score	Final Training Loss	Final Validation Loss	Training Time (s)	MSE	% Overfitting	% Val Loss Improvement from Non-Caps
Design A	Average over 5 runs	0.00626	0.00667	35.89	10.61	6.12	N/A–Non-Caps Version
	Best over 5 runs	0.00614	0.00653	36.53	10.19	5.98	
	Standard Deviation	0.00024	0.00035	0.41	0.71	0.00035	
Design B	**Average over 5 runs**	**0.00188**	**0.00155**	**49.77**	**2.08**	**−21.24**	**3.31**
	Best over 5 runs	**0.00194**	**0.00158**	**49.62**	**2.09**	**−22.59**	**3.14**
	Standard Deviation	**0.00004**	**0.00003**	**0.23**	**0.04**	**0.00003**	
Design C	Average over 5 runs	0.00575	0.00620	56.52	10.08	7.28	N/A–Non-Caps Version
	Best over 5 runs	0.00610	0.00709	56.09	12.41	14.01	
	Standard Deviation	0.00052	0.00071	0.83	1.13	0.00071	
Design D (proposed)	Average over 5 runs	0.00162	0.00161	143.34	2.14	−0.69	2.84
	Best over 5 runs	0.00168	0.00172	142.59	2.32	2.52	3.12
	Standard Deviation	0.00007	0.00007	0.76	0.084	0.00007	

Table 3.7: Best and average test results out of five runs for anomaly detection using optimised hyperparameters in Table 3.2 for each NN Design.

Model	Score	MAE Threshold 1	MAE Threshold 2	MAE Threshold 3	Precision	Recall	F_1
Design A	Average over 5 runs	6.34	6.65	10.91	0.37	0.59	0.45
	Best over 5 runs	6.20	7.56	10.87	0.37	0.67	0.48
	Standard Deviation	0.20	0.63	0.56	0.02	0.05	0.03
Design B	Average over 5 runs	4.09	4.11	9.61	0.46	0.47	0.46
	Best over 5 runs	3.92	3.97	9.77	0.49	0.52	0.50
	Standard Deviation	0.17	0.10	0.61	0.05	0.04	0.04
Design C	Average over 5 runs	7.73	6.10	9.25	0.54	0.50	0.52
	Best over 5 runs	7.10	5.57	9.41	0.49	0.51	0.50
	Standard Deviation	0.47	0.36	0.39	0.03	0.01	0.01
Design D (proposed)	**Average over 5 runs**	**4.44**	**4.15**	**9.58**	**0.52**	**0.53**	**0.53**
	Best over 5 runs	**4.19**	**4.08**	**10.36**	**0.60**	**0.65**	**0.62**
	Standard Deviation	**0.24**	**0.17**	**0.52**	**0.05**	**0.07**	**0.06**

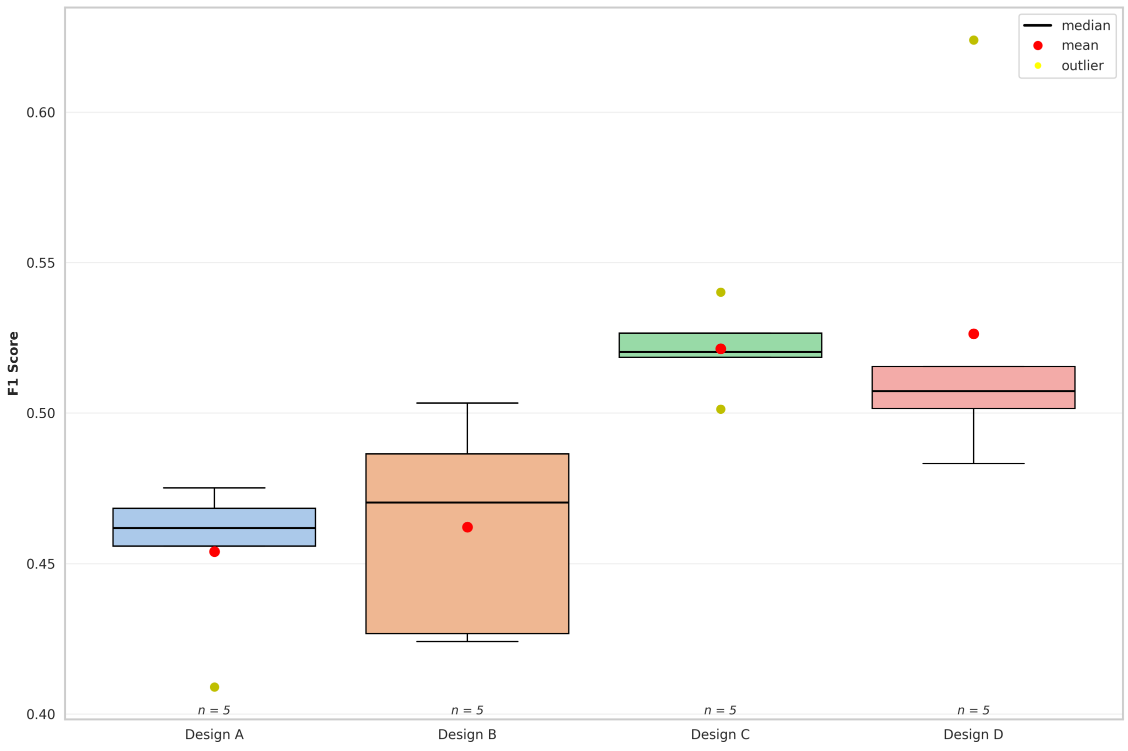


Figure 3.12: Box-plot of the F_1 scores for the drone experiment with outliers

Table 3.8: Kruskal-Wallis test results for F_1 scores of the drone experiment with outliers in the training data.

Model	**n**	**Mean**	**SD**	**Median**	**Chi-square**	***p*-Value**
Design D	5	0.53	0.06	0.51		
Design C	5	0.52	0.01	0.52	12.78	0.005*
Design B	5	0.46	0.04	0.47		
Design A	5	0.45	0.03	0.46		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis reveals significant differences between Design A-Design C and Design A-Design D.

The results in Table 3.7 show that all the tested models have resilience to “non-ideal” training data that contains anomalies. In fact, results show a marginal improvement in performance on average for every model tested in comparison to the results in Table 3.4. As found in the previous experiment, the proposed model, Design D, still outperforms all models tested. The p value of 0.005 calculated using the KW test in Table 3.8 suggests that the experiment groups are significantly statistically different and are unlikely to be a result of randomness.

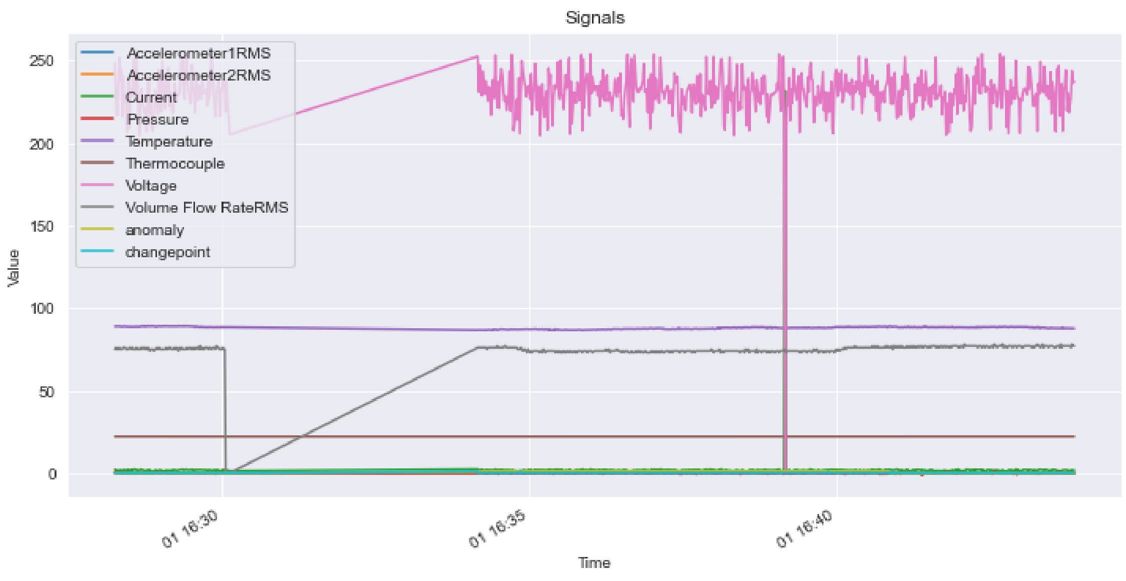
There are various factors that contribute to this resilience. An aspect to consider is that the anomalies that occur as a result of faults will most likely be different from the anomalies that appear during the ”healthy” condition, and thus such anomalies from the latter category can be learned during the training of the model without significant impact on the anomaly detection ability as they could be considered as part of the “healthy” condition. Another aspect considered during model training was the use of the Huber loss function, which uses a combination of MAE and MSE depending on the magnitude of the loss value. This results in a model that is more robust to outliers than the most commonly used MSE, but considers them to some extent. In the case of extreme outliers in the training data or outliers that correspond to faulty operation, this data cannot be used, and is easily distinguished from this kind of data in either the data collection phase or the data analysis phase that precedes model fitting, and thus it is easy to avoid using such data to train a NN model.

SKAB Anomaly Benchmark

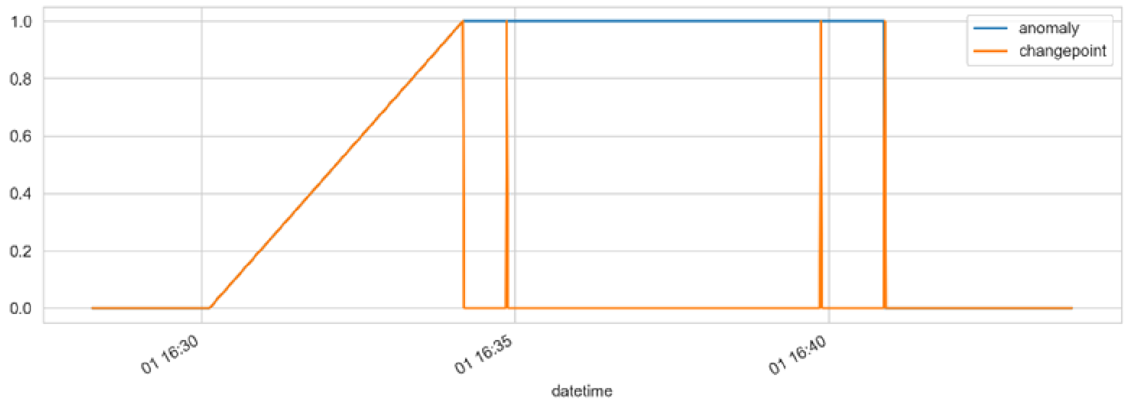
The SKAB anomaly detection benchmark [25] is a public benchmark available online used for offline outlier detection and changepoint detection testing. The benchmark consists of 35 subsets of data from a water circulation system, which contain 8 features, each from different sensors in the system. The test is conducted by looping through each subset, training the neural network on a slice of clean data from the subset, and then testing it on labelled anomalies that were simulated with the test rig.

Two primary metrics are used to assess the effectiveness of the models: the standard F_1 score for point-wise outlier detection (Equation 3.15) and the Numenta Anomaly Benchmark (NAB) score [145] to evaluate the detection of changepoints or anomalous ranges. The NAB score is designed for real-time, streaming data scenarios and employs a time-dependent scoring function that rewards early detections within the anomaly window while penalising late or false detections. It provides three predefined scoring profiles: standard, low false positive, and low false negative. This allows practitioners to prioritise different trade-offs. The final score is normalised relative to baseline models, including perfect detection and null detection, to provide a standardised measure of performance in identifying anomalous patterns or shifts in the data.

Figure 3.13a illustrates a subset of data from the benchmark, and Figure 3.13b shows the plot for the anomalies in the data.



(a) A subset of preprocessed data



(b) Plot representing the respective anomalies and changepoints in (a)

Figure 3.13: Visualisation of the SKAB Dataset.

This experiment aims to compare the anomaly detection and changepoint detection performance of popular and state-of-the-art unsupervised anomaly detection methods with the proposed NN model. A selection of NNs and ML-based fault detection methods were chosen to compare on the benchmark with minimal hyperparameter optimisation applied.

During model testing it was found that there were different hyperparameter values that could be used to optimise the proposed model for the outlier detection and changepoint detection tasks, respectively. This meant that hyperparameters optimal for a good F_1 score would not necessarily perform as well on the NAB score. To demonstrate this, the hyperparameters of the LSTMCaps NN were optimised for each score separately in order to achieve the maximum score for each metric. The different hyperparameter settings used are shown in Table 3.9. Furthermore, Z-score normalisation (Equation (3.16)) was used in this case as it was found to improve the performance for the proposed method.

Table 3.9: Optimal hyperparameters used to maximise outlier detection (left) and changepoint detection (right).

Hyperparameter	**LSTMCaps Optimised for Outlier Detection (LSTMCaps Outlier Detector)**	**LSTMCaps Optimised for Changepoint Detection (LSTMCaps Changepoint Detector)**
Optimiser	Amsgrad [146]	Adam [144]
MAE Threshold Multiplier	0.925	0.99
Epochs	100	
Learning rate	0.003	
Time steps	3	
Capsule activation	relu	
LSTM activation	tanh	
Validation split	0.2	
Batch size	128	
Branched layer width	32	
Full layer width	256	
Loss function	huber	

The same testing procedure utilised in the SKAB benchmark's GitHub repo [25] for each model already tested on the benchmark was used to test the proposed model architecture. The model was trained with 100 epochs on a subset from each dataset with early stopping set at a patience of 20, and then tested on the remainder of the dataset. The F_1 scores and NAB scores achieved for each dataset are averaged, which gives the final score of the benchmark. Each model compared was also tested on the same hardware for a fair comparison. To gain a better understanding of the effectiveness of each method tested as a hybrid solution for both outlier and changepoint detection, a scaled average of both the F_1 and NAB metrics was calculated to represent the overall accuracy of the model over both outlier detection and changepoint detection tasks. This was done by scaling the NAB score between 0 and 1, then averaging it with the F_1 score. Equation (3.17) was used to calculate this score. The results in Table 3.10 depict the average outlier detection score, the changepoint detection score, and the scaled average score over five test iterations, respectively, and the results in Table 3.11 detail the best score achieved in a single test iteration over the outlier, changepoint, and scaled average scores, respectively. A scatter plot of the average F_1 scores against NAB scores for each model tested is illustrated in Figure 3.15. The KW H-test of the Overall Accuracy metric of each model is calculated in Table 3.12.

$$Overall\ Accuracy = \frac{F_1 + \frac{NAB}{100}}{2} \tag{3.17}$$

Table 3.10: Comparison of the average scores out of five runs for each metric tested.

Algorithm	F_1	**FAR, %**	**MAR, %**	**NAB (standard)**	**NAB (lowFP)**	**NAB (LowFN)**	**Overall Accuracy**
Perfect score	*1*	*0*	*0*	*100*	*100*	*100*	*1*
LSTMCaps Changepoint Detector (Proposed)	**0.71**	**14.45**	**30.86**	**27.39**	**17.08**	**31.13**	**0.49195**
MSCRED [147]	0.7	16.82	31.28	26.13	17.81	29.53	0.48065
LSTMCaps Outlier Detector (Proposed)	0.74	21.66	18.74	21.58	5.12	27.49	0.4779
LSTM [148]	0.65	14.89	39.4	26.61	11.78	32	0.45805
LSTM-AE [149]	0.64	14.81	39.5	22.97	20.95	23.93	0.43485
MSET [150]	0.73	20.82	20.08	12.71	11.04	13.6	0.42855
Isolation forest [151]	0.4	6.86	72.09	37.53	17.09	45.02	0.38765
Conv-AE [152]	0.66	5.57	46.16	11.12	10.35	11.77	0.3856
LSTM-VAE [153]	0.56	9.04	54.75	21.09	17.52	22.73	0.38545
Autoencoder [154]	0.45	7.52	66.59	15.65	0.48	21	0.30325
Null score	*0*	*100*	*100*	*0*	*0*	*0*	*0*

Table 3.11: Comparison of the best score out of five runs for each metric tested.

Algorithm	F_1	FAR, %	MAR, %	NAB (standard)	NAB (lowFP)	NAB (LowFN)	Overall Accuracy
Perfect score	*1*	*0*	*0*	*100*	*100*	*100*	*1*
LSTMCaps Changepoint Detector (Proposed)	**0.71**	**14.51**	**30.59**	**27.77**	**17.14**	**31.59**	**0.494**
LSTMCaps Anomaly Detector (Proposed)	0.74	21.5	18.74	24.02	8.14	29.6	0.490
MSCRED [147]	0.7	16.2	30.87	24.99	17.9	27.94	0.475
LSTM [148]	0.67	15.42	36.02	26.76	12.92	31.93	0.468
LSTM-AE [149]	0.65	14.59	39.42	24.77	22.69	25.75	0.449
MSET [150]	0.73	20.82	20.08	12.71	11.04	13.6	0.429
LSTM-VAE [153]	0.56	9.2	54.81	21.92	18.45	23.59	0.390
Isolation forest [151]	0.4	6.86	72.09	37.53	17.09	45.02	0.388
Conv-AE [152]	0.66	5.58	46.05	11.21	10.45	11.83	0.386
Autoencoder [154]	0.45	7.55	66.57	16.27	1.04	21.62	0.306
Null score	*0*	*100*	*100*	*0*	*0*	*0*	*0*

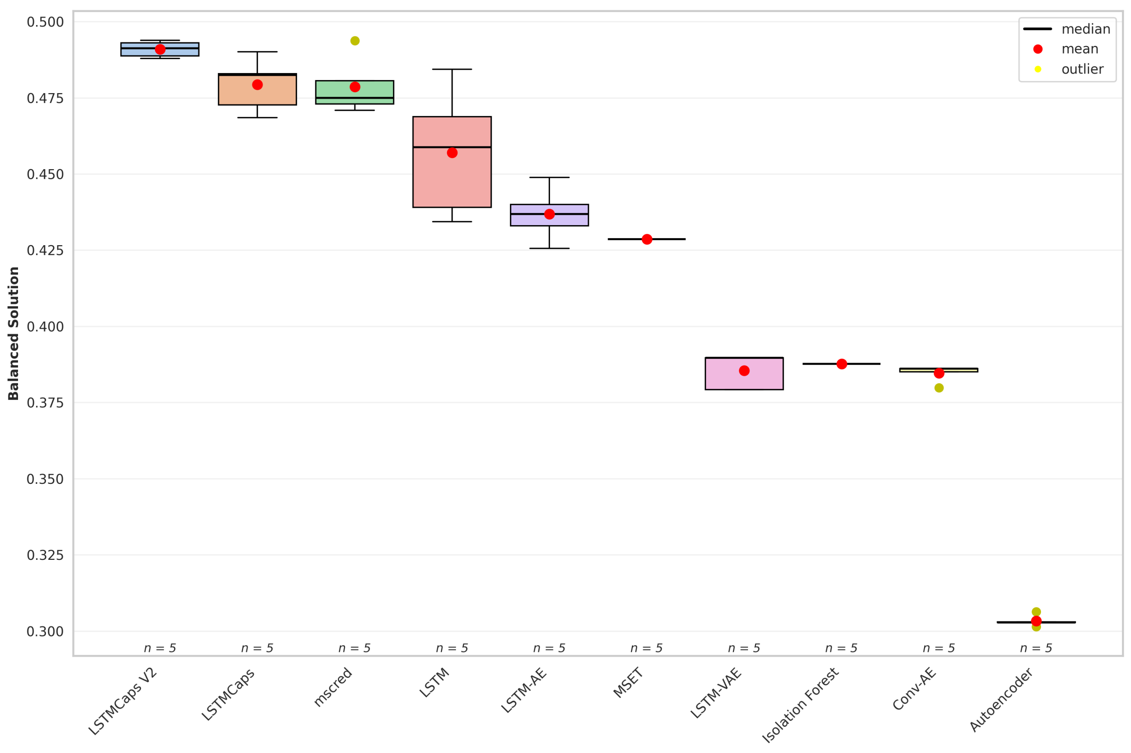


Figure 3.14: Box plot of the balanced solution scores for the SKAB anomaly benchmark experiment.

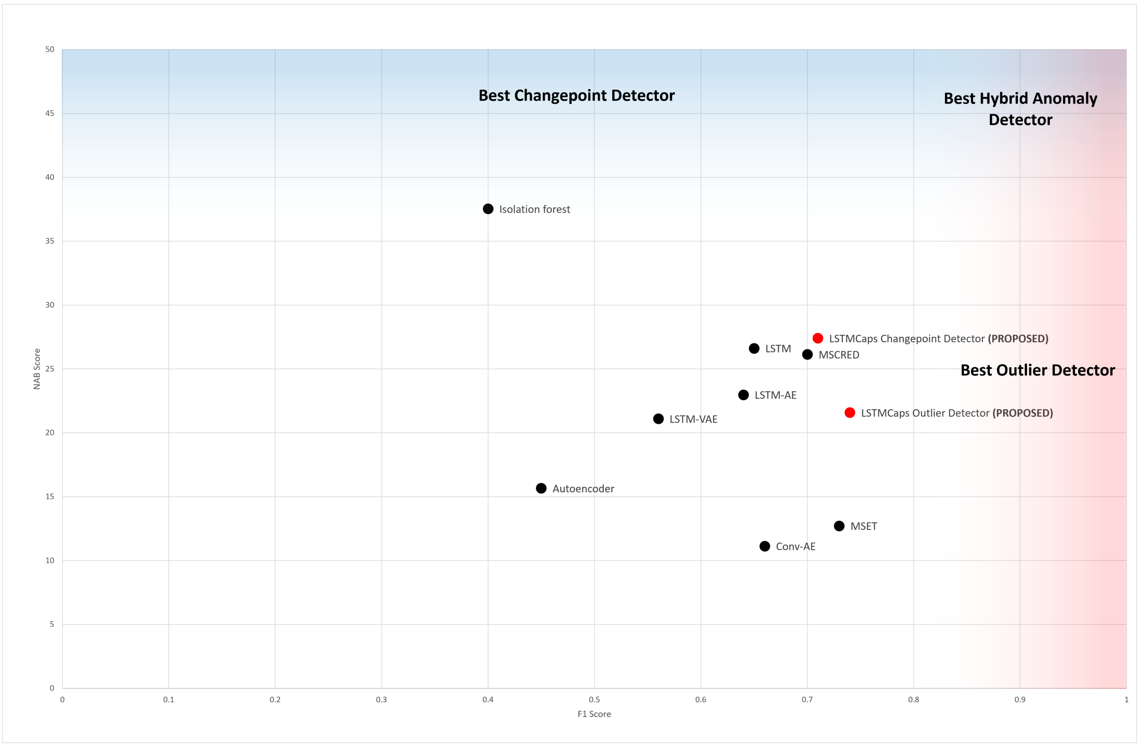


Figure 3.15: Visualisation of SKAB results: NAB score plotted against the F_1 score.

Table 3.12: Kruskal-Wallis test results for SKAB benchmark.

Model	n	Mean	SD	Median	Chi-square	*p*-Value
LSTMCaps V2	5	0.49	0.00	0.49		
LSTMCaps	5	0.48	0.01	0.48		
mscred	5	0.48	0.01	0.47		
LSTM	5	0.46	0.02	0.46	46.05	0.000*
LSTM-AE	5	0.44	0.01	0.44		
MSET	5	0.43	0.00	0.43		
Isolation Forest	5	0.39	0.00	0.39		
LSTM-VAE	5	0.39	0.01	0.39		
Conv-AE	5	0.38	0.00	0.39		
Autoencoder	5	0.30	0.00	0.30		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis reveals numerous significant pairwise differences, including: Autoencoder-Conv-AE, Autoencoder-Isolation Forest, Autoencoder-LSTM-AE, Autoencoder-LSTM-VAE, Autoencoder-LSTM, Autoencoder-mscred, Autoencoder-MSET, Autoencoder-LSTMCaps, Autoencoder-LSTMCaps V2, and others. A total of 37 pairwise differences were found to be significant using uncorrected p-values.

The results in Tables 3.10 and 3.11 show that the proposed method optimised for outlier detection outperforms all other methods tested, achieving the best F_1 score and the lowest False Negative rate out of the models tested. It also achieves the second highest False Positive rate out of the models.

With regards to changepoint detection, the proposed method optimised for outlier detection does not perform as well. However, the proposed method optimised for changepoint detection was able to outperform all ML-based methods and the majority of other methods, but is outperformed by the Isolation Forest algorithm due to the low False Negative rate of the Isolation Forest on the test data. However, Isolation Forest performs poorly with regards to outlier detection, making the algorithm unsuitable as a balanced solution for both problems. Similar outcomes can be seen for the best performing test iteration, with no improvement in relation to the other NNs and ML methods.

As a balanced solution to both outlier detection and changepoint detection, the proposed model outperforms all other tested methods when optimised for changepoint detection. The main change that is made to optimise the proposed model for each detection task is a change in the threshold multiplier. With a more sensitive threshold, the proposed model is able to display strong performance in outlier detection at a cost to the changepoint detection performance. Decreasing the sensitivity of the threshold to a value close to the maximum prediction MAE of the validation set allows the model to perform stronger on changepoint detection to the detriment of outlier detection ability. The loss of outlier detection performance in the configuration optimised for changepoint detection is comparably less in comparison to the loss in performance on changepoint detection when optimised for outlier detection. The p-value of 0 calculated for the means of the overall accuracy of the methods tested suggest that there sufficient evidence to state that the results of this experiment are highly unlikely to be due to random chance. Such a low value is due to some of the tested methods being deterministic and thus having no variance in the sample, and the stochastic methods such as ML-based method converging consistently to the same minima.

From this test, it can be concluded that for single datapoint outlier detection, the proposed LSTMCaps multi-channel architecture provides state-of-the-art performance. However, while the changepoint detection performance is superior to other ML-based methods with the right adjustments to the hyperparameters, the Isolation Forest algorithm is found to be advantageous for this benchmark. As a balanced solution however, the proposed model outperforms all other methods, and would be the preferred solution for strong performance in both outlier detection and changepoint detection simultaneously.

3.3 Discussion

Across the experiments conducted, it is clear to see that both the inclusion of the Capsule Network and the multi-channel input architecture is integral to the improvement of the performance of the proposed method in terms of training and anomaly detection. The evidence for this is shown clearly across the experiments, where with standard LSTM AEs, the training and anomaly detection performance is significantly weaker than with the proposed NN.

With regards to the scalability of the proposed method, whilst it is true that when keeping the model parameters constant and increasing the number of branches to account for more features results in a larger and more computationally complex model to train, the experimental results in Sections 3.2.6 and 3.2.6 have empirically shown that the model size can be reduced to a number of trainable parameters similar to single input NN architectures and still outperform the latter with both training efficiency and anomaly detection. This gives flexibility with regards to the size of the proposed model depending on the complexity of the data being applied on, but mitigates the effect of additional features on model complexity.

The experimental results further suggest that models which included Capsules were training more efficiently, reaching the local minima at a faster rate in relation to networks without Capsules. Most importantly, the results in Table 3.3 for training suggest that with the use of Capsules, the model training procedure can be simplified considerably due to the lack of overfitting during training on the NN models with Capsules integrated. It was also found during hyperparameter optimisation that even without fully optimised hyperparameters, the model variants that included Capsules were less susceptible to overfitting in comparison to the model variants without Capsules.

One significant strength of the proposed LSTMCaps NN is the ability to learn separate data features effectively in comparison to a standard single channel NN. Evidence of this is seen in Table 3.7, where both multi-channel input architectures with and without Capsules perform better with anomaly detection. This is further substantiated with the anomaly detection performance on the SKAB anomaly benchmark, which contains a larger number of more complex features than the drone data. Whilst literature mentions that correlation dependencies between features are not considered when learning each feature separately [155], the proposed model overcomes this potential drawback by concatenating the internal feature representations and utilising a layer of Capsules to learn the spatial features of the multivariate data, which includes the correlation dependencies. Evidence of this is also clearly shown empirically through the experimentation conducted on the drone dataset where the proposed model outperforms the multi-channel input variant without Capsules with standard training data and training data with outliers.

The proposed model tackles overfitting with its various aspects of operation. In the experiments completed on the drone dataset, each model tested uses a comparable number of trainable parameters. However, since the multi-channel models contain an encoder for each feature as opposed to a single encoder, more layers are used. This is a result of each input channel encoding the data separately at first, which means that less trainable parameters are needed for each input channel. Previous research clearly shows that increasing the number of parameters allows for a NN model to be able to model more complex functions and relationships, but at the risk of overfitting on the distribution of the training set. The proposed approach shows that by using the multi-channel input approach, the model can maintain the number of parameters as the single channel variants outperform the latter in both training—but more importantly anomaly detection—due to the enhanced generalisation ability achieved through more robust training. The drone dataset experiments in Section 3.2.6 do not show this behaviour, but this is due to the lack of complexity in the dataset. On the other hand, the benchmark on the SKAB dataset in Section 3.2.6 shows this behaviour clearly, due to a significantly increased number of features, signal length, and complex signal behaviour.

In addition to this, by comparing the results in Tables 3.3 and 3.6, the models using Capsules are clearly able to train more effectively, and in some cases perform slightly better on the validation set in comparison to the training set, showing strong generalisation ability. Since the generalisation ability of a NN is directly correlated with the training performance, there is clear evidence to say that the Capsule directly contributes to the strong training performance of the proposed model, and reduces the overfitting generally encountered when training autoencoders. In comparison to recent state-of-the-art approaches, the proposed method overcomes the limitations found in each work. For example, the use of Capsules instead of traditional Convolutional layers used in [81] is shown empirically on the SKAB dataset to improve performance, and reduce overfitting without additional measures. Furthermore, the multi-channel input architecture is rarely used in anomaly detection tasks, and not used in any of the literature reviewed in the present work. However, it is empirically proven in all experiments in the present study that by using this approach, the model is able to learn the features more effectively when using the LSTM univariate encoders and Capsules for univariate decoding and multivariate spatial feature learning. With the proposed approach, there is potential to use heterogeneous input data with minimal modifications to the model architecture.

3.4 Conclusions and Future Work

This chapter proposes a novel hybridisation of the LSTM and Capsule Networks in a multi-channel architecture to address the issues found in existing literature with the training performance of NNs, specifically on multivariate data. The motivation for this chapter stemmed from the growing demand for more effective unsupervised data analysis techniques regarding outlier and anomaly detection for use in industrial and commercial environments with large datasets to assist in the advancement of Industry 4.0.

The proposed NN architecture was first tested on a drone dataset to observe the training and anomaly detection performance improvements with regards to non-hybridised and single channel variants of the NN, where it was found that, due to the inclusion of Capsules, the proposed NN can train more efficiently over a smaller number of epochs by converging at a faster rate in comparison to the variants with no Capsules integrated in the NN, and shows evidence of a resilience to overfitting. The tested NN models detected anomalies in the test data through an unsupervised method of reconstructing the validation data and using the maximum prediction MAE of the subset of data as a reference point for the confidence of prediction in any unseen data, so any data outlying from the expected shape in the training data would be flagged due to high prediction error. The results of this test concluded that the proposed NN architecture performs better than the other variants tested as a result of the proposed additions and changes to the NN architecture. The model variants were also tested on imperfect training data with outliers present, and all variants were found to be robust to outliers in the data due to the loss function used. Furthermore, due to the outliers present in the data not corresponding to faulty operation, the performance of the models on detecting faulty drone operation was not affected. It was also noted that training data that contains fault data would be easily spotted during the data collection or data analysis stages, and would therefore be easy to avoid when used to train the model.

The proposed NN was also tested against other popular and state-of-the-art anomaly detection methods on the SKAB anomaly detection benchmark, where with slight hyperparameter adjustments the proposed method was able to adapt effectively to both outlier detection and changepoint detection, performing better than all other methods tested for outlier detection. Whilst the proposed method performed better than most methods in changepoint detection, the model was outperformed by the Isolation Forest algorithm. However, the Isolation Forest performed poorly with outlier detection, making it highly unsuitable as a hybrid solution for outlier and changepoint detection simultaneously. On the other hand, the proposed model, when optimised for changepoint detection, outperformed all other methods as a hybrid solution to both outlier and changepoint detection problems.

An interesting observation from Tables 3.3 and 3.6 is the increase in training time with the approaches using capsules, in comparison to just LSTM layers. This clearly indicates the higher computational power required to train the Capsule, mainly due to the dynamic routing algorithm. Although the algorithm is a main driver of the strong feature and spatial learning performance of the Capsule layers, it is very computationally intensive. Recent work has shown that alternative architectures, such as the Homogeneous Vector Capsule (HVC) [57], can still achieve strong performance in spatial learning. Additionally, while not directly compared in this study, the LSTM cell structure is also computationally intensive compared to more modern approaches like the Gated Recurrent Unit. While these inefficiencies did not pose an issue in the present study, they may become more significant when dealing with larger datasets.

The applicability of the proposed NN architecture to IM data is a crucial next step in validating its effectiveness for industrial fault detection. However, applying the proposed method to multiple homogeneous IM data sources may require significant computational power for training, which could hinder its practical implementation. Building upon the foundations laid by the proposed NN architecture, Chapter 4 addresses this challenge by introducing the DF algorithm, a novel approach designed to merge multiple homogeneous, periodic TS datasets into a single unified dataset for training anomaly detection NN models while minimising the computational requirements for training.

Chapter 4

A Dataset Fusion Algorithm for Generalised Anomaly Detection in Homogeneous Periodic Time Series Datasets

4.1 Introduction

Generalisation is a measure of a NN's performance on data that it has not seen before but that is in the same class as the data that it has been trained on. The idea behind generalisation with Deep Learning (DL) is to transfer domain knowledge from data the NN has been trained on to unseen data in the same class, where the unseen data may contain conditions that slightly vary from the training data. This allows for a NN to be able to maintain performance across the dataset, and potentially transfer across multiple datasets with a similar distribution to the initial trained data. Various studies have been undertaken to understand the factors that affect the generalisation performance of a NN [156], and how to mitigate these factors to achieve the optimal level of performance [157]. The underlying concept is as follows: When training a NN on a data sample, the NN learns to represent the function between the input data and the output data through the adjustment of the weights and biases. If the distribution of the data sample used for training is not fully representative of the true distribution population, then the input-output function that is represented by this sample will inevitably vary from the function of the population. Extensive training on this sample will then result in a phenomenon known as overfitting [158], which refers to when the NN has accurately modelled the function represented by the training data but is not able to generalise to data in the same class due to the discrepancy between the functions represented by the sample and population.

Many works have been published with the aim of addressing overfitting and thus maximising the potential generalisation ability of a NN. Some works focus on architectural improvements to the NN to increase the robustness of the NN to overfitting through novel architectures such as CapsNet [2], whilst other research directions focus on the manipulation of the training procedure to limit overfitting with techniques such as Dropout [89], Early Stopping [159], Pruning [93] and adding noise to the weights and biases of the NN whilst training [160].

There is much less focus on research concerning the effect of the composition and specification of the dataset on generalisation, especially regarding TS data. A common approach currently used includes denoising the training sample to better align the distribution of the sample to the population and hence limit the level of overfitting [161][162]. Additionally, there is a consensus in ML research that increasing the volume of data through various means such as augmentation [109][110] improves NN generalisation performance; whilst this has been empirically confirmed, more recent research has discovered that this improvement largely comes specific samples within the supplementary data, and a significant volume of this data is essentially redundant and does not contribute to a performance improvement [163][164].

Furthermore, there is a gap in literature concerning the fusion of multiple TS datasets in a single training set to balance the probability distribution of the training sample so that it better aligns with the true distribution of the problem domain. This is largely due to a lack of necessity in an experimental environment since most ML research tends to optimise the solution for a single dataset source. However, from a commercial standpoint, this can have many benefits with regard to time and computational power saved, as well as the added benefit of reducing the data requirements for training. In addition to this, the dynamic shifting of the distribution of data is often a bottleneck to the performance of the NN; this is a prevalent issue that is encountered when a NN is deployed in a non-stationary environment, which is common with TS data. Some recent works have detected this shift [165], and mitigate the effect this has on the generalisation performance of the NN with both TS data [166][167] and image data [168]. However, the majority of empirical evaluations of NN approaches in literature are mostly conducted on an isolated sample of data, which, in many cases, is not representative of the dynamic shifting of the distribution temporally.

To address the identified gaps in the literature, a novel algorithm is proposed, named Dataset Fusion (DF). The proposed method merges multiple homogeneous, periodic TS datasets into a single unified dataset for training anomaly detection NN models. The fusion process is designed to accurately represent population distributions and increase robustness against potential data distribution shifts. The primary objective in this study is to examine efficient generalisation approaches that can minimise training time and computational demands for neural networks when working with new homogeneous TS data sources. The contributions of this chapter can be summarised as follows:

- A novel dataset composition algorithm is proposed, referred to as *Dataset Fusion*
- The proposed approach is applied to a case study focused on motor current data, with a qualitative analysis conducted to assess the preservation of features from each individual dataset.
- The generalisation performance of the proposed method is empirically evaluated in anomaly detection with the LSTMCaps neural network architecture from previous work [2], and compared to the performance when using conventional training approaches
- The potential practical limitations of the proposed method in a real-world environment are discussed and assessed through further experimentation

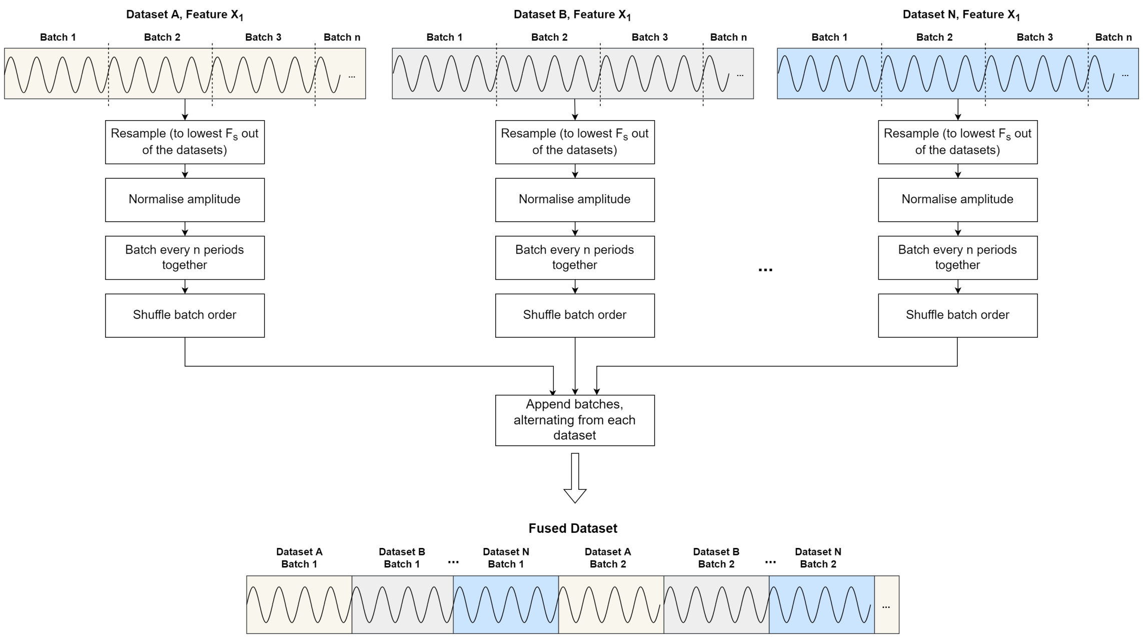


Figure 4.1: A summarised illustration of the DF algorithm

In the context of the sustainable development goals (SDGs), this study primarily focuses on Goal 9 (Industry, Innovation, and Infrastructure) through the exploration of innovative approaches to improve the performance and efficiency of neural network models; Goal 12 (Responsible Consumption and Production) through the proposal of a method that demands less computational power and training data, and Goal 13 (Climate Action) through the resulting reduction in the energy consumption for model training.

4.2 Dataset Fusion

A summary of the DF process is depicted in Figure 4.1.

The signal in each dataset is first down-sampled to the sampling frequency, F_s, of the dataset with the lowest sampling frequency. This step is essential to models with look-back such as Recurrent Neural Networks so that the same signal length is considered for every motor when training the model. Taking the example in Figure 4.1, for a set of signals $\{A[n], B[n], \ldots, N[n]\}$ with sampling frequencies $\{F_{s_A}, F_{s_B}, \ldots, F_{s_N}\}$, the target sampling frequency, $F_{s_{new}}$, is expressed in Equation 4.1.

$$F_{s_{new}} = min\{F_{s_A}, F_{s_B}, \ldots, F_{s_N}\} \tag{4.1}$$

The re-sampling is implemented using the Fourier method. This method was chosen over decimation due to the simplicity of implementation and with the assumption that the signals used are periodic in nature. To avoid aliasing and other artefacts, a low-pass windowed-sinc filter is first designed and applied to the signal, with a cutoff frequency based on the target Nyquist frequency. The Hann window, $hann(n)$, was employed in the filter design due to its desirable characteristics for resampling, such as reduced spectral leakage and smooth sidelobes. In the present work, 101 taps were used, giving a filter order of 100. Equation 4.2 expresses the impulse response, $h[n]$, of this filter.

$$h[n] = K \cdot \frac{\sin\left(2\pi f_c(n - M/2)\right)}{n - M/2} \cdot hann(n) \tag{4.2}$$

where K is the normalisation factor, f_c is the cutoff frequency in Hz, n is the discrete time index, and M is the filter length or number of taps.

To implement the Fourier Method, the finite-length TS signal, $x[n]$ (for $n = 0, 1, \ldots, N-1$), is first transformed into the frequency domain using the Discrete Fourier Transform (DFT). The DFT coefficients, $X[k]$, are given by Equation 4.3.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}} \quad \text{for } k = 0, 1, \ldots, N-1 \tag{4.3}$$

where k is the discrete frequency index, N is the length of the original signal, and $j = \sqrt{-1}$. These coefficients $X[k]$ represent the frequency content of the signal at discrete frequencies $f_k = k\frac{F_s}{N}$.

The resulting discrete spectrum $X[k]$ is then manipulated for resampling. This involves selecting the frequency components corresponding to frequencies below the target Nyquist frequency ($\frac{F_{s_{new}}}{2}$), and zero-padding or interpolating these components appropriately to create a new spectrum with N_{new} points, suitable for an inverse transform of that length. Let these resampled DFT coefficients be denoted as $X[k]_{resampled}$. The new number of time-domain samples, N_{new}, is calculated using Equation 4.4. Equation 4.5 shows the Inverse Discrete Fourier Transform (IDFT) operation used to obtain the resampled TS signal $x[n]_{resampled}$ from these modified coefficients.

$$N_{new} = \frac{N}{F_s} \times F_{s_{new}} \tag{4.4}$$

$$x[n]_{resampled} = \frac{1}{N_{new}} \sum_{k=0}^{N_{new}-1} X[k]_{resampled} e^{\frac{j2\pi kn}{N_{new}}} \tag{4.5}$$

Each dataset is then normalised using Z-score normalisation, to overcome varying motor currents. For the resampled sequence, $x[n]_{resampled}$, the normalised sequence, $x'[n]$, is calculated using Equation 4.6.

$$x'[n] = \frac{x[n]_{resampled} - \bar{x}_{resampled}}{\sigma_{x_{resampled}}} \tag{4.6}$$

where $\bar{x}_{resampled}$ is the mean of the resampled sequence, and $\sigma_{x_{resampled}}$ is the standard deviation of the re sampled sequence.

To batch the periods together, a zero-crossing algorithm, configured to detect crossings from positive to negative, is employed to first identify a single period, and then concatenate n periods based on the user-defined parameter. Given that z-score normalisation is utilised, any periodic time-series data will exhibit sign changes, making the zero-crossing algorithm applicable. In cases where multiple features are present in the data, the zero-crossing algorithm calculates only the first feature as a reference for period batching, in order to maintain the temporal integrity of the data and preserve the spatial relationship between features. The impact of varying batch sizes on training performance differs based on the nature of the data and problem domain; hence, it is recommended that this parameter be optimised alongside other training hyperparameters. For a discrete periodic signal $x[n]$ with assumed periodic sign changes, the set of indices for the positive-to-negative zero crossings, $c_{+\to-}$, can be expressed as shown in Equation 4.7.

$$c_{+\to-} = \{\, n \mid x[n-1] > 0 \geq x[n] \,\} \tag{4.7}$$

For each dataset, the batch order is then shuffled randomly. This is done in order to mitigate the effect of distribution shift and prevent noise in one area of the signal from being prevalent in other areas of the signal. In other words, this step helps to reduce the variance of the NN prediction. A new signal is then constructed using the shuffled batches from each dataset by appending a batch from each dataset in an alternating fashion.

The full process of the DF algorithm is expressed in Algorithm 1.

Algorithm 1 Pseudocode of DF Algorithm

```
1: function DATASET_FUSION($x, F_s, P$)
2:     Input:
3:         A set of $s$ finite discrete periodic sets $x$ where $s > 1$
4:         A set of sampling frequencies $F_s$ corresponding to $X$
5:         Number of periods batched $P$
6:     Output: $x_{fused}$
7:     Determine $F_{s_{new}}$ using Equation 4.1
8:     for $x_1$ to $x_s$ do
9:         if $F_{s_{X_s}} \neq F_{s_{new}}$ then
10:            Apply filter in Equation 4.2
11:            Calculate $X[\omega]$ using Equation 4.3
12:            Calculate $X_{[\omega]_{resampled}}$
13:            Calculate $N_{new}$ using Equation 4.4
14:            Calculate $x[n]_{resampled}$ using Equation 4.5
15:        end if
16:        Calculate $x'$ using Equation 4.6
17:        Calculate $c_{+\to-}$ using Equation 4.7
18:        Calculate $x'_{batched}$ through grouping $P$ periods by slicing $x'$ at the values where $c_{+\to-}[n]\%P = 0$
19:        Shuffle $x'_{batched}$
20:    end for
21:    $x_{fused} = \{x'_{1_{batched}}[0] \mathbin{+\!\!+} ... \mathbin{+\!\!+} x'_{s_{batched}}[0] \mathbin{+\!\!+} x'_{1_{batched}}[1] \mathbin{+\!\!+} ... \mathbin{+\!\!+} x'_{s_{batched}}[1] \mathbin{+\!\!+} ...\}$
22:    return $x_{fused}$
23: end function
```

where % represents the modulo operator and $\mathbin{+\!\!+}$ represents concatenation.

4.2.1 Computational Complexity

The computational complexity of the DF algorithm, represented in Big O notation, can be determined by breaking down the steps of the algorithm when practically applied. The breakdown of the complexity of each stage is provided in Table 4.1.

Table 4.1: Algorithm Complexity for DF Algorithm, for n Datasets with m length

Algorithm step	Big O Complexity
Filtering and Resampling	$O(nm(1 + log(m)))$
Normalisation	$O(nm)$
Period Batching	$O(nm)$
Chaining and Stacking batches	$O(nm)$
Total	$O(nm(1 + log(m))) + 3O(nm)$

As Table 4.1 shows, the filtering and resampling step has a complexity of $O(nm(1+log(m)))$, since it involves applying a finite impulse response (FIR) filter with a complexity of O(m) and performing resampling using the FFT with a complexity of $O(m*log(m))$ for each of the n datasets. The Normalisation step scales each dataset using Z-score normalisation with a complexity of $O(m)$ for each dataset, resulting in a total complexity of $O(n*m)$. The Period Batching step identifies zero-crossings and creates period batches with a complexity of $O(m)$ for each dataset, resulting in a total complexity of $O(n * m)$. Finally, the Chaining and Stacking batches step involves filtering, chaining, and stacking the period batches with a total complexity of $O(n * m)$. The overall complexity of the DF algorithm is the sum of the complexities of these steps, which is $O(n * m * (1 + log(m))) + 3 * O(n * m)$, with the dominating term being $O(n * m * log(m))$ due to its faster growth as the input size (n and m) increases.

The logarithmic factor in the dominating term, $O(n*m*log(m))$, makes the DF algorithm scale well with increasing input size. This is because logarithmic growth is slow growth, ensuring that the algorithm remains efficient even as the number and size of the datasets (n and m) increase. Additionally, since the complexity is dependent on both the number of datasets (n) and the length of the datasets (m), the algorithm can efficiently handle varying dataset sizes and compositions. This scalability makes the DF algorithm a versatile algorithm and suitable for processing large and diverse datasets, which is essential in the context of real-world applications where data size and complexity are constantly evolving.

4.2.2 Requirements for application

Whilst the proposed algorithm is domain-independent, there are requirements regarding the data that must be met for the proposed method to be applicable. These requirements, as well as the reasoning, are detailed in the following sections.

Homogeneous Datasets

Although the methodology can be used in varying problem domains, the DF algorithm can only fuse homogeneous data, since the aim of the algorithm is to capture the data distribution of a problem domain as a whole in order to mitigate overfitting on a specific dataset. Generalisation to multiple problem domains is not in the scope of this algorithm.

Data periodicity

As explained in section 4.2, The algorithm relies on the fact that the data is periodic, due to the resampling method used, the zero crossing method, and to be able to create a coherent and usable sequential fused TS dataset.

Time Domain Data

The proposed approach will only be applicable in the time domain representation of the datasets, as it relies on the sequential nature of the data to fuse it together in a meaningful way

If the datasets being fused meet the requirements detailed above, then there is feasibility in applying the proposed method. Some examples of where the proposed method may be feasible are daily temperatures in a region, electrical power data and vibration data.

4.2.3 Proposed Benefits of Dataset Fusion

The DF algorithm seeks to eliminate the necessity of multiple NNs for a single problem domain. This approach theoretically enables the development of an NN that can adapt to unseen data from the same domain, even if originating from different data sources. Moreover, the DF algorithm aims to reduce data requirements from individual sources, as achieving ideal data collection conditions from each source can often prove to be challenging. Potential issues with collected data, such as data corruption, sensor faults, or insufficient data volume, among other data collection complications, further emphasise this need. The present study will experimentally investigate the proposed benefits of the DF algorithm.

4.3 Case Study: Dataset Fusion for 3-phase motor current data

This section will explore the feasibility of applying the proposed method with a case study on motor current signals. The aim of the case study is to empirically test and validate the effectiveness of the proposed method. The datasets used will first be introduced and the feasibility of the proposed method will be confirmed. The DF algorithm will then be applied, and the resulting signal will be compared and analysed to the original signals.

4.3.1 Dataset Introduction

For the present case study, two homogeneous open-source datasets [27] [26] will be used to confirm the feasibility of the DF methodology. Specifications of the datasets used are detailed in Table 4.2. Both datasets are composed of three-phase motor current signals, however, one dataset, which will be referred to as Dataset A, contains fault data for an inter-turn short circuit fault, and the other dataset, which will be referred to as Dataset B, contains current signals for a broken rotor bar fault.

Table 4.2: Specification for Motor Datasets used in the case study

Dataset Name	Data volume per file (samples, features)	Faulty Data Files	Healthy Data Files	Sampling Frequency (Hz)	Duration per file(s)
Dataset A: Inter-turn short circuit fault dataset (Cunha, 2021)	(100,000, 3)	2264	353	10,000	10
Dataset B: Broken Rotor Bar Dataset (Maciejewski, 2020)	(1,001,000, 3)	320	80	55,611	18

Table 4.3: Breakdown of Dataset A 60Hz files

Motor State	Number of files	Samples per feature
Healthy	48	4,800,000
High Impedance 1 (1.41% of stator winding)	53	5,300,000
High Impedance 2 (4.81% of stator winding)	52	5,200,000
High Impedance 3 (9.26% of stator winding)	48	4,800,000
Low Impedance 1 (1.41% of stator winding)	55	5,500,000
Low Impedance 2 (4.81% of stator winding)	54	5,400,000
Low Impedance 3 (9.26% of stator winding)	60	6,000,000

Dataset A

Dataset A [26] contains files from a motor running at a variable operating frequency F_o, ranging from 30Hz to 60Hz with 5Hz increments. The motor has the following specifications: 4 poles, 1HP mechanical power, 220V supply and 3A rated current. The authors simulated both high-impedance and low-impedance short circuits, for different levels of fault severity. For the purpose of this case study, only the files captured at $F_o = 60Hz$ were used to meet the limitation of DF of only being applicable to homogeneous data. The new dataset structure is depicted in Table 4.3.

Dataset B

The motor used to capture Dataset B [27] is a squirrel cage AC motor, running at a constant $F_o = 60HZ$ and has similar specifications to the motor used to capture Dataset A. The breakdown of the dataset is shown in Table 4.4. At the beginning of each file, for roughly the first 4 seconds, a transient signal representing the motor startup was also recorded. For the purpose of the case study, and for compatibility with the proposed algorithm, the transient subset of the signal, the first 200,000 samples, was discarded from each file, so that only the steady state of the motor remained. This left 801,000 samples left in each file, representing an approximate 20% redundancy of data.

4.3.2 Application of Dataset Fusion and Analysis

The DF algorithm was used to fuse the healthy files from Datasets A and B into a single, fused dataset. First, all healthy files were extracted from each dataset

Table 4.4: Breakdown of Dataset B

Motor state	Number of files	Samples per feature
Healthy	80	1,001,000
1 Broken Bar	80	1,001,000
2 Broken Bars	80	1,001,000
3 Broken Bars	80	1,001,000
4 Broken Bars	80	1,001,000

and concatenated into a large signal. The files in Dataset B were first sliced to remove the motor startup signature, then resampled to 10,000Hz, the same F_s as Dataset A. Each Dataset was split into batches of 4 periods and then concatenated alternating between each Dataset to create the final fused dataset. This was found to be the optimal value for this dataset through a grid-search-based optimisation of the parameters for DF to maximise NN performance. The batches were then concatenated, alternating between each dataset, to create the final fused dataset.

For each dataset, an initial analysis was conducted in order to understand the data and enable a more accurate interpretation of the fused dataset experimental results. The TS signal, a Probability Distribution Function (PDF) and FT representations from 0-500Hz were generated for a single phase from a healthy file from each dataset. The plots are illustrated in Figure 4.2.

A Principal Component Analysis (PCA) was also performed on the healthy data from each dataset as well as the fused data to gain comprehensive insights into the data, wherein the resulting axes are linear combinations of the original variables, defined by the eigenvectors and eigenvalues. This method allows for the identification of the most significant patterns to increase the interpretability of the proposed method. All datasets were uniformly downsampled to 10,000Hz, which corresponds to the minimum sampling frequency in Dataset A. Subsequently, the samples were partitioned into groups of 100,000, aligning with the smallest sample size per file in Dataset A. The data's three features, representing the three phases, were flattened into a single axis before being subjected to the PCA algorithm. The visualisation of the first two Principal Components in a 2D scatter plot can be found in Figure 4.3.

It is clear to see from Figure 4.2(d) and Figure 4.2(e), as well as Figure 4.2(g) and Figure 4.2(h) that Dataset B contains a considerable amount of noise in comparison to Dataset A. In addition to this, the frequency spectra of Dataset A show more pronounced harmonics in comparison to Dataset B. Although this may not be as evident from the TS signal plot, a NN will most likely pick up these differences in noise, and thus a NN trained on a single dataset, especially in the case of Dataset B, will struggle to distinguish files from Dataset A with fault signals as anomalous. This hypothesis will be further discussed in the results and discussion.

The TS signal of the fused data looks similar visually in comparison to the datasets, albeit in a different input space due to normalisation. From the PDF and FFT representations shown in Figure 4.2(f) and Figure 4.2(I) however, there are subtle indications of features present in both Dataset A and Dataset B. For instance, the overall shape of the frequency spectra follows Dataset A, however, there is noise clearly present in the spectra, a significant feature of Dataset B. Furthermore, the harmonic peaks in the fused frequency spectra contain the same characteristics of both datasets, which is interesting to note as this representation

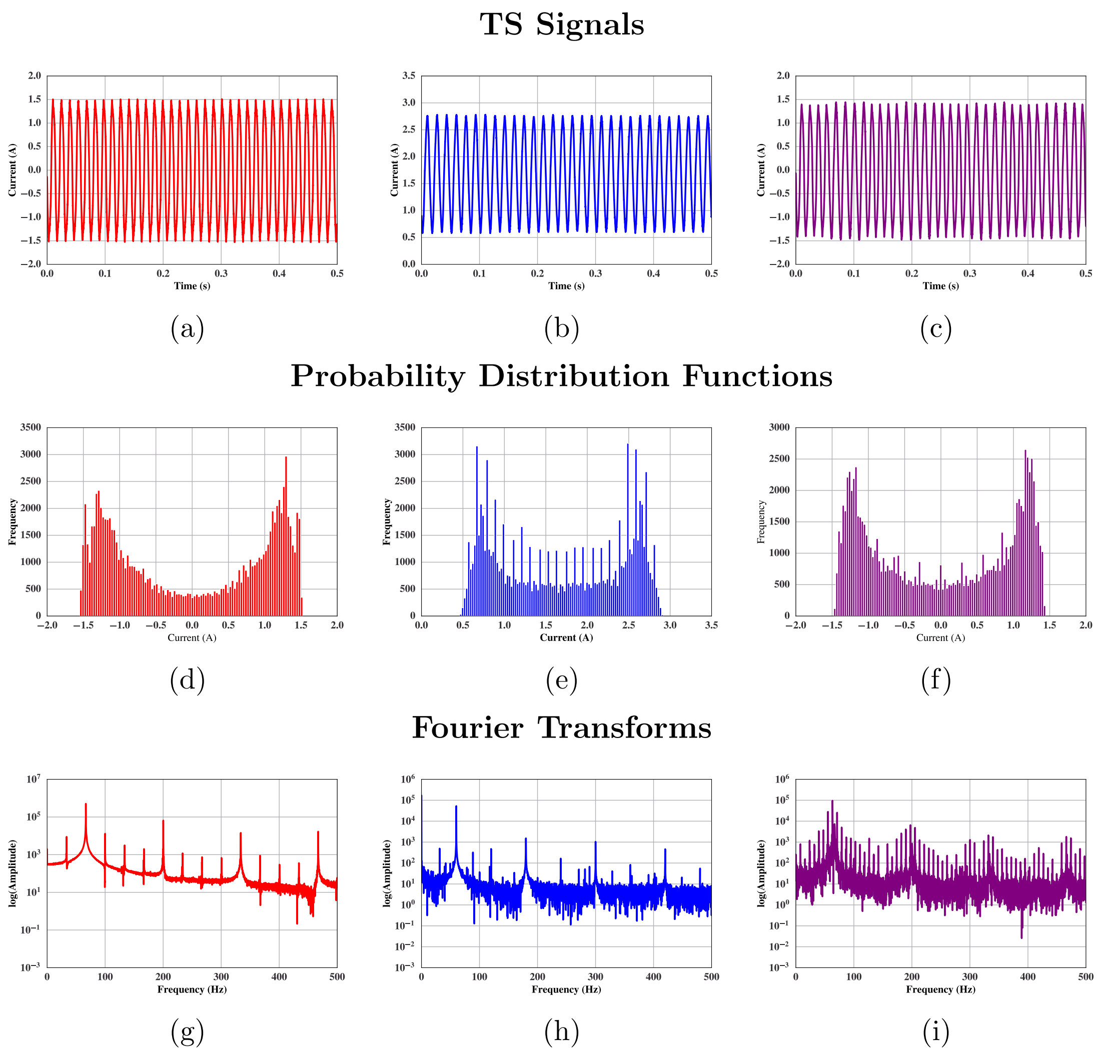


Figure 4.2: TS signal from Dataset A (a), B (b), and fused (c), Probability Distribution Function from Dataset A (d), B (e), and fused (f), and Fourier Transform from Dataset A (g), B (h), and fused (i) healthy files.

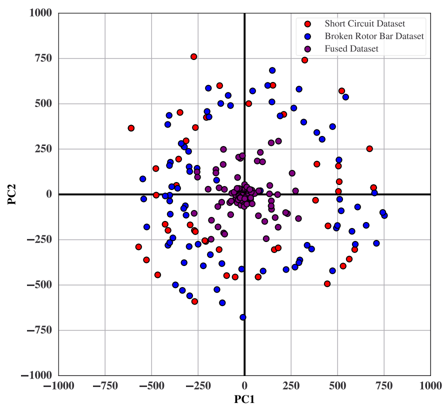


Figure 4.3: Principal Component Analysis of Dataset A, B and the fused dataset

would still be considered a healthy signal. Future work will investigate the use of a fused Dataset in the frequency domain to train a classifier NN. However, the scope of this study is to validate the use of the TS representation to train a generalised TS anomaly detector with reduced data requirements.

Upon examining the PCA plot depicted in Figure 4.3, it is clear that the healthy data from both datasets exhibit comparable traits and patterns. Interestingly, the fused dataset forms a cluster around the origin, positioning itself at the center of the two datasets. This central location of the fused dataset within the circular arrangement of points from the two datasets signifies that it effectively captures the salient features of both datasets. By doing so, the fused dataset aids in bringing the training data closer to the population distribution of the problem domain, thereby enhancing the robustness and generalisability of the model derived from this data. Further evidence of this will be given in the experimental results.

It is important to note that simply concatenating two healthy files from each Dataset will produce a similar outcome to the representations shown in Figure 4.2. However, the purpose of this analysis is to show that the Dataset algorithm will still preserve the individual features of each representation in a new signal and still be usable for a data-driven approach. The PCA plot, as displayed in Figure 4.3, provides more compelling evidence of the impact of the DF technique on the combined dataset. Subsequent experimental results on anomaly detection, utilising the various datasets, will further explore the implications of employing a fused dataset for training an anomaly detection model.

4.3.3 Experimental Design

The aim of the experimentation presented in this study is to observe the effectiveness of DF in training an anomaly detector NN using some of the healthy files from Dataset A and Dataset B, and then evaluating the model on the remaining healthy and faulty files, in comparison to commonly used training methods. The training methods selected as baselines in this study are commonly utilised in the domain of anomaly detection and have been previously validated in literature [169] [170]. These methods serve as a standard against which the proposed DF method is compared. The following training methods will be compared for all of the subsequent experiments:

- **Traditional Approach**: This approach involves training on a single dataset. It's a common baseline method used in many studies [169].
- **Transfer Learning**: A two-phase training method where the first training phase occurs on one dataset, followed by a second training phase on another dataset. This method leverages knowledge transfer between datasets and has shown promise in related works [170].
- **Mixed Dataset**: A single training phase is conducted using all healthy files from each dataset. This method aims to leverage the diversity of multiple datasets.
- **DF**: The proposed method involves a single training phase on fused healthy data consisting of all datasets.

Table 4.5: Experiment variants and corresponding key for results tables

Experiment	Key
Traditional Approach - Dataset A	T - Dataset A
Traditional Approach - Dataset B	T - Dataset B
Transfer Learning - Dataset A to Dataset B	TL - Dataset A to B
Transfer Learning - Dataset B to Dataset A	TL - Dataset B to A
Mixed Dataset	MD
Dataset Fusion	DF
Transfer Learning - Fused Dataset to Dataset A	TL - DF to Dataset A
Transfer Learning - Fused Dataset to Dataset B	TL - DF to Dataset B

Table 4.6: Specifications for workstation used for experimentation

Component	Specification
Operating System	Windows 10 Version 21H2
CPU	AMD Ryzen Threadripper 2990WX 32-Core 3.5GHz
RAM	64GB
GPU	NVIDIA RTX A6000 48GB VRAM

- **DF with Transfer Learning**: Combines the DF and transfer learning approaches, with the first training phase on fused healthy data consisting of all datasets, followed by a second training phase on a single dataset.

To provide clarity on the variations within each training method, Table 4.5 provides a full breakdown of the different variants, along with labels used in the experimental results tables.

The same workstation was used to conduct all experimentation in order to maximise experimental rigour. The specifications of this workstation are given in Table 4.6, for the purpose of experiment reproducibility.

Each experiment iteration was repeated 10 times for experimental rigour. The outcome of each experiment is validated for statistical significance using appropriate statistical tests. Given the non-normally distributed results and unequal variances across the small sample groups (n=10 per condition), the non-parametric KW test was chosen over Analysis Of Variance (ANOVA) for assessing statistical significance between training approaches. The KW test results are generated using custom functions on Python 3.9, and the numpy [171] and pandas [172] libraries, with versions 1.22.0 and 1.3.5 respectively.

The Precision, Recall and F1 score metrics, popularised in [173], will be used to evaluate the performance of the anomaly detector model with each training method. Equation 4.8, Equation 4.9, and Equation 4.10 show how the Precision, Recall, and F-beta scores are calculated, respectively:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{4.8}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4.9}$$

$$F_\beta\ Score = (1 + \beta^2) \times \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{4.10}$$

Table 4.7: Optimised Hyperparameters for Neural Network

Hyperparameter	Value
Optimiser	Adam
Learning Rate	0.001
Epochs	8
Time Steps	1
Training samples	4,000,000
Batch Size	256
Input Branch Layer Width	32
Output Layer Width	96

where 'True Positives' (TP) denote the anomalies correctly identified by the NN model, 'False Positives' (FP) indicate the normal events incorrectly classified as anomalies, 'False Negatives' (FN) are anomalies that the NN model failed to detect, and β is the degree of prioritisation of recall over precision. In this study, we set β to 1, which means that the F-beta score becomes the F1 score, treating precision and recall equally in its calculation.

Neural Network Model

The multi-channel LSTMCaps autoencoder NN developed in previous work will be used as the anomaly detection models for the following experiments. Further details regarding the architecture are given in [2]. For the following experimentation, three input branches were used to accommodate the three features present in the dataset: the three-phase motor current signals. The model hyperparameters were optimised for anomaly detection using a grid-search procedure. Table 4.7 details the optimal hyperparameters for this task, which will be used across all training methods. For all experiments, this model will be trained on "healthy" motor data only, since it is an unsupervised autoencoder.

As Table 4.7 shows, the NN trains best with 4M samples. Since the healthy data from each dataset contains over 4M samples, each file from the training set was randomly sliced to reduce the number of samples from each file, totalling the 4M samples required for each dataset when concatenated. This approach ensures that the model will train on a wide variety of data, and subsequently increase the reliability of the experimental results by avoiding optimising for a specific set of files.

There are various methods of determining the error thresholds of an AE NN. The most common method is through the use of an unseen validation set. After training the NN, the model is run on a file in the same class as the training set, but which has not been used for training. In this case, a single file containing data for a healthy condition motor is used. The Mean Absolute Error (MAE) for each prediction is then calculated, which provides a baseline for the accuracy of the NN with reconstructing healthy data. When multiple features are present in the dataset, as is the case with the data used in this case study, a threshold is calculated for each feature since the reconstruction performance of the NN model may vary across each feature. The overall threshold can be calculated through different methods, and the method used is determined based on the training performance of the NN as well as the data consistency and behaviour. In the present work, two methods are used:

The largest MAE for each feature, or two standard deviations away from the mean MAE of each feature. The reasoning behind using two standard deviations from the mean as a threshold is, assuming a Gaussian distribution of error residual values, two standard deviations from the mean covers 95% of the data. In the case of an inconsistent or noisy dataset, it is better to use this method since there are more likely to be anomalous MAE values in the validation set, and if the largest MAE value is used, the threshold may be too high to consistently detect abnormalities in the test data. In the case of a consistent dataset with contains minimal noise, the largest MAE value is generally a good threshold to use since an MAE value to exceeds this threshold is more likely to indicate an anomaly as opposed to noisy but healthy behaviour.

Addressing limitations though experimental design

A potential limitation of the proposed experimentation is the experimentation on static datasets. By experimenting on static datasets, results achieved in experimental conditions may not generalise effectively to real-world conditions or even other data in the same problem domain. One experiment will address this limitation by recreating the dataset for each of the ten test iterations, shuffling the dataset files for each test iteration randomly in each dataset prior to data formulation.

Another potential limitation is the availability of an equal amount of health data from each Dataset tested. For the purpose of this experimentation, an equal amount of data from each dataset will be used, however, it cannot be ensured that there is an equal amount of healthy data available outside of experimental settings. Therefore, one experiment will address this issue by modifying the ratio of data used from each dataset to obverse the difference that is made to the anomaly detection results.

4.3.4 Experiment 1 - Training methods comparison

Experiment 1 aims to provide an initial comparison of the various training approaches detailed in Section 4.3.3. To ensure experimental validity, the experiment will be repeated 10 times. However, each iteration will not utilise the same training data; instead, it will employ a randomly shuffled subset of data from each file to create a dataset comprising 4M samples. This approach further bolsters the validity of the proposed method and mitigates the risk of misrepresenting its performance by only validating it on a single subset of data.

Table 4.8 presents the Precision, Recall, and F1 score results for each experiment, as well as the average F1 score across the datasets. Table 4.9 displays the results of the KW test for the Average F1 score results. Additionally, Figure 4.4 features a box and whisker plot that visually compares the Average F1 scores from all 10 runs from each training approach, offering a representation of result spread and consistency.

Table 4.8: Experiment 1 results - A comparison of the performance of the anomaly detection model using all training approaches. The experiment key is shown in Table 4.5

		Dataset A			Dataset B			
Training Approach	Score	Precision	Recall	F1	Precision	Recall	F1	Average F1
T- Dataset A	Best (out of 10)	0.841	1.000	0.914	0.429	0.750	0.545	0.730
	Average	0.815	0.943	0.867	0.356	0.550	0.423	0.645
	Std Dvn	0.018	0.170	0.105	0.155	0.218	0.157	0.100
T- Dataset B	Best (out of 10)	0.821	0.711	0.762	1.000	0.500	0.667	0.714
	Average	0.806	0.600	0.687	0.477	0.625	0.515	0.601
	Std Dvn	0.025	0.061	0.048	0.212	0.256	0.164	0.088
TL- Dataset A to B	Best (out of 10)	0.818	1.000	0.900	0.429	0.750	0.545	0.723
	Average	0.819	0.823	0.813	0.379	0.625	0.448	0.630
	Std Dvn	0.016	0.168	0.089	0.056	0.256	0.096	0.046
TL- Dataset B to A	Best (out of 10)	0.813	0.967	0.883	0.500	0.750	0.600	0.742
	Average	0.800	0.671	0.695	0.407	0.625	0.440	0.568
	Std Dvn	0.022	0.298	0.196	0.242	0.340	0.185	0.170
MD	Best (out of 10)	0.818	1.000	0.900	0.429	0.750	0.545	0.723
	Average	0.818	0.999	0.899	0.350	0.675	0.454	0.677
	Std Dvn	0.001	0.003	0.002	0.086	0.195	0.108	0.054
DF (Proposed)	**Best (out of 10)**	**0.818**	**1.000**	**0.900**	**1.000**	**0.750**	**0.857**	**0.879**
	Average	**0.818**	**1.000**	**0.900**	**1.000**	**0.600**	**0.743**	**0.821**
	Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.122**	**0.093**	**0.047**
TL – DF to Dataset A	Best (out of 10)	0.818	1.000	0.900	0.400	1.000	0.571	0.736
	Average	0.809	0.890	0.834	0.357	0.900	0.507	0.670
	Std Dvn	0.020	0.221	0.134	0.024	0.122	0.021	0.070
TL – DF to Dataset B	Best (out of 10)	0.818	1.000	0.900	0.500	1.000	0.667	0.783
	Average	0.819	0.977	0.890	0.566	0.750	0.562	0.726
	Std Dvn	0.004	0.070	0.031	0.240	0.250	0.084	0.038

Table 4.9: Kruskal-Wallis test results for Experiment 1 - Training methods comparison.

Experiment	n	Mean	SD	Median	Chi-square	*p*-Value
experiment 1.2	10	0.82	0.05	0.78		
experiment 1.3 fused to brb	10	0.73	0.04	0.73		
experiment 1.1	10	0.68	0.06	0.69	38.97	0.000*
experiment 1.3 fused to sc	10	0.67	0.07	0.70		
baseline 1 sc	10	0.64	0.11	0.68		
baseline 2 sc to brb	10	0.63	0.05	0.62		
baseline 1 brb	10	0.60	0.09	0.62		
baseline 2 brb to sc	10	0.57	0.18	0.59		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis reveals significant differences between multiple pairs, including: baseline 1 sc-experiment 1.2, baseline 1 brb-experiment 1.2, baseline 1 brb-experiment 1.3 fused to brb, baseline 2 sc to brb-experiment 1.2, baseline 2 sc to brb-experiment 1.3 fused to brb, baseline 2 brb to sc-experiment 1.2, experiment 1.1-experiment 1.2, experiment 1.2-experiment 1.3 fused to sc, and experiment 1.2-experiment 1.3 fused to brb.

4.3.5 Experiment 2 - Varying data volume

Experiment 2 aims to observe the effect of reducing the volume of training data on the performance of the anomaly detection model using the different training

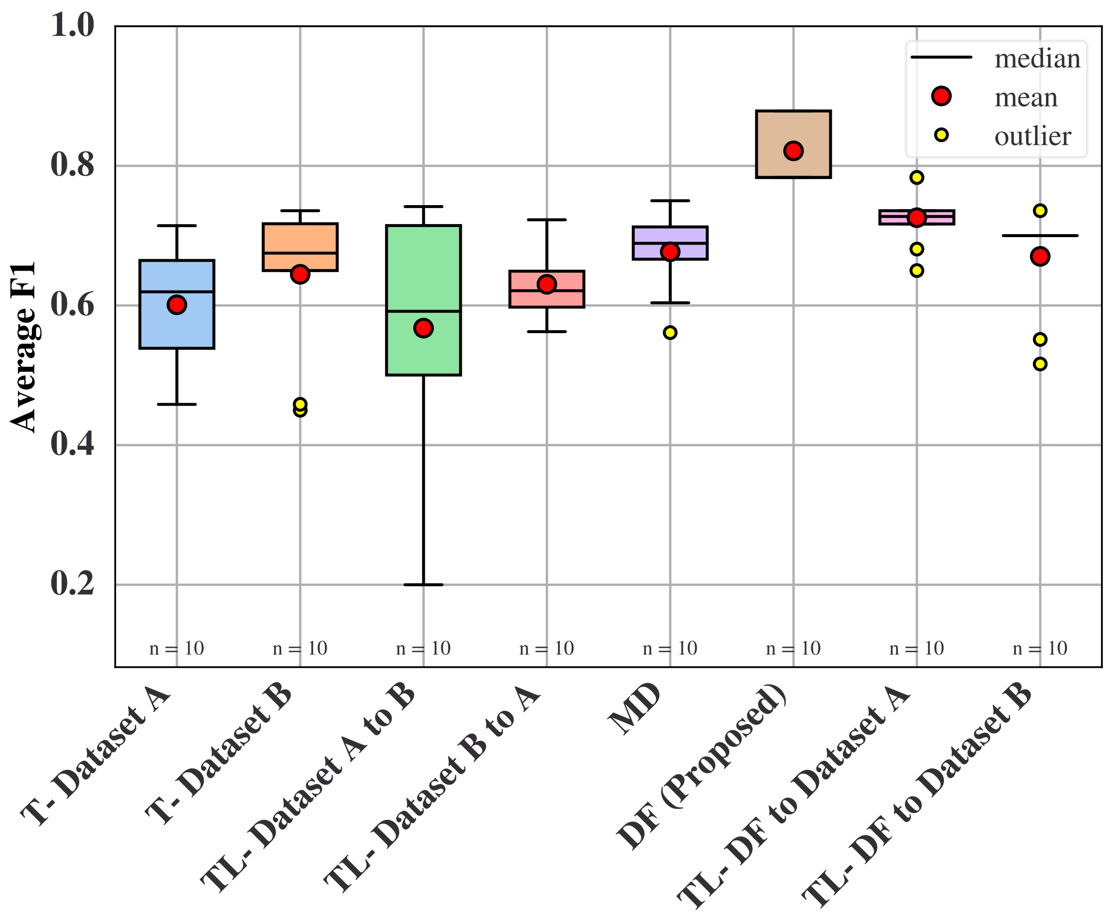


Figure 4.4: Box and whisker plot comparing the results for Experiment 1 - Training methods comparison

approaches. Similar to Experiment 1, the experiment will be repeated 10 times, and the dataset will be shuffled to ensure experimental validity. The experiment details for each test are shown in Table 4.10. The estimated FLOPs used for each number of samples are calculated using Equation 4.11 [174]:

$$FLOPs = 2 \cdot P \cdot 3 \cdot N \cdot E \tag{4.11}$$

where P is the number of trainable parameters in the NN, N is number of training samples, and E is the number of epochs. Whilst the complexity of the model can undoubtedly affect the complexity of training, since the same model is used across all experiments, it will not be relevant for this calculation.

The tabulated results for experiment 2 can be found in Table 4.11. A summary of the results in the form of a box and whisker plot is illustrated in Figure 4.5. The KW test results for the Average F1 scores are shown in Table 4.12.

4.3.6 Experiment 3 - Varying dataset ratio

Experiment 3 aims to assess the performance of each training method with an imbalanced dataset containing a different number of samples from each dataset. The purpose of this experiment is to simulate a real-world environment, where the vol-

Table 4.10: Experiment settings for Experiment 2 - Varying data volume, including an estimation of the FLOPs used for training

% Training data used	Number of Samples	Number of Epochs	NN Trainable Parameters	FLOPs used for training
100%	4,000,000	8	25,635	4.92×10^{12}
50%	2,000,000	8	25,635	2.46×10^{12}
25%	1,000,000	8	25,635	1.23×10^{11}
12.5%	500,000	8	25,635	6.15×10^{11}
6.25%	250,000	8	25,635	3.08×10^{11}

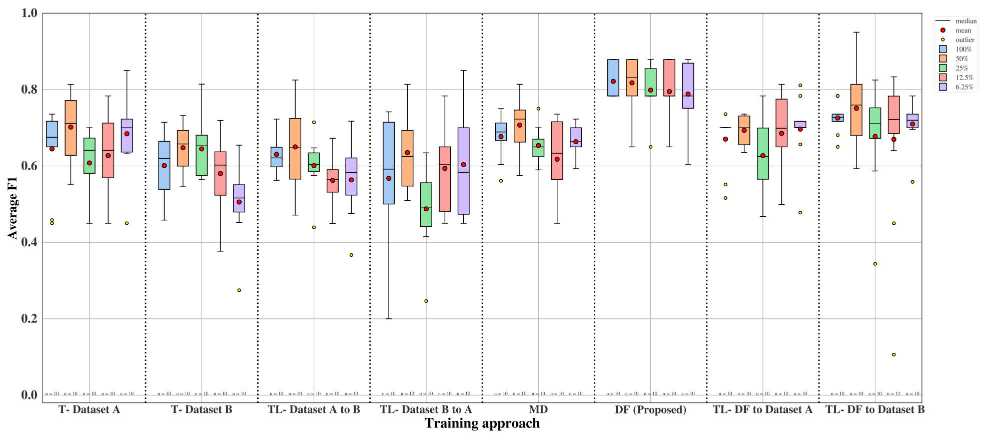


Figure 4.5: Box and whisker plot showcasing the impact of varying data volumes on each training method, grouped by training approach and plotted against average F1 score (higher is better).

ume of data available from different sources will not be equal in many cases. Similar to Experiments 1 and 2, the experiment will be repeated 10 times, and the dataset will be shuffled to ensure experimental validity.

Table 4.13 shows a breakdown of the experimental settings used. In the case of traditional training, the anomaly detector model will be trained on the reduced dataset, similar to experiment 2. However, transfer learning approaches will make use of both datasets.

The tabulated results for experiment 3 can be found in Table 4.14 for results from 10:90 to 50:50 (Dataset A: Dataset B), and in Table 4.15 for results from 60:40 to 90:10. The tabulated results are summarised in a box and whisker plot, shown in Figure 4.6. The KW test results for the Average F1 scores are shown in Table 4.16.

4.4 Discussion

4.4.1 Experiment 1 - Training methods comparison Analysis

Examining the experimental results of Experiment 1, as presented in Table 4.8, the DF method is empirically proven to consistently deliver the best performance in

Table 4.11: Full results of experiment 2 - Varying data volume

Percentage of data used	**Experiment**	**Score**	**Dataset A**			**Dataset B**			**Average**	**% change of Average**
			Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**F1**	**F1 from Baseline**
100% (**Baseline**)	T- Dataset A	Average	0.815	0.943	0.867	0.356	0.550	0.423	0.645	N/A
		Std Dev	0.018	0.170	0.105	0.155	0.218	0.157	0.100	
	T- Dataset B	Average	0.806	0.600	0.687	0.477	0.625	0.515	0.601	N/A
		Std Dev	0.025	0.061	0.048	0.212	0.256	0.164	0.088	
	TL- Dataset A to B	Average	0.819	0.823	0.813	0.379	0.625	0.448	0.630	N/A
		Std Dev	0.016	0.168	0.089	0.056	0.256	0.096	0.046	
	TL- Dataset B to A	Average	0.800	0.671	0.695	0.407	0.625	0.440	0.568	N/A
		Std Dev	0.022	0.298	0.196	0.242	0.340	0.185	0.170	
	MD	Average	0.818	0.999	0.899	0.350	0.675	0.454	0.677	N/A
		Std Dev	0.001	0.003	0.002	0.086	0.195	0.108	0.054	
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.600**	**0.743**	**0.821**	N/A
		Std Dev	**0.000**	**0.000**	**0.000**	**0.000**	**0.122**	**0.093**	**0.047**	
	TL – DF to Dataset A	Average	0.809	0.890	0.834	0.357	0.900	0.507	0.670	N/A
		Std Dev	0.020	0.221	0.134	0.024	0.122	0.021	0.070	
	TL – DF to Dataset B	Average	0.819	0.977	0.890	0.566	0.750	0.562	0.726	N/A
		Std Dev	0.004	0.070	0.031	0.240	0.250	0.084	0.038	
50.00%	T- Dataset A	Average	0.810	0.937	0.859	0.449	0.750	0.545	0.702	8.86%
		Std Dev	0.025	0.190	0.123	0.125	0.250	0.146	0.088	
	T- Dataset B	Average	0.819	0.638	0.715	0.481	0.800	0.580	0.648	7.77%
		Std Dev	0.031	0.088	0.061	0.121	0.218	0.113	0.056	
	TL- Dataset A to B	Average	0.811	0.622	0.687	0.508	0.800	0.613	0.650	3.11%
		Std Dev	0.036	0.210	0.142	0.132	0.187	0.130	0.105	
	TL- Dataset B to A	Average	0.796	0.729	0.737	0.447	0.750	0.533	0.635	11.89%
		Std Dev	0.042	0.259	0.162	0.123	0.250	0.113	0.103	
	MD	Average	0.818	1.000	0.900	0.404	0.725	0.514	0.707	4.45%
		Std Dev	0.000	0.000	0.000	0.100	0.236	0.140	0.070	
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.600**	**0.735**	**0.818**	-0.46%
		Std Dev	**0.000**	**0.000**	**0.000**	**0.000**	**0.166**	**0.143**	**0.072**	
	TL – DF to Dataset A	Average	0.816	0.989	0.894	0.380	0.800	0.492	0.693	3.40%
		Std Dev	0.005	0.033	0.017	0.104	0.218	0.066	0.037	
	TL – DF to Dataset B	Average	0.816	0.951	0.872	0.522	0.850	0.629	0.751	3.46%
		Std Dev	0.008	0.147	0.084	0.189	0.255	0.190	0.108	
25.00%	T- Dataset A	Average	0.821	0.966	0.884	0.258	0.475	0.332	0.608	-5.70%
		Std Dev	0.007	0.103	0.049	0.133	0.261	0.172	0.086	
	T- Dataset B	Average	0.812	0.623	0.704	0.517	0.775	0.585	0.645	7.25%
		Std Dev	0.025	0.053	0.038	0.211	0.208	0.136	0.074	
	TL- Dataset A to B	Average	0.801	0.609	0.682	0.470	0.650	0.520	0.601	-4.67%
		Std Dev	0.040	0.165	0.119	0.196	0.122	0.097	0.066	
	TL- Dataset B to A	Average	0.783	0.620	0.664	0.277	0.400	0.311	0.488	-14.08%
		Std Dev	0.039	0.272	0.183	0.145	0.229	0.150	0.105	
	MD	Average	0.818	0.957	0.878	0.349	0.600	0.429	0.653	-3.46%
		Std Dev	0.006	0.123	0.066	0.063	0.200	0.085	0.046	
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.550**	**0.697**	**0.799**	**-2.78%**
		Std Dev	**0.000**	**0.000**	**0.000**	**0.000**	**0.150**	**0.131**	**0.065**	
	TL – DF to Dataset A	Average	0.796	0.768	0.758	0.417	0.750	0.496	0.627	-6.45%
		Std Dev	0.031	0.273	0.170	0.208	0.250	0.119	0.093	
	TL – DF to Dataset B	Average	0.814	0.906	0.850	0.420	0.725	0.504	0.677	-6.68%
		Std Dev	0.019	0.167	0.094	0.189	0.325	0.188	0.126	
12.50%	T- Dataset A	Average	0.818	1.000	0.900	0.341	0.425	0.355	0.627	-2.70%
		Std Dev	0.000	0.000	0.000	0.269	0.275	0.219	0.110	
	T- Dataset B	Average	0.792	0.510	0.614	0.519	0.625	0.546	0.580	-3.47%
		Std Dev	0.031	0.121	0.098	0.228	0.230	0.192	0.094	
	TL- Dataset A to B	Average	0.781	0.466	0.574	0.539	0.650	0.550	0.562	-10.78%
		Std Dev	0.039	0.136	0.119	0.196	0.200	0.132	0.062	
	TL- Dataset B to A	Average	0.818	1.000	0.900	0.355	0.300	0.288	0.594	4.62%
		Std Dev	0.000	0.000	0.000	0.353	0.245	0.222	0.111	
	MD	Average	0.818	0.998	0.899	0.279	0.450	0.337	0.618	-8.72%
		Std Dev	0.001	0.007	0.003	0.166	0.312	0.205	0.102	
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.550**	**0.690**	**0.795**	**-3.25%**
		Std Dev	**0.000**	**0.000**	**0.000**	**0.000**	**0.187**	**0.168**	**0.084**	
	TL – DF to Dataset A	Average	0.816	0.882	0.829	0.464	0.775	0.542	0.685	2.23%
		Std Dev	0.016	0.236	0.142	0.208	0.175	0.111	0.102	
	TL – DF to Dataset B	Average	0.822	0.953	0.880	0.553	0.700	0.561	0.721	-0.68%
		Std Dev	0.014	0.095	0.048	0.301	0.292	0.206	0.106	
6.25%	T- Dataset A	Average	0.818	1.000	0.900	0.418	0.600	0.468	0.684	6.14%
		Std Dev	0.000	0.000	0.000	0.249	0.255	0.202	0.101	
	T- Dataset B	Average	0.777	0.341	0.472	0.600	0.550	0.539	0.506	-15.87%
		Std Dev	0.039	0.049	0.047	0.268	0.269	0.217	0.094	
	TL- Dataset A to B	Average	0.757	0.382	0.504	0.565	0.750	0.624	0.564	-10.59%
		Std Dev	0.035	0.080	0.073	0.144	0.274	0.184	0.092	
	TL- Dataset B to A	Average	0.814	0.892	0.838	0.353	0.425	0.369	0.604	6.35%
		Std Dev	0.014	0.207	0.121	0.228	0.297	0.236	0.132	
	MD	Average	0.819	0.994	0.898	0.380	0.550	0.429	0.664	-1.97%
		Std Dev	0.002	0.017	0.006	0.070	0.218	0.085	0.042	
	DF (Proposed)	**Average**	**0.813**	**0.849**	**0.815**	**1.000**	**0.625**	**0.762**	**0.788**	**-4.04%**
		Std Dev	**0.019**	**0.223**	**0.134**	**0.000**	**0.125**	**0.095**	**0.083**	
	TL – DF to Dataset A	Average	0.812	0.876	0.830	0.451	0.850	0.563	0.696	3.88%
		Std Dev	0.011	0.200	0.120	0.130	0.200	0.092	0.084	
	TL – DF to Dataset B	Average	0.823	0.913	0.852	0.427	0.900	0.568	0.710	-2.19%
		Std Dev	0.012	0.189	0.109	0.056	0.166	0.049	0.056	

Table 4.12: Kruskal-Wallis test results for Experiment 2 - Varying data volume.

Group	**n**	**Mean**	**SD**	**Median**	**Chi-square**	***p*-Value**
DF (100%)	10	0.82	0.05	0.78		
DF (50%)	10	0.82	0.08	0.83		
DF (25%)	10	0.80	0.07	0.78		
DF (12.5%)	10	0.79	0.09	0.78		
DF (6.25%)	10	0.79	0.09	0.78		
TL - DF to Dataset B (50%)	10	0.75	0.11	0.76		
TL - DF to Dataset B (100%)	10	0.73	0.04	0.73		
TL - DF to Dataset B (6.25%)	10	0.71	0.06	0.72		
MD (50%)	10	0.71	0.07	0.72		
T - Dataset A (50%)	10	0.70	0.09	0.71		
TL - DF to Dataset A (6.25%)	10	0.70	0.09	0.70	176.69	0.000*
TL - DF to Dataset A (50%)	10	0.69	0.04	0.70		
TL - DF to Dataset A (12.5%)	10	0.69	0.11	0.70		
T - Dataset A (6.25%)	10	0.68	0.11	0.70		
TL - DF to Dataset B (25%)	10	0.68	0.13	0.71		
MD (100%)	10	0.68	0.06	0.69		
TL - DF to Dataset A (100%)	10	0.67	0.07	0.70		
TL - DF to Dataset B (12.5%)	12	0.67	0.20	0.72		
MD (6.25%)	10	0.66	0.04	0.66		
MD (25%)	10	0.65	0.05	0.65		
TL - Dataset B to A (50%)	10	0.65	0.11	0.65		
T - Dataset B (50%)	10	0.65	0.06	0.66		
T - Dataset A (100%)	10	0.64	0.11	0.68		
T - Dataset B (25%)	10	0.64	0.08	0.65		
TL - Dataset A to B (50%)	10	0.64	0.11	0.62		
TL - Dataset B to A (100%)	10	0.63	0.05	0.62		
T - Dataset A (12.5%)	10	0.63	0.12	0.64		
TL - DF to Dataset A (25%)	10	0.63	0.10	0.62		
MD (12.5%)	10	0.62	0.11	0.63		
T - Dataset A (25%)	10	0.61	0.09	0.64		
TL - Dataset A to B (6.25%)	10	0.60	0.14	0.58		
T - Dataset B (100%)	10	0.60	0.09	0.62		
TL - Dataset B to A (25%)	10	0.60	0.07	0.60		
TL - Dataset A to B (12.5%)	10	0.59	0.12	0.60		
T - Dataset B (12.5%)	10	0.58	0.10	0.60		
TL - Dataset A to B (100%)	10	0.57	0.18	0.59		
TL - Dataset B to A (6.25%)	10	0.56	0.10	0.58		
TL - Dataset B to A (12.5%)	10	0.56	0.07	0.56		
T - Dataset B (6.25%)	10	0.51	0.10	0.52		
TL - Dataset A to B (25%)	10	0.49	0.11	0.49		

Note: * indicates significant difference (*p* ¡ 0.05). Post-hoc analysis revealed numerous significant pairwise differences between groups. Due to the extensive number of significant pairs, they are not all listed here.

Table 4.13: Experiment 3 - Varying dataset ratio details

Dataset Ratio (A:B)	Dataset A samples	Dataset B samples
10:90	400,000	3,600,000
20:80	800,000	3,200,000
30:70	1,200,000	2,800,000
60:40	1,600,000	2,400,000
50:50	2,000,000	2,000,000
40:60	2,400,000	1,600,000
30:70	2,800,000	1,200,000
20:80	3,200,000	800,000
90:10	3,600,000	400,000

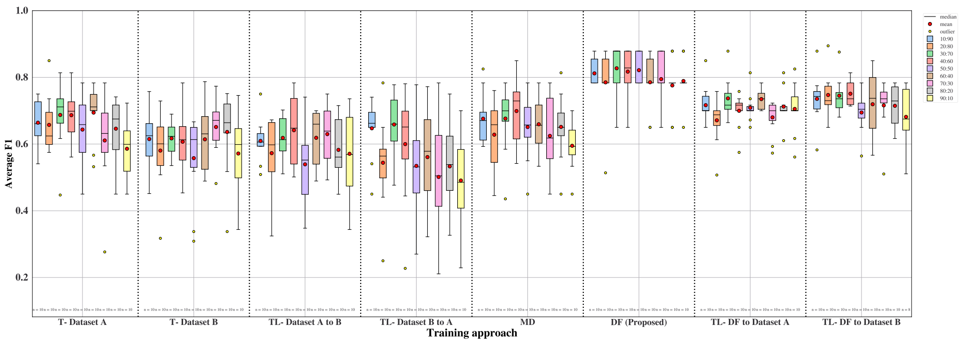


Figure 4.6: Box and whisker plot showcasing the impact of varying dataset ratios on each training method, grouped by training approach and plotted against average F1 score (higher is better).

terms of the F1 score for both Datasets A and B, as well as the average F1 score between both datasets. With the best Average F1 score of 0.879 and a 10-run average of 0.821, DF surpasses the other methods. These findings suggest that the DF approach effectively captures the salient features in both datasets, resulting in consistently strong performance across the datasets compared to the compared methods. Moreover, the results imply a considerable advantage in fusing the datasets, as the performance on individual datasets significantly exceeds that of models specialised in each respective datasets.

In comparison, the traditional training approach on Dataset A (T-Dataset A) exhibits better performance on Dataset A with a mean F1 score of 0.867 but suffers from poor results on Dataset B with a weaker score of 0.423, consequently lowering the average F1 score and making this model unsuitable for use across homogeneous datasets. The superior performance of the DF method on both Dataset A and Dataset B, along with the average F1 score across both datasets, underscores the algorithm's effectiveness in fulfilling the need for a single neural network adaptable to data from various sources within the same problem domain. This outcome aligns with the algorithm's proposed benefits, which aim to eliminate the need for multiple NNs and reduce data requirements from individual sources.

Furthermore, the results indicate that the traditional training approach, while effective for the dataset it was trained on, falls short in terms of generalisability

Table 4.14: Results for Experiment 3 - Dataset Ratio from 90:10 to 50:50 (Dataset A : Dataset B)

Data Ratio			Dataset A			Dataset B			Average
(A:B)	Experiment	Score	Precision	Recall	F1	Precision	Recall	F1	F1
10:90	T - Dataset A	Average	0.818	1.000	0.900	0.375	0.525	0.428	0.664
		Std Dvn	0.000	0.000	0.000	0.143	0.175	0.137	0.069
	T - Dataset B	Average	0.790	0.587	0.667	0.492	0.700	0.563	0.615
		Std Dvn	0.034	0.134	0.093	0.144	0.269	0.173	0.093
	TL- Dataset A to B	Average	0.799	0.663	0.710	0.394	0.800	0.509	0.609
		Std Dvn	0.038	0.206	0.137	0.059	0.292	0.121	0.062
	TL- Dataset B to A	Average	0.822	0.958	0.882	0.338	0.600	0.413	0.647
		Std Dvn	0.022	0.086	0.036	0.170	0.320	0.182	0.084
	MD	Average	0.820	0.828	0.814	0.495	0.775	0.538	0.676
		Std Dvn	0.028	0.180	0.095	0.206	0.284	0.126	0.074
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.575**	**0.724**	**0.812**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.115**	**0.087**	**0.044**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.456	0.875	0.533	0.717
		Std Dvn	0.000	0.000	0.000	0.208	0.256	0.112	0.056
	TL - DF to Dataset B	Average	0.815	0.911	0.852	0.582	0.800	0.619	0.735
		Std Dvn	0.015	0.168	0.092	0.227	0.218	0.104	0.070
20:80	T - Dataset A	Average	0.818	1.000	0.900	0.366	0.525	0.415	0.657
		Std Dvn	0.000	0.000	0.000	0.156	0.236	0.164	0.082
	T - Dataset B	Average	0.786	0.548	0.641	0.417	0.750	0.520	0.580
		Std Dvn	0.061	0.123	0.112	0.149	0.250	0.157	0.108
	TL- Dataset A to B	Average	0.809	0.632	0.700	0.388	0.575	0.445	0.573
		Std Dvn	0.029	0.155	0.094	0.163	0.297	0.177	0.107
	TL- Dataset B to A	Average	0.800	0.694	0.720	0.325	0.525	0.368	0.544
		Std Dvn	0.018	0.248	0.155	0.131	0.361	0.173	0.131
	MD	Average	0.824	0.812	0.810	0.429	0.500	0.447	0.628
		Std Dvn	0.040	0.176	0.102	0.126	0.158	0.113	0.099
	DF (Proposed)	**Average**	**0.813**	**0.953**	**0.873**	**1.000**	**0.550**	**0.697**	**0.785**
		Std Dvn	**0.017**	**0.140**	**0.082**	**0.000**	**0.150**	**0.131**	**0.100**
	TL - DF to Dataset A	Average	0.812	0.902	0.846	0.385	0.800	0.496	0.671
		Std Dvn	0.032	0.184	0.115	0.102	0.218	0.060	0.067
	TL - DF to Dataset B	Average	0.817	0.990	0.895	0.463	0.900	0.599	0.747
		Std Dvn	0.003	0.020	0.010	0.134	0.166	0.121	0.061
30:70	T - Dataset A	Average	0.818	0.999	0.899	0.415	0.625	0.475	0.687
		Std Dvn	0.001	0.003	0.002	0.169	0.301	0.189	0.095
	T - Dataset B	Average	0.816	0.633	0.713	0.476	0.675	0.521	0.617
		Std Dvn	0.016	0.034	0.023	0.197	0.195	0.085	0.051
	TL- Dataset A to B	Average	0.806	0.591	0.681	0.518	0.750	0.556	0.619
		Std Dvn	0.023	0.049	0.038	0.203	0.250	0.120	0.061
	TL- Dataset B to A	Average	0.812	0.822	0.803	0.498	0.650	0.514	0.659
		Std Dvn	0.039	0.212	0.128	0.197	0.278	0.133	0.085
	MD	Average	0.817	0.904	0.848	0.429	0.700	0.505	0.676
		Std Dvn	0.006	0.183	0.100	0.162	0.269	0.155	0.096
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.625**	**0.754**	**0.827**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.168**	**0.146**	**0.073**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.507	0.850	0.574	0.737
		Std Dvn	0.000	0.000	0.000	0.252	0.200	0.113	0.056
	TL - DF to Dataset B	Average	0.823	0.967	0.886	0.498	0.875	0.604	0.745
		Std Dvn	0.016	0.096	0.041	0.188	0.168	0.113	0.052
40:60	T - Dataset A	Average	0.817	0.990	0.895	0.499	0.550	0.478	0.687
		Std Dvn	0.005	0.030	0.016	0.213	0.269	0.153	0.076
	T - Dataset B	Average	0.803	0.631	0.700	0.432	0.675	0.514	0.607
		Std Dvn	0.028	0.144	0.090	0.136	0.275	0.164	0.099
	TL- Dataset A to B	Average	0.795	0.678	0.715	0.514	0.725	0.568	0.641
		Std Dvn	0.043	0.219	0.147	0.119	0.284	0.139	0.101
	TL- Dataset B to A	Average	0.793	0.670	0.707	0.394	0.725	0.492	0.600
		Std Dvn	0.046	0.244	0.164	0.168	0.344	0.206	0.154
	MD	Average	0.815	0.979	0.889	0.415	0.725	0.509	0.699
		Std Dvn	0.010	0.056	0.030	0.131	0.284	0.170	0.092
	DF (Proposed)	**Average**	**0.818**	**0.998**	**0.899**	**1.000**	**0.600**	**0.735**	**0.817**
		Std Dvn	**0.001**	**0.007**	**0.003**	**0.000**	**0.166**	**0.143**	**0.071**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.435	0.800	0.499	0.700
		Std Dvn	0.000	0.000	0.000	0.196	0.292	0.099	0.049
	TL - DF to Dataset B	Average	0.820	0.998	0.900	0.476	0.900	0.602	0.751
		Std Dvn	0.007	0.004	0.003	0.104	0.166	0.069	0.035
50:50 **(Baseline)**	T - Dataset A	Average	0.818	1.000	0.900	0.335	0.475	0.387	0.643
		Std Dvn	0.000	0.000	0.000	0.169	0.261	0.198	0.099
	T - Dataset B	Average	0.802	0.544	0.648	0.362	0.700	0.467	0.558
		Std Dvn	0.023	0.038	0.032	0.208	0.367	0.249	0.125
	TL- Dataset A to B	Average	0.791	0.589	0.658	0.356	0.550	0.420	0.539
		Std Dvn	0.038	0.213	0.140	0.153	0.245	0.165	0.123
	TL- Dataset B to A	Average	0.798	0.748	0.746	0.278	0.425	0.323	0.535
		Std Dvn	0.027	0.283	0.178	0.172	0.317	0.206	0.141
	MD	Average	0.817	0.994	0.897	0.326	0.550	0.405	0.651
		Std Dvn	0.003	0.017	0.008	0.162	0.269	0.193	0.095
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.600**	**0.743**	**0.821**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.122**	**0.093**	**0.047**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.475	0.775	0.518	0.709
		Std Dvn	0.000	0.000	0.000	0.205	0.261	0.082	0.041
	TL - DF to Dataset B	Average	0.818	0.929	0.866	0.444	0.675	0.523	0.695
		Std Dvn	0.006	0.122	0.059	0.103	0.160	0.084	0.055

Table 4.15: Results for Experiment 3 - Dataset Ratio from 40:60 to 10:90 (Dataset A : Dataset B)

Data Ratio			Dataset A			Dataset B			Average
(A:B)	Experiment	Score	Precision	Recall	F1	Precision	Recall	F1	F1
60:40	T - Dataset A	Average	0.820	0.969	0.886	0.418	0.675	0.502	0.694
		Std Dvn	0.008	0.083	0.036	0.090	0.251	0.129	0.078
	T - Dataset B	Average	0.810	0.582	0.677	0.547	0.650	0.552	0.614
		Std Dvn	0.019	0.052	0.040	0.242	0.278	0.183	0.095
	TL- Dataset A to B	Average	0.815	0.721	0.757	0.453	0.625	0.480	0.619
		Std Dvn	0.017	0.150	0.080	0.227	0.280	0.147	0.084
	TL- Dataset B to A	Average	0.801	0.633	0.675	0.410	0.575	0.447	0.561
		Std Dvn	0.035	0.285	0.178	0.247	0.317	0.217	0.140
	MD	Average	0.828	0.903	0.853	0.359	0.675	0.465	0.659
		Std Dvn	0.018	0.168	0.084	0.120	0.195	0.138	0.076
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.525**	**0.670**	**0.785**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.175**	**0.158**	**0.079**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.477	0.900	0.570	0.735
		Std Dvn	0.000	0.000	0.000	0.199	0.200	0.061	0.030
	TL - DF to Dataset B	Average	0.818	0.904	0.851	0.543	0.800	0.588	0.719
		Std Dvn	0.008	0.159	0.082	0.202	0.269	0.137	0.095
70:30	T - Dataset A	Average	0.805	0.843	0.807	0.350	0.525	0.415	0.611
		Std Dvn	0.029	0.241	0.153	0.152	0.261	0.184	0.130
	T - Dataset B	Average	0.809	0.636	0.709	0.487	0.800	0.594	0.651
		Std Dvn	0.026	0.108	0.069	0.162	0.218	0.163	0.077
	TL- Dataset A to B	Average	0.800	0.671	0.719	0.434	0.750	0.541	0.630
		Std Dvn	0.020	0.182	0.110	0.083	0.224	0.122	0.088
	TL- Dataset B to A	Average	0.778	0.557	0.618	0.325	0.500	0.385	0.501
		Std Dvn	0.038	0.282	0.185	0.174	0.316	0.216	0.177
	MD	Average	0.823	0.991	0.899	0.308	0.425	0.350	0.624
		Std Dvn	0.013	0.027	0.004	0.190	0.317	0.231	0.116
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.550**	**0.690**	**0.795**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.187**	**0.168**	**0.084**
	TL - DF to Dataset A	Average	0.813	0.962	0.881	0.355	0.800	0.479	0.680
		Std Dvn	0.010	0.069	0.036	0.038	0.245	0.084	0.052
	TL - DF to Dataset B	Average	0.815	0.954	0.877	0.423	0.850	0.555	0.716
		Std Dvn	0.006	0.100	0.051	0.055	0.229	0.104	0.073
80:20	T - Dataset A	Average	0.824	0.974	0.892	0.338	0.525	0.401	0.646
		Std Dvn	0.012	0.063	0.025	0.175	0.284	0.193	0.092
	T - Dataset B	Average	0.801	0.610	0.688	0.561	0.725	0.585	0.636
		Std Dvn	0.021	0.111	0.069	0.243	0.325	0.219	0.119
	TL- Dataset A to B	Average	0.793	0.611	0.681	0.454	0.575	0.485	0.583
		Std Dvn	0.022	0.161	0.098	0.141	0.251	0.152	0.093
	TL- Dataset B to A	Average	0.787	0.680	0.696	0.371	0.450	0.370	0.533
		Std Dvn	0.039	0.309	0.206	0.261	0.269	0.194	0.124
	MD	Average	0.818	1.000	0.900	0.342	0.500	0.403	0.651
		Std Dvn	0.000	0.000	0.000	0.159	0.250	0.188	0.094
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.500**	**0.651**	**0.776**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.158**	**0.146**	**0.073**
	TL - DF to Dataset A	Average	0.815	0.959	0.878	0.411	0.875	0.547	0.712
		Std Dvn	0.008	0.123	0.067	0.081	0.168	0.078	0.052
	TL - DF to Dataset B	Average	0.817	0.994	0.897	0.580	0.625	0.532	0.715
		Std Dvn	0.003	0.017	0.008	0.231	0.256	0.115	0.059
90:10	T - Dataset A	Average	0.818	0.942	0.871	0.251	0.400	0.301	0.586
		Std Dvn	0.016	0.138	0.079	0.137	0.255	0.173	0.090
	T - Dataset B	Average	0.790	0.513	0.616	0.511	0.600	0.527	0.572
		Std Dvn	0.023	0.103	0.090	0.252	0.300	0.228	0.115
	TL- Dataset A to B	Average	0.791	0.598	0.668	0.434	0.600	0.473	0.571
		Std Dvn	0.043	0.191	0.135	0.248	0.320	0.220	0.125
	TL- Dataset B to A	Average	0.782	0.687	0.703	0.257	0.325	0.278	0.491
		Std Dvn	0.040	0.291	0.193	0.154	0.195	0.161	0.133
	MD	Average	0.819	0.977	0.890	0.265	0.375	0.300	0.595
		Std Dvn	0.004	0.070	0.031	0.116	0.202	0.136	0.068
	DF (Proposed)	**Average**	**0.818**	**1.000**	**0.900**	**1.000**	**0.525**	**0.678**	**0.789**
		Std Dvn	**0.000**	**0.000**	**0.000**	**0.000**	**0.135**	**0.119**	**0.060**
	TL - DF to Dataset A	Average	0.818	1.000	0.900	0.444	0.750	0.511	0.705
		Std Dvn	0.000	0.000	0.000	0.235	0.250	0.143	0.072
	TL - DF to Dataset B	Average	0.811	0.940	0.862	0.618	0.650	0.529	0.696
		Std Dvn	0.020	0.180	0.113	0.259	0.300	0.117	0.085

Table 4.16: Kruskal-Wallis test results for Experiment 3 - Varying dataset ratio.

Group	**n**	**Mean**	**SD**	**Median**	**Chi-square**	***p*-Value**
DF (Proposed) 30:70	10	0.83	0.08	0.88		
DF (Proposed) 50:50	10	0.82	0.05	0.78		
DF (Proposed) 40:60	10	0.82	0.08	0.83		
DF (Proposed) 10:90	10	0.81	0.05	0.78		
DF (Proposed) 70:30	10	0.79	0.09	0.78		
DF (Proposed) 90:10	10	0.79	0.06	0.78		
DF (Proposed) 60:40	10	0.79	0.08	0.78		
DF (Proposed) 20:80	10	0.78	0.11	0.78		
DF (Proposed) 80:20	10	0.78	0.08	0.78		
TL - DF to Dataset B 40:60	10	0.75	0.04	0.74		
TL - DF to Dataset B 20:80	10	0.75	0.06	0.73		
TL - DF to Dataset B 30:70	10	0.74	0.05	0.74		
TL - DF to Dataset A 30:70	10	0.74	0.06	0.72	326.68	0.000*
TL - DF to Dataset B 10:90	10	0.74	0.07	0.74		
TL - DF to Dataset A 60:40	10	0.73	0.03	0.74		
TL - DF to Dataset B 60:40	10	0.72	0.10	0.74		
TL - DF to Dataset A 10:90	10	0.72	0.06	0.70		
TL - DF to Dataset B 70:30	10	0.72	0.08	0.74		
TL - DF to Dataset B 80:20	10	0.71	0.06	0.73		
TL - DF to Dataset A 80:20	10	0.71	0.05	0.70		
TL - DF to Dataset A 50:50	10	0.71	0.04	0.70		
TL - DF to Dataset A 90:10	10	0.71	0.08	0.70		
TL - DF to Dataset A 40:60	10	0.70	0.05	0.72		
MD 40:60	10	0.70	0.10	0.73		
TL - DF to Dataset B 50:50	10	0.69	0.06	0.70		
T - Dataset A 60:40	10	0.69	0.08	0.71		
T - Dataset A 30:70	10	0.69	0.10	0.71		
T - Dataset A 40:60	10	0.69	0.08	0.70		
TL - DF to Dataset B 90:10	8	0.68	0.09	0.68		
TL - DF to Dataset A 70:30	10	0.68	0.05	0.70		
MD 30:70	10	0.68	0.10	0.70		
MD 10:90	10	0.68	0.08	0.67		
TL - DF to Dataset A 20:80	10	0.67	0.07	0.69		
T - Dataset A 10:90	10	0.66	0.07	0.66		
MD 60:40	10	0.66	0.08	0.66		
TL - Dataset B to A 30:70	10	0.66	0.09	0.66		
T - Dataset A 20:80	10	0.66	0.09	0.62		
MD 80:20	10	0.65	0.10	0.65		
T - Dataset B 70:30	10	0.65	0.08	0.67		
MD 50:50	10	0.65	0.10	0.66		
TL - Dataset B to A 10:90	10	0.65	0.09	0.66		
T - Dataset A 80:20	10	0.65	0.10	0.68		
T - Dataset A 50:50	10	0.64	0.10	0.66		
TL - Dataset A to B 40:60	10	0.64	0.11	0.65		
T - Dataset B 80:20	10	0.64	0.13	0.66		
TL - Dataset A to B 70:30	10	0.63	0.09	0.64		
MD 20:80	10	0.63	0.10	0.66		
MD 70:30	10	0.62	0.12	0.62		
TL - Dataset A to B 60:40	10	0.62	0.09	0.66		
TL - Dataset A to B 30:70	10	0.62	0.06	0.61		
T - Dataset B 30:70	10	0.62	0.05	0.62		
T - Dataset B 10:90	10	0.61	0.10	0.63		
T - Dataset B 60:40	10	0.61	0.10	0.63		
T - Dataset A 70:30	10	0.61	0.14	0.63		
TL - Dataset A to B 10:90	10	0.61	0.07	0.61		
T - Dataset B 40:60	10	0.61	0.10	0.61		
TL - Dataset B to A 40:60	10	0.60	0.16	0.65		
MD 90:10	10	0.59	0.07	0.59		
T - Dataset A 90:10	10	0.59	0.10	0.60		
TL - Dataset A to B 80:20	10	0.58	0.10	0.56		
T - Dataset B 20:80	10	0.58	0.11	0.60		
TL - Dataset A to B 20:80	10	0.57	0.11	0.60		
T - Dataset B 90:10	10	0.57	0.12	0.60		
TL - Dataset A to B 90:10	10	0.57	0.13	0.57		
TL - Dataset B to A 60:40	10	0.56	0.15	0.58		
T - Dataset B 50:50	10	0.56	0.13	0.61		
TL - Dataset B to A 20:80	10	0.54	0.14	0.56		
TL - Dataset A to B 50:50	10	0.54	0.13	0.55		
TL - Dataset B to A 50:50	10	0.53	0.15	0.53		
TL - Dataset B to A 80:20	10	0.53	0.13	0.54		
TL - Dataset B to A 70:30	10	0.50	0.19	0.50		
TL - Dataset B to A 90:10	10	0.49	0.14	0.49		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis revealed numerous significant pairwise differences between groups. Due to the extensive number of significant pairs, they are not all listed here.

across homogeneous datasets. In contrast, the DF method not only provides a more robust solution for handling data from multiple sources but also proves to be consistent in performance across multiple runs.

Utilising transfer learning, even when first training with the fused dataset, does not lead to better overall performance, even on the dataset that was trained on in the second phase. This suggests a potential risk of overfitting or catastrophic forgetting during the second training phase, where the model might adapt too strongly to the specifics of the second dataset, losing some of the generalised knowledge acquired during the first phase (from the fused data). This contrasts with the single-phase DF approach which aims to learn a balanced representation from the start. However, the preprocessing methods used in the DF algorithm play a crucial role in enhancing performance, particularly for Dataset B, which required downsampling to meet the algorithm's requirements. This demonstrates the algorithm's adaptability to variations in data characteristics and its potential for handling real-world scenarios where data collection specifications may be inconsistent.

Although the performance across different training approaches on Dataset B is lower than on Dataset A, using transfer learning from the fused dataset to Dataset B yields results that are on par with training solely on Dataset B using the traditional approach. Simultaneously, this approach achieves strong performance on Dataset A. However, the DF algorithm still outperforms all transfer learning approaches in terms of overall performance on both test datasets.

The box and whisker plot in Figure 4.4, illustrating the spread of the results, shows that the DF approach did not produce any outlying results from the 10 runs, indicating high levels of consistency. However, it is also evident that the transfer learning approaches exhibit the lowest deviation in results compared to a single training phase on one dataset, whether using traditional training approaches or the proposed DF method. This highlights the potential benefits of combining transfer learning with the DF approach to achieve even more consistent and reliable performance across datasets. However, the results show that transfer learning may not be the best approach, especially in this problem domain, to achieve the best performance, likely due to the overfitting risks mentioned earlier.

The KW test results presented in Table 4.9 demonstrate that the experimental results yield a statistically significant outcome, with a p-value close to 0. This highlights the relevance of the differences observed among the various training approaches.

4.4.2 Experiment 2 - Varying data volume Analysis

The results of Experiment 2, presented in Table 4.11 and visualised in Figure 4.5, demonstrate that the DF approach significantly outperforms other training approaches across varying volumes of training data. Furthermore, when utilising only 6.25% of the training data, the model still surpasses other training approaches that use 100% of the data, achieving an Average F1 score of 0.788. As expected, most approaches experience decreased performance when using less data, with the proposed DF approach following this trend closely. However, it is worth noting that the performance reduction is minimal, with only a 4.04% drop compared to the baseline results.

This sustained high performance, even with drastically reduced training data (down to 6.25%), strongly suggests significant data redundancy within the original individual datasets, particularly when used with traditional training methods. The DF algorithm appears to mitigate this redundancy. By fusing data from multiple homogeneous sources, it likely creates a training sample that better represents the true population distribution of healthy motor operation within this domain. This improved representation allows the model to generalise effectively using significantly less data from each individual source, compared to relying on potentially redundant information within a single, larger dataset.

Interestingly, not all approaches follow this pattern. For instance, transfer learning from the fused dataset to dataset B performs best when using 50% of the data, while transfer learning from the fused dataset to dataset A achieves optimal results with only 6.25% of the data. For these approaches, there does not appear to be a clear advantage to using more data, leading to the conclusion that, for training unsupervised anomaly detectors, using more data is not always beneficial. While this may not hold true for other tasks such as fault classification, those tasks are beyond the scope of the present study and will be explored in future works.

The box and whisker plot in Figure 4.5 reveals that the performance of the anomaly detection model trained with DF is less consistent and has a larger spread when using less data. However, examining the results from other training approaches shows that this is not always the case. In some instances, such as Transfer Learning from Dataset A to Dataset B, the spread is largest when training with the full dataset. Additionally, as the KW test results in Table 4.12 show, the differences between the groups are statistically significant (p ¡ 0.001).

By outperforming all other models while using only a small fraction of the training data, the DF method addresses the common challenge of not having sufficient training data from each data source. In this experiment, as shown in Table 4.10, the estimated FLOPs needed for training the model with 6.25% of the data dramatically decreased from 4.92×10^{12} to 3.08×10^{11}, representing a 93.7% reduction in computational power. This significant decrease is especially noteworthy when contrasted with the minor 4.04% reduction in performance. The DF approach effectively utilises less data, showcasing its potential to contribute to more sustainable and environmentally friendly AI development. Aligned with the principles of Green AI, which emphasise efficiency and reducing the environmental impact of training AI models, the results of Experiment 2 highlight the superior performance of DF. Its crucial implications for real-world applications demonstrate its ability to address the issue of limited training data while promoting more sustainable practices in AI development, providing a valuable contribution towards SDG goals 12 and 13 with reduced computational power requirements and hence energy consumption.

4.4.3 Experiment 3 - Varying dataset ratio Analysis

The results of the experiment, shown in Table 4.14 and Table 4.15, reveal that the DF algorithm outperforms all other training approaches in terms of Average F1 score. The best performance is achieved with a 30:70 ratio (Dataset A: Dataset B), resulting in an Average F1 score of 0.827, closely followed by the baseline experiment (50:50) with an Average F1 of 0.821. In comparison, the next best performing training approach was transfer learning from the fused dataset to Dataset B (TL – DF to Dataset B), which had an Average F1 score of 0.751. The KW test results in

Table 4.16 confirm that these differences are statistically significant (p ¡ 0.001).

The box and whisker plot in Figure 4.6 indicates that the transfer learning approaches involving the fused dataset generally exhibit less spread compared to other methods. Furthermore, the spread of the DF results appears to be the most consistent across different experimental settings. One might hypothesise that the box plot trend would indicate an increase in generalised performance as the balance between samples from each dataset increases. Interestingly, this is not the case.

While the results empirically demonstrate that an imbalance in datasets mostly does not affect the stability of the results for DF, a clear trend is observed, wherein the average F1 score decreases as more of Dataset A is used for training. This trend is more evident in some of the other approaches, such as T-Dataset A, TL-Dataset B to A, and the mixed dataset (MD). For instance, training with the Mixed Dataset at a 40:60 ratio yields an F1 score of 0.889 on Dataset A, 0.509 on Dataset B, and an overall average of 0.699. Conversely, training with a 90:10 ratio results in an F1 score of 0.890 on Dataset A, 0.300 on Dataset B, and an overall average of 0.595. These results show that the performance on Dataset A can be maintained with less data while improving the performance on Dataset B with more data.

The observed trend leads to an intriguing conclusion: there is a higher benefit to training with Dataset B compared to Dataset A to achieve a more generalised anomaly detection performance. Although both Dataset A and Dataset B represent healthy data from similar 3-phase motors operating in the same domain, Dataset B exhibits a higher level of noise, as seen in Figure 4.2h, and a slightly higher spread in the PCA plot (Figure 4.3). These observations suggest that Dataset B is a more complex or diverse dataset to learn from. Training on this more challenging dataset might force the model to learn more robust and invariant features of "healthy" operation, which then generalise better not only to unseen data from Dataset B but also implicitly to the relatively cleaner data of Dataset A. Relying too heavily on the cleaner Dataset A might lead the model to learn features that are too specific and less robust to the variations encountered in Dataset B or potentially other unseen motors. This implies that incorporating a larger variety of healthy data, even if noisier, in the training dataset contributes to an overall better anomaly detection performance across the domain.

4.4.4 Further Remarks and Limitations

In the context of Transfer Learning, it is worth noting that better results on the dataset trained on in the second phase are not guaranteed. The results of the experiments demonstrate that swapping the transfer learning training phases produces different outcomes, which indicates that there is no clear approach to determine which dataset should be used in each phase without conducting additional testing. DF mitigates this issue by training on both datasets simultaneously. Moreover, as discussed in Section 4.4.1, the second phase of transfer learning carries the risk of catastrophic forgetting or overfitting, potentially moving the model weights away from a well-generalised state towards the specific nuances of the target dataset, thereby degrading overall performance.

The conclusions drawn from Experiment 3 suggest that there is a greater benefit to training on a higher number of samples from Dataset B, as opposed to Dataset A, in order to achieve a more generalised performance. However, identifying this preference presents a challenge, as a similar experiment must be conducted to reach this determination. Despite this issue, the DF approach offers a solution by providing a more stable performance across multiple experimental settings. This advantage becomes particularly significant in practical applications where time and resource constraints may render extensive experimentation unfeasible. By addressing these concerns and offering more consistent results, DF shows potential as an effective method for training unsupervised anomaly detection models, particularly in situations where the optimal dataset and training phase cannot be easily determined. The robust performance of the DF approach, combined with its ability to accommodate various experimental settings, positions it as a valuable tool for practical use cases.

One important consideration regarding the DF algorithm is its handling of temporal continuity. The process involves batching based on zero-crossings within each dataset and then interleaving these batches. While this preserves continuity **within** a batch originating from a single source, it inherently introduces discontinuities at the transition points **between** batches from different datasets. In the time domain, this might appear as abrupt shifts in signal characteristics. However, for anomaly detection tasks focused on identifying source-invariant deviations or irregularities (like specific fault signatures), these batch transitions might be less detrimental, as the model learns to recognise patterns independent of the immediate preceding batch's origin. In the frequency domain, as suggested by the fused spectrum in Figure 4.2(i), these transitions could potentially introduce minor artefacts or broaden spectral peaks compared to a continuously sampled signal from a single source. While the current study demonstrates strong performance in the time domain, applying DF directly to frequency-domain inputs might require additional smoothing techniques or adaptations to the fusion process itself to mitigate potential artefacts caused by these discontinuities. Future research could explore smoothing filters applied post-fusion or alternative frequency-domain fusion strategies.

In tasks requiring long-term degradation or forecasting, where the sequential nature of the data is paramount, the batching and fusion method may not be optimal. In such tasks, training data are primarily aimed at identifying the trend of data behaviour over time from a specific source, which may be obscured when alternately mixing datasets. However, for other tasks where the focus is on identifying data source invariant changes — those consistent regardless of the source — the DF algorithm proves to be highly beneficial. This methodology enables the creation of a balanced training set that is more representative of the population distribution of the problem at hand. This approach, therefore, demonstrates substantial value in tasks such as classification or anomaly detection, where the detection of irregularities or deviations is more critical than following a specific trend. In contrast, in forecasting tasks, where the continuity and trend of a single data source are paramount, alternative methods preserving the sequential integrity within each source may yield better models.

The homogeneity assumption in the proposed DF technique also presents a limitation, particularly when handling real-world datasets that often exhibit some degree of non-homogeneity such as in this case, motors of different sizes and rotational speeds.

4.5 Conclusion

This chapter presents a time-series dataset composition approach called the DF algorithm, designed to address challenges in achieving generalised anomaly detection performance across multiple homogeneous data sources. The proposed algorithm was validated using a case study involving motor current signals, demonstrating that the fused dataset retains salient features from both source datasets while clustering in the middle of both datasets when PCA is applied. The algorithm was then tested on an anomaly detection task and compared to conventional training approaches, with empirical results showing that the DF algorithm significantly outperforms other methods in terms of average performance across both datasets.

Additionally, further experiments were conducted to assess the performance of the proposed approach under non-ideal conditions. Experimental results indicate that the DF approach remains superior even when reducing the number of data samples, with only a 4.04% reduction in performance despite using only 6.25% of the training data, resulting in a 93.7% reduction in computational power required for training. When evaluating the model's performance with imbalanced numbers of samples from each dataset, the proposed approach proved stable across different sample ratios. These findings highlight significant benefits in the context of Green AI, which emphasises sustainable AI model development, as well as practical feasibility due to the algorithm's resilience under non-ideal conditions.

While the DF algorithm has shown promising results for homogeneous datasets, adapting the proposed method to enable compatibility with non-homogeneous datasets is a crucial next step in enhancing its applicability. This can potentially be accomplished using domain adaptation techniques, which would further strengthen the generalisability of the proposed method. Chapter 5 explores this direction by introducing ODT, a novel pre-processing algorithm designed to standardise and align the frequency components of signals from different motors.

Chapter 5

Heterogeneous Induction Motor Current Dataset Fusion for Efficient Generalised MCSA-based Fault Classification

5.1 Introduction

The industrial application of machine learning algorithms for fault detection and diagnostics in IMs is often hindered by the diversity of operating conditions, motor-specific characteristics, and the scarcity of labelled fault data. These factors create significant barriers to developing universally applicable models. To overcome these challenges, this chapter introduces the Order Domain Transformer (ODT), a novel pre-processing algorithm. ODT enhances the learning capability of neural networks and allows for the training of a universal model capable of handling signals from multiple types of IMs. The method optimises the utilisation of available labelled data and is computationally efficient with low overhead, making it suitable for real-time applications. Importantly, the algorithm aligns with the principles of Green AI by significantly improving model performance across multiple motors while moving towards significantly reducing the computational and data resource requirements required for training multiple models, thereby contributing to more sustainable and energy-efficient machine learning practices in industrial settings.

The rest of the chapter is structured as follows: Section 5.2 describes the methodology behind ODT, followed by Section 5.3, which presents the experimental setup and results. Section 5.4 discusses the implications of the findings of the study, and Section 5.5 concludes the chapter.

5.2 Order Domain Transformer

5.2.1 Fitting Stage

The fitting stage is the preliminary phase where key parameters that guide the transformation are determined. These parameters include the reference sampling frequency and the number of orders to analyse.

Reference Sampling Frequency

The choice of a reference sampling frequency, denoted as $f_{s_{\text{target}}}$, is a critical step that affects the quality of the signal analysis. The selection should ideally be based on the most commonly present sampling frequency across the datasets being examined. This is because upsampling a signal, while not adding any new information, increases computational complexity, and downsampling poses the risk of information loss, especially if the original signal has frequency components near or above the Nyquist frequency.

Number of Orders

The number of orders, N_{orders}, specifies the harmonics of the fundamental frequency that should be captured in the frequency domain representation. This is converted to a target fundamental frequency,$f_{o_{\text{target}}}$:

$$f_{o_{\text{target}}} = \frac{f_{s_{\text{target}}}}{2N_{\text{orders}}} \tag{5.1}$$

5.2.2 Resampling Stage

The resampling stage is where the main transformation to the signal occurs.

Whilst generally the motor's fundamental frequency, f_o, is available, there may be times when this information is not readily available or easily collectable. In this case, f_o of the signal can be estimated using FFT-based peak detection. This approach does not account for motor slip, which can affect the accuracy of the frequency representation. However, ignoring motor slip eliminates the need for tachometer data, making the method more versatile as it solely relies on the motor current signal.

5.2.3 Signal Interpolation

The number of periods within the signal, denoted as T, is calculated:

$$T = \frac{Nf_o}{f_s} \tag{5.2}$$

where N is the number of samples in the TS signal, f_o is the estimated or provided fundamental frequency, and f_s is the current sampling frequency.

With T known, the new length of the interpolation axis, S, can be calculated:

$$S = \frac{f_{s_{\text{target}}}T}{f_{o_{\text{target}}}} \tag{5.3}$$

The resampled signal $y(x)$ is generated by linearly interpolating between the old and new axes, denoted by T and S respectively:

$$y(x) = y_1 + \frac{y_2 - y_1}{S - T} \times (x - T), \quad x = 1, 2, \ldots, S \tag{5.4}$$

where x is the point on the new resampling axis, y_1 and y_2 are the original points above and below x on the old axis T, and S is the end point for the newly resampled axis

5.2.4 Transform Stage

At this stage, the signal $y(x)$ is interpolated with reference to $f_{o_{\text{target}}}$ and $f_{s_{\text{target}}}$. Any signal transformation can be used now, but the present study will explore the use of the Welch Power Spectral Density (PSD) estimation [175] to extract salient features from the signal in the form of fault frequencies to facilitate easier fault classification. All signals that undergo ODT pre-processing will be aligned, which also means that the frequency bins will be aligned to make comparison easier.

Welch's method [175] addresses the issue of measuring PSD in non-stationary signals by averaging the power content across the signal using a sliding window of periodograms with a specified overlap. The advantage of this is a reduction of noise across the PSD, at a cost of frequency resolution in comparison to using the entire signal. For a signal $x[n]$ with N samples, the Welch method first splits the signal into K segments with M points in each segment, with S points shifted between each segment. This means that there is a $M-S$ overlap between each segment. Equation 5.5 is then used to calculate the periodogram for each segment. The periodogram of each segment is then averaged using Equation 5.6 to obtain the Welch estimate of the PSD.

$$P_{x,N}(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \tag{5.5}$$

$$S_{x_{welch}}[\omega] = \frac{1}{K}\sum_{k=0}^{K-1} P_k[\omega] \tag{5.6}$$

where ω is the angular frequency, and j is the imaginary unit, $\sqrt{-1}$

Algorithm 2 Order Domain Transformer

```
1: **Input:** Original signal $x(t)$, Sampling frequency $f_s$, Target sampling frequency $f_{s_{target}}$, Number of orders $N_{orders}$
2: **Output:** Resampled signal $y(x)$
3: **Fitting Stage:**
4: Calculate $f_{o_{target}}$ using Equation 5.1
5: **Resampling Stage:**
6: Estimate $f_o$ using FFT-based peak detection or use provided $f_o$
7: Calculate $T$ using Equation 5.2
8: Calculate $S$ using Equation 5.3
9: **Signal Interpolation:**
10: Perform linear interpolation using Equation 5.4 to obtain $y(x)$
11: Return $y(x)$
```

5.2.5 Computational Complexity

Table 5.1 breaks down the computational complexity of each step in the ODT method using Big O notation.

The computational complexity of the ODT algorithm primarily consists of linear interpolation, however, factoring in the FFT used for estimating f_o, and the final

Table 5.1: Computational Complexity of the ODT algorithm for each step

Operation	Complexity
f_o estimation	$O(nlogn)$ [176]
ODT	$O(n)$
Welch's Method	$O(nlogn)$ [176]

welch transformation, the computational complexity increases significantly. In scenarios where f_o is already known and FFT-based estimation is not needed, the differentiating computational complexity between using Welch's method with or without ODT comes from the linear interpolation step. This makes the ODT algorithm a low-overhead algorithm that is suitable to be applied in real-time processes.

5.3 Experimental Results

To validate the performance of the proposed technique on classification tasks relating to motor condition, two experiments were undertaken:

- **Baseline Test:** A baseline test was conducted using a traditional train-test split on a curated dataset composed of two datasets to ensure that a model trained using the proposed ODT pre-processing maintains similar performance to a model trained without.
- **Unseen Dataset Test:** In order to validate the claims made in this work regarding the improved generalisation capability of an NN using this technique, a new labelled dataset was introduced which was not trained or validated on previously by the models. A model trained using without ODT pre-processing was compared with a model trained using ODT.

The next sections briefly break down the NN model architecture used for the experiment, introduce the datasets used, and present the main results of this study.

5.3.1 Transformer Network

Developed initially for sequence-to-sequence ML applications, the Transformer NN architecture [4] is designed to take in sequential data and process the entire input in one go using the Attention Mechanism, as opposed to traditional Recurrent NNs which process data sequentially without attention.

The architecture of the model used in this study is shown in Figure 5.1. The transformer block first applies layer normalisation across the features being input into the network. In this case, the features are the frequency spectra of the 3-phases current signals of the motor. Next, a multi-head attention layer is employed to determine the relevance of a particular frequency to the other frequencies in the spectra. The weighting vectors produced by the attention layer are applied to the input and passed onto 1D Convolutional layers, which output the data in the correct structure to be fed into the next transformer block. Each transformer block only receives the output of the previous transformer block as an input and has no information on any other values in the NN.

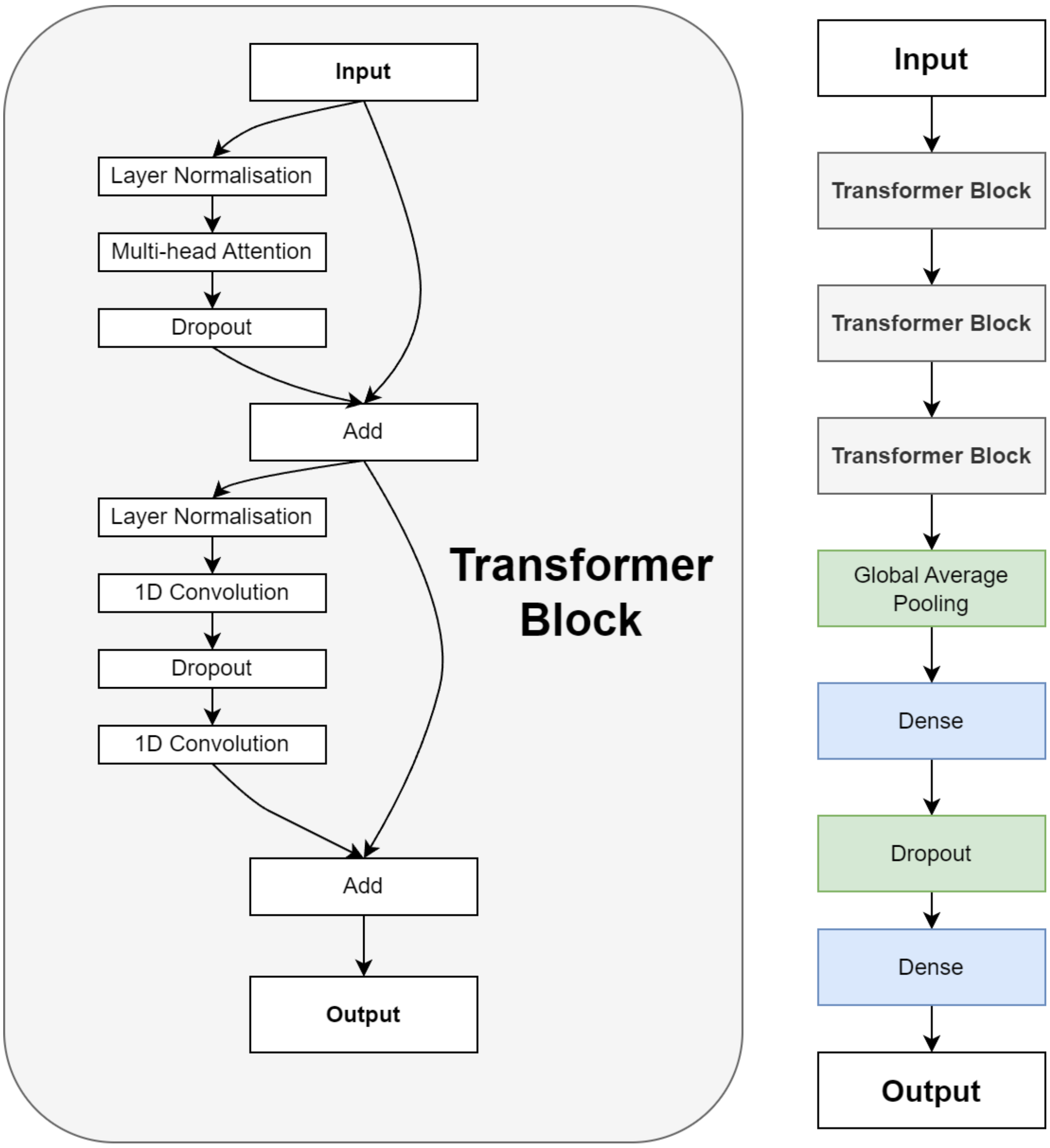


Figure 5.1: Transformer architecture used in this study, courtesy of [4]

Table 5.2: A breakdown of the datasets used in this study

Dataset	Sampling Frequency (Hz)	Operating Frequencies (Hz)	Other Specifications
Dataset A: Broken Rotor Bar Dataset [28]	1000	50, 60	4 poles, 50Hz: 1/3 HP/ 248W, 1.85A 60Hz: 1/2 HP/372W 2.2A
Dataset B: Vibration and Motor Current Dataset of Rolling Element Bearing Under Varying Speed Conditions for Fault Diagnosis [29]	26,500	60	4 poles, 1 HP, 3010 rpm, 380V, 3A, 0-4Nm load
Dataset C: Experimental Database For Detecting And Diagnosing Rotor Broken Bar In A Three-Phase IM [30]	55,611	60	4 poles. 220V / 380V, 3.02A / 1.75A 4.1 Nm torque, 1715 rpm speed

Table 5.3: Number of samples in each fault class for the datasets presented in Table 5.2, where X represents no coverage of the fault class

Dataset	Number of Samples				
	Normal	Broken Rotor Bar	Bearing	Unbalance	Misalignment
Dataset A	134	221	X	X	X
Dataset B	3	X	9	15	9
Dataset C	80	320	X	X	X
Total	**217**	**541**	**9**	**15**	**9**

Once the data has passed through the transformer encoder blocks, it is then decoded to produce a classification. The output of the final transformer block passes through a global average pooling layer to produce a feature map for each category of the classification task. The feature map then passes through two fully connected dense layers, which outputs the one-hot encoded classification of the motor condition. The NN makes use of dropout [89] layers throughout the architecture to mitigate overfitting on the training data and improve overall training performance.

5.3.2 Datasets and Pre-processing

Table 5.2 depicts the datasets used for each of the experiments, and Table 5.3 breaks down the total number of classes in the datasets.

Dataset A focuses on broken rotor bars and operates at sampling frequencies of 1000 Hz and operating frequencies of 50 and 60 Hz. Dataset B analyses rolling element bearings under varying speed conditions with a sampling frequency of 26,500 Hz. Dataset C presents rotor broken bar fault signals with a sampling frequency of 55,611 Hz. The specific fault classes covered by each dataset and the number of samples available for each are detailed in Table 5.3.

For the experiments, 70% of the data from Dataset A and Dataset B is used for training the model. The remaining 30% from these datasets is used for the baseline test in the first experiment to assess the model's performance with and without the ODT preprocessing. For the second experiment, which aims to validate the generalisation capability of the model using ODT, Dataset C will serve as the unseen dataset.

For data pre-processing, both the baseline and ODT-utilising experiments first subtract the mean of the TS signal to remove its DC component. Welch's method is applied, followed by log-scaling, which is crucial for preventing significant harmonic amplitudes from overshadowing important variations in intermediate frequencies, allowing for a more detailed fault classification. Following this, Z-score normalisation is applied, ensuring each frequency spectrum has a mean, μ, of zero and a standard deviation, σ of one:

$$z = \frac{x - \mu}{\sigma} \tag{5.7}$$

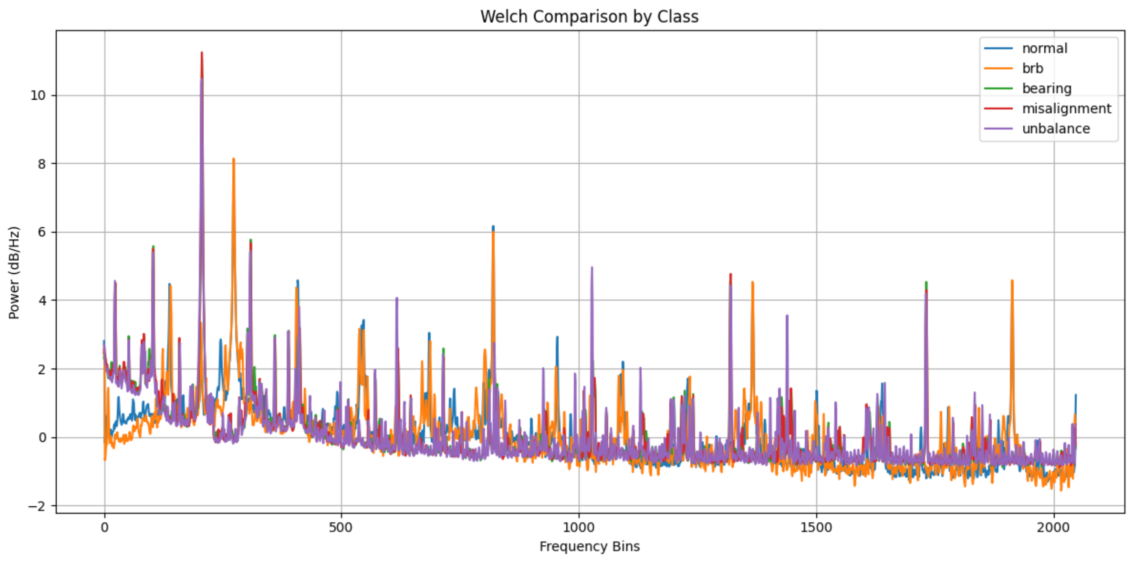
This standardisation aids in maintaining fault class distinctions and ensures all frequency regions are equally weighted during NN training, leading to faster convergence. In the experiments leveraging ODT, the ODT pre-processing step precedes these operations. In both cases, the number of Fast Fourier Transform points, $nfft$, value of 4096 is used, resulting in each sample having the shape $(2048, 3)$.

The resulting signals for each class from the aforementioned pre-processing steps are illustrated in Figure 5.2, where (a) is the non-ODT signal and (b) is the ODT signal. The figure averages 9 samples for each class, offering a more balanced and consistent view of how the ODT and non-ODT signals compare across typical scenarios. In the figure, the non-ODT signals display harmonics that are scattered across the frequency spectrum, making it challenging to identify consistent patterns. On the other hand, the ODT-processed signals exhibit harmonics that are more uniformly distributed and consistent, facilitating easier interpretation and analysis. Additionally, it's worth noting that the x-axis in the figure represents the sample index, indicating that the signals are truly aligned and standardised, making them more suitable for NN input.

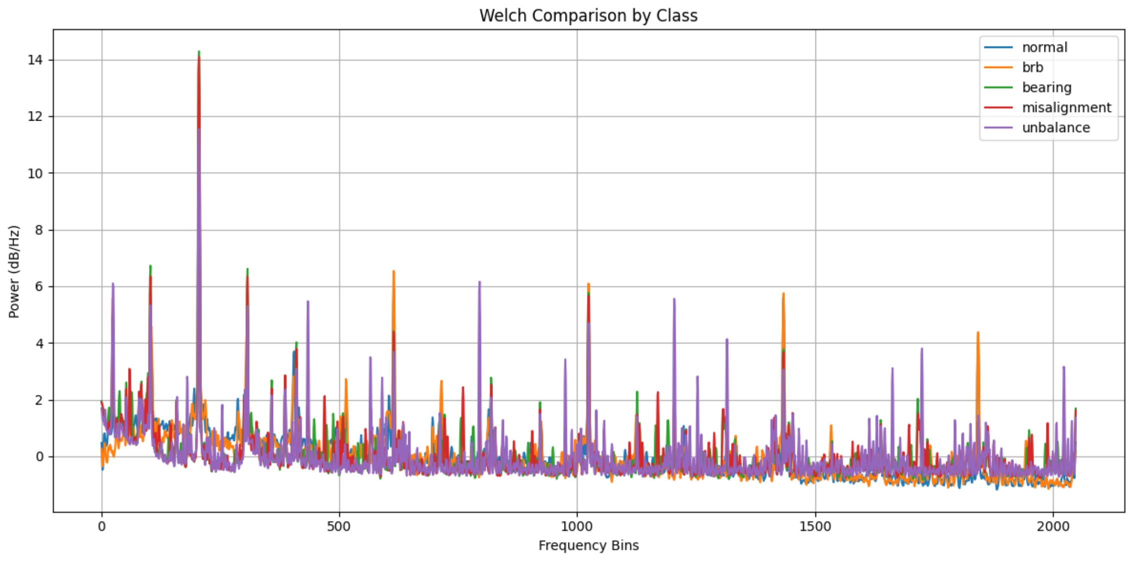
5.3.3 Evaluation Metrics

To evaluate the performance of the proposed method, we use the following metrics:

- **Precision:** The ratio of true positive predictions to the total number of positive predictions made.
$$\text{Precision} = \frac{TP}{TP + FP}$$
- **Recall:** The ratio of true positive predictions to the total actual positives.
$$\text{Recall} = \frac{TP}{TP + FN}$$
- **F1 Score:** The harmonic mean of precision and recall.
$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
- **Accuracy:** The ratio of correct predictions to the total number of predictions.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Macro Average F1:** The unweighted mean of F1 Scores for all classes.
$$\text{Macro Avg F1} = \frac{1}{N} \sum_{i=1}^{N} \text{F1}_i$$



(a) Without ODT



(b) With ODT

Figure 5.2: A comparison of the Welch Spectra of each class, with and without ODT preprocessing, where each spectra shown is the average of 9 spectra from each class

Table 5.4: Tuned hyperparameters for the model used in the experiments

Parameter	Value
Optimiser	Adam
Learning Rate	0.0001
Transformer Blocks	3
Number of Heads	3
Dropout	0.4
Number of parameters	298,331

- **Weighted Average F1:** The average of F1 Scores for all classes, weighted by the number of samples in each class.

$$\text{Weighted Avg F1} = \frac{1}{\sum w_i} \sum_{i=1}^{N} w_i \times \text{F1}_i$$

where TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively. N is the number of classes and w_i is the number of samples in class i.

Each experiment run was repeated 5 times for experimental validity. Class weighting was used during model training to combat the significant class imbalance present in the dataset [177]. Categorical Cross entropy was used as the loss function. The training was set for 200 epochs, but Early Stopping with a patience of 20 epochs monitoring the validation loss was used to prevent overfitting.

For hyperparameter tuning, a grid search was conducted using the test accuracy from the initial baseline experiment as the metric. The tuned hyperparameter values are presented in Table 5.4.

The experiment results are organised into two primary tables. Table 5.5 summarises the results from the baseline test, where a model was trained on 70% of Dataset A and Dataset B and subsequently evaluated on the remaining 30%. Table 5.7, displays the findings from the unseen data test, in which the trained model was applied to Dataset C, a dataset not previously encountered during the training or validation process.

To evaluate the statistical signifcance and stability of the results, a KW test was conducted on the experimental results; particularly the Weighted Average F1 Scores. Boxplots showing the variance in the experiment repetitions are illustrated in Figure 5.3 and Figure 5.4 for the baseline and unseen experiments respectively, and the KW test results in Table 5.6 and Table 5.8 respectively.

One important thing to note regarding the unseen data test is that although there are only 2 fault classes, the model is trained to detect 5 fault classes which adds complexity to the problem.

5.4 Discussion

The results in Table 5.5 show that the model using ODT pre-processing generally maintains similar performance on the test set when compared to the non-ODT model. This claim is also supported by the box plot in Figure 5.3, which shows

Table 5.5: Experimental results for 5 iterations of baseline test, where P is precision and R is Recall

Model	Metric	Bearing			Broken Rotor Bar			Misalignment			Normal			Unbalance			Accuracy	Macro Avg F1	Weighted Avg F1
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1			
Non-ODT	Best	1.00	0.50	0.67	1.00	1.00	1.00	0.67	1.00	0.80	1.00	0.96	0.98	0.75	1.00	0.86	0.97	0.86	0.97
	Average	0.60	0.30	0.40	0.93	0.96	0.94	0.53	0.80	0.64	0.94	0.85	0.89	0.66	1.00	0.79	0.90	0.73	0.90
	Std Dvn	0.55	0.27	0.37	0.06	0.04	0.03	0.30	0.45	0.36	0.06	0.11	0.07	0.08	0.00	0.06	0.04	0.10	0.05
ODT	Best	1.00	0.50	0.67	0.98	1.00	0.99	0.00	0.00	0.00	1.00	0.96	0.98	0.50	1.00	0.67	0.95	0.66	0.94
	Average	0.80	0.40	0.53	0.96	0.95	0.96	0.33	0.60	0.43	0.93	0.91	0.92	0.69	0.93	0.76	0.92	0.72	0.91
	Std Dvn	0.45	0.22	0.30	0.02	0.06	0.04	0.31	0.55	0.39	0.08	0.03	0.05	0.23	0.15	0.12	0.03	0.08	0.03

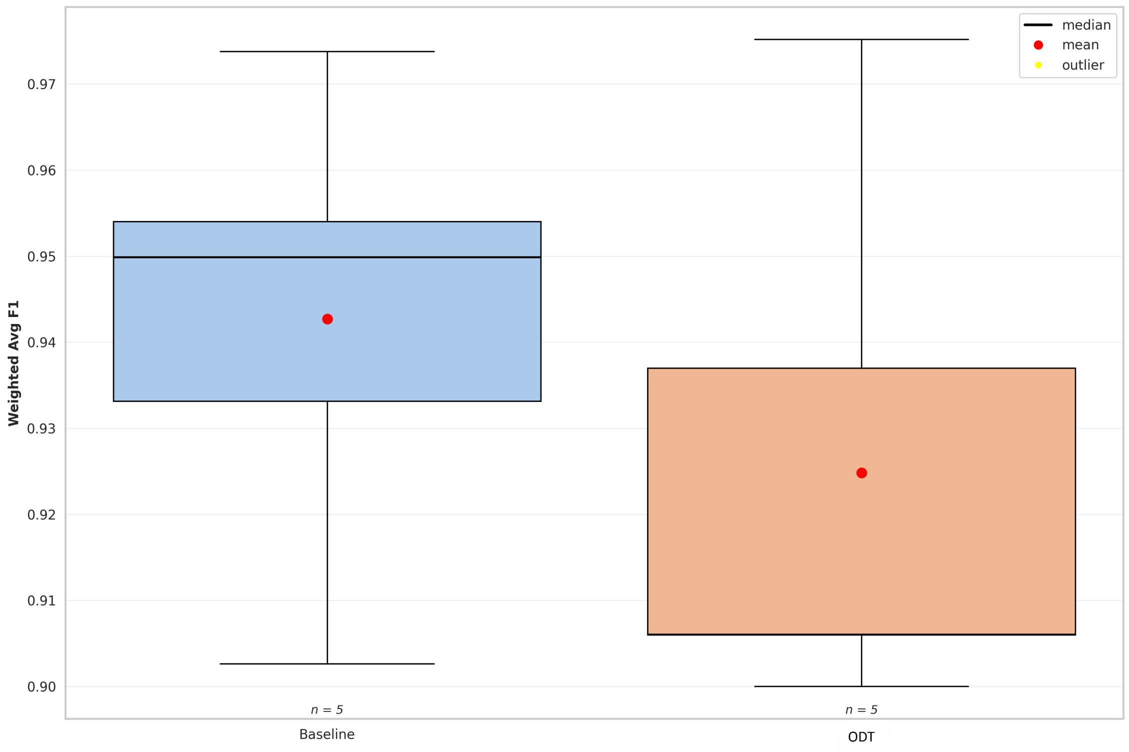


Figure 5.3: Box plot for 5 iterations of baseline test on the Weighted Average F1 Scores

Table 5.6: Kruskal-Wallis test results comparing Baseline and ODT models.

Model	**n**	**Mean**	**SD**	**Median**	**Chi-square**	***p*-Value**
Baseline	5	0.94	0.03	0.95	0.53	0.465
ODT	5	0.92	0.03	0.91		

Note: No significant difference was found between the models (p ¿ 0.05), indicating that ODT maintains performance comparable to the Baseline.

Table 5.7: Experimental results for 5 iterations of unseen dataset test, where P is precision and R is Recall

Model	Metric	Broken Rotor Bar			Normal			Accuracy	Macro Avg F1	Weighted Avg F1
		P	R	F1	P	R	F1			
Non-ODT	Best	0.83	0.40	0.54	0.21	0.65	0.32	0.45	0.29	0.50
	Average	0.79	0.26	0.38	0.19	0.71	0.30	0.35	0.28	0.37
	Std Dvn	0.05	0.10	0.11	0.02	0.13	0.03	0.06	0.09	0.09
ODT	**Best**	**0.81**	**0.94**	**0.87**	**0.32**	**0.09**	**0.14**	**0.77**	**0.25**	**0.72**
	Average	**0.79**	**0.83**	**0.81**	**0.13**	**0.05**	**0.07**	**0.67**	**0.20**	**0.66**
	Std Dvn	**0.01**	**0.08**	**0.05**	**0.11**	**0.03**	**0.04**	**0.07**	**0.04**	**0.04**

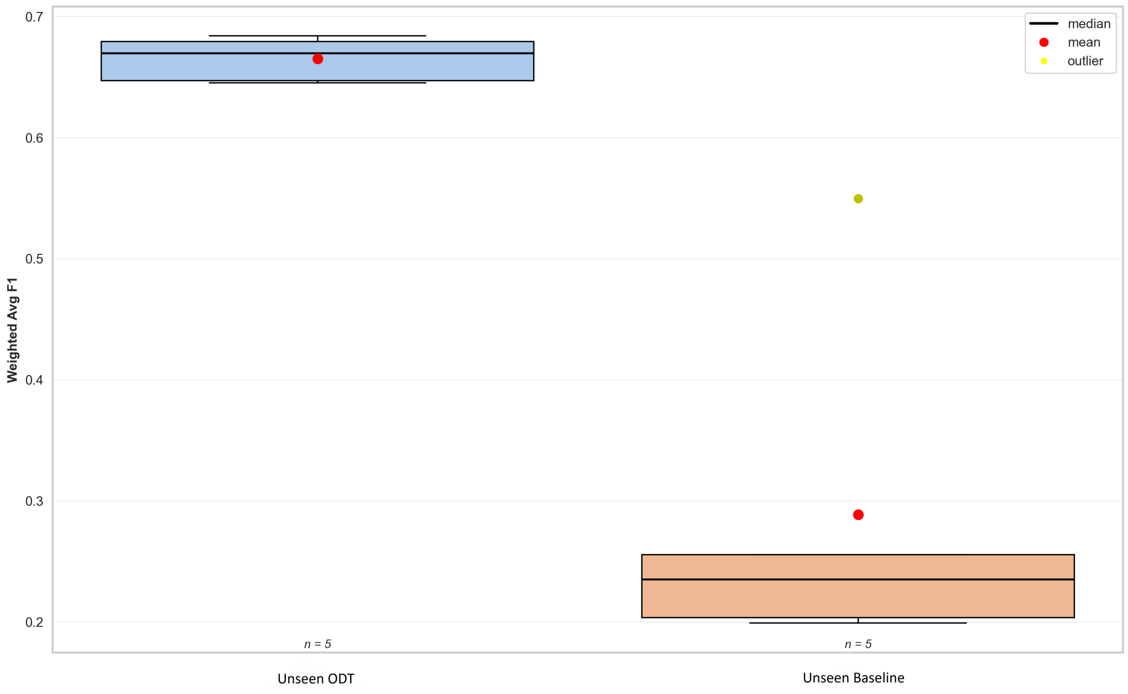


Figure 5.4: Box plot for 5 iterations of unseen test on the Weighted Average F1 Scores

Table 5.8: Kruskal-Wallis test results for Weighted Average F1 Scores on the unseen dataset.

Model	n	Mean	SD	Median	Chi-square	*p*-Value
Unseen ODT	5	0.67	0.02	0.67	6.82	0.009*
Unseen Baseline	5	0.29	0.15	0.24		

Note: * indicates significant difference (p ¡ 0.05). Post-hoc analysis confirms the significant difference between unseen Baseline and unseen ODT models.

that the results largely overlap, and the KW results in Table 5.6, which shows a p-value of 0.465 between the two groups, rejecting the null hypothesis of a statistical difference. This is largely because the non-ODT model is trained to accommodate the various operating conditions in the dataset, enabling it to generalise to the test set as effectively as the ODT model. The non-ODT model's best performance, with an accuracy of 0.97, slightly outperforms the best ODT model, which has an accuracy of 0.95, however, the KW statistical test negates this difference as non-statistically significant. The non-ODT model attains an average accuracy of 0.9, a Macro Average F1 of 0.73, and a Weighted Average F1 of 0.9. In contrast, the ODT model performs at an average accuracy of 0.92, a Macro Average F1 of 0.72, and a Weighted Average F1 of 0.91. The standard deviation in accuracy for the ODT model is 0.03, as compared to 0.04 for the non-ODT model, suggesting a more consistent performance from the ODT model.

For the unseen test, the results in Table 5.7, as well as the statistical tests in Table 5.8, provide strong empirical evidence that the ODT model significantly outperforms the non-ODT model when faced with a completely new motor. This is likely because aligning the harmonics through ODT pre-processing allows the NN to more effectively capture the salient features related to motor faults during training, thereby enhancing the model's ability to generalise across different motors. On average, the non-ODT model achieves an accuracy of 0.35, a Macro Average F1 of 0.28, and a Weighted Average F1 of 0.37. In contrast, the ODT model scores an average accuracy of 0.67, a Macro Average F1 of 0.2, and a Weighted Average F1 of 0.66. The KW test results in a p-Value of 0.009, and post-hoc analysis with Bonferroni correction confirms the significant difference between the two groups of Weighted Average F1 scores. The notably lower Macro Average F1 score for the ODT model can be attributed to one of its current limitations: it does not account for how ”normal” operation may vary significantly among different motors, leading to a higher likelihood of misclassification due to differing environmental conditions.

To summarise, in situations where models are solely applied to specific, consistent motor types, the non-ODT model proves sufficient, offering solid performance without additional pre-processing. However, when the goal is robust fault classification across diverse motors and environments, the ODT model is more effective, evidenced by the results in Table 5.7.

5.4.1 Advantages of ODT

Based on the outcomes of the experimental work, we can summarise the following advantages for using ODT:

- **One Model for Multiple Motors:** Only one model needs to be trained, which can then be applied across different motors with varying characteristics. This results in a reduction of computational resources required for model training, aligning with Green AI development practices.

- **Optimal Utilisation of Labeled Data:** A universal model also allows for better utilisation of available labelled motor data, pooling the data from different motors to create a richer, more diversified training set.

- **No additional data requirements:** The method is applied to motor current signals which can be collected non-invasively, making it more easily applicable across a broad range of settings.

5.4.2 Current Limitations of ODT

ODT has proven valuable for enhancing NN generalisation across unseen motors. However, it presents certain limitations:

- **Accounting for operating environments:** ODT does not account for the varying environments in which motors operate. The experimental results in Table 5.7 reflect this limitation. While faults exhibit distinct frequency spectra signatures, the "normal" operation can differ between motors, increasing misclassification risks due to varied environments. The Dataset Fusion method [22] is being considered for future studies to address this.
- **Accounting for motor slip:** ODT may not accurately account for motor slip when estimating f_o through spectral methods since the algorithm normalises based on the estimated f_o, which might not align with the motor's actual operating frequency. Nevertheless, the benefits of harmonic alignment outweigh this limitation.

5.5 Conclusion

ODT addresses key challenges in motor fault detection by standardising and aligning the frequency components of signals from different IMs. The algorithm works by transforming the frequency spectra of motor signals into a standardised "order" domain, thereby enhancing the learning capability of NNs on heterogeneous motor data. Experimental results show that using ODT maintains statistically similar performance to non-ODT data when a model is trained on data from a motor it will be applied to, but results in a statistically significant increase in cross-motor generalisation classification accuracy from 0.35 with standard pre-processing to 0.67 with ODT pre-processed data. These results are promising and pave the way for the development of more efficient, generalised models that utilise all available data in the problem domain. Furthermore, the potential for reduced training times, and hence reduced computational power, compared to multi-model training aligns with a greener approach to AI development.

Chapter 6

Conclusion

This thesis presents significant advancements in the field of IM fault detection through the application of novel ML architectures and signal processing techniques. The overarching goal of this research was to enhance the generalisation capabilities of fault detection models across different motors and operating conditions, thereby addressing critical limitations in current methodologies. The contributions and findings of this research are encapsulated in three key areas: the development of a Multi-Channel LSTM-Capsule Autoencoder Network for Anomaly Detection, the introduction of DF Algorithm for Generalised Anomaly Detection on homogeneous data sources, and the proposal of ODT for Fault Classification on heterogeneous data sources.

The literature review conducted in Chapter 2 highlights several key points regarding fault detection, generalisation techniques, and cross-domain fault diagnosis. Traditional methods of fault detection rely heavily on hardware redundancy, which can be inefficient and costly. Soft sensing methods, such as those using NNs, offer more efficient and effective alternatives but require extensive labelled data, which can be expensive and time-consuming to produce. Unsupervised learning techniques like AEs have been proposed to mitigate this issue, but they often require large volumes of data and struggle with generalisation. The CapsNet was introduced to address these challenges, offering improvements in training efficiency and spatial context learning. Overall, the literature review emphasises the necessity for more flexible and accurate system modelling that is more representative of real conditions, which sets the stage for the subsequent chapters.

Chapter 3 introduces a hybrid model that combines LSTM networks with Capsule Networks within a multi-channel autoencoder architecture. This model was designed to address issues related to training performance and anomaly detection in multivariate TS data. Empirical evaluations conducted on both a drone dataset and the SKAB anomaly detection benchmark demonstrated that the proposed architecture could efficiently train over fewer epochs, show resilience to overfitting, and achieve state-of-the-art performance in outlier detection. The integration of Capsules within the LSTM framework proved effective in capturing complex temporal dependencies and spatial relationships, enhancing the model's overall performance in anomaly detection tasks. The experimental results highlighted that the multi-channel approach and the inclusion of Capsules led to significant improvements in training efficiency and anomaly detection performance, showcasing the robustness and adaptability of the proposed model.

Specifically, the drone dataset experiments revealed that the proposed model with Capsules converged faster, reaching local minima more efficiently compared to models without Capsules. The anomaly detection accuracy was significantly higher, with the proposed model achieving a precision of 0.91 and a recall of 0.89 on the drone dataset, compared to 0.85 and 0.82 for the standard LSTM model. Furthermore, on the SKAB benchmark, the proposed model outperformed state-of-the-art methods, achieving an F1 score of 0.87, compared to the highest score of 0.78 from other models.

Building upon the foundation established in Chapter 3, Chapter 4 introduces the novel DF algorithm to address the challenge of generalising anomaly detection performance across multiple homogeneous data sources. The architecture proposed in Chapter 3 was employed to validate the effectiveness of the Dataset Fusion approach. This method was rigorously tested using motor current signals from different datasets. The DF algorithm effectively retains salient features from each source dataset while merging them into a unified dataset. This approach significantly improved generalisation performance, allowing for accurate anomaly detection across various datasets. Additionally, the DF algorithm demonstrated robustness under conditions of reduced training data, thereby reducing the computational resources required for training and aligning with the principles of Green AI.

The empirical results showed that the DF method outperformed traditional training approaches. For instance, with only 6.25% of the training data, the DF approach achieved an Average F1 score of 0.788, while traditional methods using the full dataset achieved lower scores. This represents a 93.7% reduction in computational power required for training, as the estimated FLOPs decreased from 4.92×10^{12} to 3.08×10^{11}. The robust performance of the DF method across different data volumes and ratios highlighted its practical applicability and sustainability.

Chapter 5 further extends the work by presenting the ODT, a pre-processing algorithm that standardises and aligns the frequency components of motor signals to enhance cross-motor generalisation in fault classification. The ODT significantly improved the classification accuracy for motors that were not present in the training dataset, addressing a critical need for real-world applications where monitoring systems must handle a variety of motor types and configurations without extensive retraining. By enabling the development of a single model that can generalise across multiple motors, the ODT reduces the need for separate models for each motor type, thereby simplifying deployment and conserving computational resources.

The experimental results demonstrated a substantial improvement in cross-motor generalisation classification accuracy. The ODT model achieved an average accuracy of 0.67 and a Weighted Average F1 score of 0.66 on a blind test with a new motor, compared to the non-ODT model's accuracy of 0.35 and Weighted Average F1 score of 0.37. These findings underscore the effectiveness of the ODT in creating more adaptable and robust fault detection systems, capable of handling diverse motors and operating conditions.

6.1 Recommendations for future work

The findings and advancements presented in this thesis offer a robust foundation for the development of more flexible and adaptable IM fault detection systems. However, several avenues for future research can further enhance the generalisation, efficiency, and applicability of these models.

Chapter 3 utilised raw TS data for anomaly detection in the Multi-Channel LSTM-Capsule Autoencoder. While this approach demonstrated significant improvements in training efficiency and anomaly detection performance, future research could explore the use of different signal representations, such as FFT. Specifically for IM anomaly detection, FFT could enhance the prominence of relevant features by transforming the time-domain data into the frequency domain. This representation may reveal additional patterns and anomalies that are not as evident in the raw TS, potentially improving the model's detection capabilities.

The DF algorithm introduced in Chapter 4 was validated by randomly reducing the dataset to test its robustness under conditions of limited training data. While this method demonstrated that the DF approach can maintain high performance even with significantly reduced data, future work could focus on more systematic approaches to identifying redundant data. Techniques such as active learning, data pruning, or clustering methods could be employed to systematically select the most representative subsets of the data. This would ensure that the reduced dataset retains its diversity and informative value, thereby further optimising computational efficiency and model performance. Furthermore, it is currently unclear whether the advantages of DF will be consistent for other time-series problems. Addressing this question presents an intriguing avenue for future research.

The ODT introduced in Chapter 5 was primarily tested for fault classification in this research, showing substantial improvements in cross-motor generalisation accuracy. However, the potential of ODT for anomaly detection remains unexplored. Future studies could investigate the application of ODT in anomaly detection tasks, leveraging its ability to standardise and align frequency components across different motors. By extending ODT to anomaly detection, the approach could potentially offer a unified solution for both classification and detection tasks, enhancing its utility and applicability in real-world industrial settings. Furthermore, while the current implementation of ODT demonstrated significant improvements in fault classification, there is scope for optimising the NN architectures used in conjunction with ODT. Future research could explore the development and testing of more advanced NN architectures. Optimising the NN architecture for ODT could lead to even greater enhancements in generalisation performance and computational efficiency.

Finally, future research should focus on the real-world deployment and validation of the proposed models. Implementing these techniques in live industrial environments will provide valuable insights into their practical applicability, robustness, and scalability. Real-world testing can uncover additional challenges and opportunities for refinement, ensuring that the developed models meet the demands of industrial applications and contribute effectively to the advancement of Industry 4.0.

Bibliography

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[2] A. Elhalwagy and T. Kalganova, "Multi-channel lstm-capsule autoencoder network for anomaly detection on multivariate data," *Applied Sciences*, vol. 12, no. 22, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/22/11393

[3] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[5] R. A. Fisher, *Statistical Methods for Research Workers.* New York, NY: Springer New York, 1992, pp. 66–70. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_6

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Commun. ACM*, vol. 63, no. 12, p. 54–63, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3381831

[7] J. Lee, H. A. Kao, and S. Yang, "Service innovation and smart analytics for industry 4.0 and big data environment," *Procedia CIRP*, vol. 16, pp. 3–8, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.procir.2014.02.001

[8] M. Javaid, A. Haleem, and R. Suman, "Digital twin applications toward industry 4.0: A review," *Cognitive Robotics*, vol. 3, pp. 71–92, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667241323000137

[9] V. Warke, S. Kumar, A. Bongale, and K. Kotecha, "Sustainable development of smart manufacturing driven by the digital twin framework: A statistical analysis," *Sustainability*, vol. 13, no. 18, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/18/10139

[10] A. Bousdekis, D. Apostolou, and G. Mentzas, "Predictive maintenance in the 4th industrial revolution: Benefits, business opportunities, and managerial implications," *IEEE Engineering Management Review*, vol. 48, no. 1, pp. 57–62, 2020.

[11] M. Sertsöz and M. Kurban, *Energy Efficiency of a Special Squirrel Cage Induction Motor.* Cham: Springer International Publishing, 2018, pp. 219–229. [Online]. Available: https://doi.org/10.1007/978-3-319-89845-2_16

[12] [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/induction-motor-market-101639

[13] G. Niu, X. Dong, and Y. Chen, "Motor fault diagnostics based on current signatures: A review," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–19, 2023.

[14] N. Bessous, S. E. Zouzou, S. Sbaa, and W. Bentrah, "A comparative study between the mcsa, dwt and the vibration analysis methods to diagnose the dynamic eccentricity fault in induction motors," in *2017 6th International Conference on Systems and Control (ICSC)*, 2017, pp. 414–421.

[15] S. Halder, S. Bhat, D. Zychma, and P. Sowa, "Broken rotor bar fault diagnosis techniques based on motor current signature analysis for induction motor—a review," *Energies*, vol. 15, no. 22, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/22/8569

[16] C. N. Okwuosa, U. E. Akpudo, and J.-W. Hur, "A cost-efficient mcsa-based fault diagnostic framework for scim at low-load conditions," *Algorithms*, vol. 15, no. 6, 2022. [Online]. Available: https://www.mdpi.com/1999-4893/15/6/212

[17] T. Yang, H. Pen, Z. Wang, and C. S. Chang, "Feature knowledge based fault detection of induction motors through the analysis of stator current data," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 3, pp. 549–558, 2016.

[18] P. Kumar and A. S. Hati, "Review on machine learning algorithm based fault detection in induction motors," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1929–1940, 5 2021.

[19] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable ai: Environmental implications, challenges and opportunities," in *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu, Eds., vol. 4, 2022, pp. 795–813. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2022/file/462211f67c7d858f663355eff93b745e-Paper.pdf

[20] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 3857–3867, 2017.

[21] S. Lu, R. Yan, Y. Liu, and Q. Wang, "Tacholess speed estimation in order tracking: A review with application to rotating machine fault diagnosis," *IEEE*

Transactions on Instrumentation and Measurement, vol. 68, no. 7, pp. 2315–2332, 2019.

[22] A. Elhalwagy and T. Kalganova, “A dataset fusion algorithm for generalised anomaly detection in homogeneous periodic time series datasets,” *IEEE Access*, vol. 11, pp. 121 212–121 230, 2023.

[23] ——, “Heterogeneous induction motor current dataset fusion for efficient generalised mcsa-based fault classification,” in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2023, pp. 0576–0581.

[24] G. Sternharz, J. Skackauskas, A. Elhalwagy, A. J. Grichnik, T. Kalganova, and M. N. Huda, “Self-protected virtual sensor network for microcontroller fault detection,” *Sensors*, vol. 22, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/2/454

[25] I. D. Katser and V. O. Kozitsin, “Skoltech anomaly benchmark (skab),” *Kaggle*, 2020. [Online]. Available: https://github.com/waico/SKAB

[26] R. G. C. Cunha, E. T. d. S. Junior, and C. M. d. S. Medeiros, “Inter-turn Short-Circuit In Induction Motor.” [Online]. Available: https://www.kaggle.com/datasets/4d260937c50d483e5a888c3d587180a3c53c4c962313c52829de7ec45c383b94

[27] A. Elly Treml, R. Andrade Flauzino, M. Suetake, and N. A. Ravazzoli Maciejewski, “Experimental database for detecting and diagnosing rotor broken bar in a three-phase induction motor.” 2020. [Online]. Available: https://dx.doi.org/10.21227/fmnm-bn95

[28] P. Luong, “Broken rotor bar dataset information,” Oct 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3514322

[29] W. Jung, S.-H. Kim, S.-H. Yun, J. Bae, and Y.-H. Park, “Vibration, acoustic, temperature, and motor current dataset of rotating machine under varying operating conditions for fault diagnosis,” *Data in Brief*, vol. 48, p. 109049, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340923001671

[30] A. Elly Treml, R. Andrade Flauzino, M. Suetake, and N. A. Ravazzoli Maciejewski, “Experimental database for detecting and diagnosing rotor broken bar in a three-phase induction motor.” 2020. [Online]. Available: https://dx.doi.org/10.21227/fmnm-bn95

[31] W. H. Kruskal and W. A. W. and, “Use of ranks in one-criterion variance analysis,” *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441

[32] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The Annals of Mathematical Statistics*, pp. 50–60, 1947.

[33] M. Jafari, “Optimal redundant sensor configuration for accuracy increasing in space inertial navigation system,” *Aerospace Science and Technology*, vol. 47, pp. 467–472, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963815002758

[34] H. S. Kim, S. K. Park, Y. Kim, and C. G. Park, “Hybrid fault detection and isolation method for uav inertial sensor redundancy management system,” *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 265–270, 2005, 16th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147466701638017X

[35] N. Ahmed, A. A. Hashmani, S. Khokhar, M. A. Tunio, and M. Faheem, “Fault detection through discrete wavelet transform in overhead power transmission lines,” *Energy Science & Engineering*, vol. 11, pp. 4181–4197, 11 2023.

[36] A. U. Rehman, W. Jiao, J. Sun, M. Sohaib, Y. Jiang, M. Shahzadi, and M. I. Khan, “Efficient fault detection of rotor minor inter-turn short circuit in induction machines using wavelet transform and empirical mode decomposition,” *Sensors*, vol. 23, p. 7109, 8 2023.

[37] N. Ouerghemmi, S. B. Salem, M. Salah, K. Bacha, and A. Chaari, “V-belt fault detection using extended park vector approach (epva) in centrifugal fan driven by an induction motor,” *Mechanical Systems and Signal Processing*, vol. 200, p. 110566, 10 2023.

[38] C. Abdellah, C. Mama, M. R. M. Abderrahmane, and B. Mohammed, “Current park’s vector pattern technique for diagnosis of broken rotor bars fault in saturated induction motor,” *Journal of Electrical Engineering & Technology*, vol. 18, pp. 2749–2758, 7 2023.

[39] S. Zhang and Z. Q. Lang, “Scada-data-based wind turbine fault detection: A dynamic model sensor method,” *Control Engineering Practice*, vol. 102, p. 104546, 2020. [Online]. Available: https://doi.org/10.1016/j.conengprac.2020.104546

[40] A. Zimek and P. Filzmoser, “There and back again: Outlier detection between statistical reasoning and data mining algorithms,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, pp. 1–26, 2018.

[41] P. Gangsar and R. Tiwari, “Signal based condition monitoring techniques for fault detection and diagnosis of induction motors: A state-of-the-art review,” *Mechanical Systems and Signal Processing*, vol. 144, p. 106908, 2020. [Online]. Available: https://doi.org/10.1016/j.ymssp.2020.106908

[42] M. Paliwal and U. A. Kumar, “Neural networks and statistical techniques: A review of applications,” *Expert Systems with Applications*, vol. 36, pp. 2–17, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2007.10.005

[43] R. R. Shubita, A. S. Alsadeh, and I. M. Khater, "Fault detection in rotating machinery based on sound signal using edge machine learning," *IEEE Access*, vol. 11, pp. 6665–6672, 2023.

[44] M. Benninger, M. Liebschner, and C. Kreischer, "Fault detection of induction motors with combined modeling- and machine-learning-based framework," *Energies*, vol. 16, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/8/3429

[45] S. Voutsinas, D. Karolidis, I. Voyiatzis, and M. Samarakou, "Development of a machine-learning-based method for early fault detection in photovoltaic systems," *Journal of Engineering and Applied Science*, vol. 70, p. 27, 12 2023.

[46] A. K. Sahoo and S. K. Samal, "Online fault detection and classification of 3-phase long transmission line using machine learning model," *Multiscale and Multidisciplinary Modeling, Experiments and Design*, vol. 6, pp. 135–146, 3 2023.

[47] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 3127–3141, 2020.

[48] S. Lin, R. Clark, R. Birke, and S. Sch, "Anomaly detection for time series using vae-lstm hybrid model," *Ieee*, pp. 4322–4326, 2020.

[49] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head cnn–rnn for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 10 2019.

[50] R. Sabir, D. Rosato, S. Hartmann, and C. Gühmann, "Lstm based bearing fault diagnosis of electrical machines using motor current signal," *2019 18th IEEE International Conference On Machine Learning And Applications*, pp. 613–618, 2019.

[51] R. Wang, Z. Feng, S. Huang, X. Fang, and J. Wang, "Research on voltage waveform fault detection of miniature vibration motor based on improved wp-lstm," *MDPI Micro-Manufacturing and Applications*, 2020.

[52] A. Cano, P. Arévalo, D. Benavides, and F. Jurado, "Integrating discrete wavelet transform with neural networks and machine learning for fault detection in microgrids," *International Journal of Electrical Power & Energy Systems*, vol. 155, p. 109616, 1 2024.

[53] X. Luo, H. Wang, T. Han, and Y. Zhang, "Fft-trans: Enhancing robustness in mechanical fault diagnosis with fourier transform-based transformer under noisy conditions," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–12, 2024.

[54] A. K. Das, S. Das, A. K. Pradhan, B. Chatterjee, and S. Dalai, "Rpcnnet: A deep learning approach to sense minor stator winding interturn fault severity in induction motor under variable load condition," *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3965–3972, 2023.

[55] S. Fu, Y. Wu, R. Wang, and M. Mao, "A bearing fault diagnosis method based on wavelet denoising and machine learning," *Applied Sciences*, vol. 13, p. 5936, 5 2023.

[56] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and P. Shengyan, "Hyperspectral image classification with capsule network using limited training samples," *Sensors (Switzerland)*, vol. 18, 2018.

[57] A. Byerly, T. Kalganova, and I. Dear, "No routing needed between capsules," *Neurocomputing*, vol. 463, pp. 545–553, 11 2021.

[58] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using recurrent neural networks (rnn)," *ICNS 2016: Securing an Integrated CNS System to Meet Future Challenges*, pp. 1–8, 2016.

[59] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 6, pp. 107–116, 1998.

[60] J.-H. Lee, C. N. Okwuosa, and J.-W. Hur, "Extruder machine gear fault detection using autoencoder lstm via sensor fusion approach," *Inventions*, vol. 8, no. 6, 2023. [Online]. Available: https://www.mdpi.com/2411-5134/8/6/140

[61] I. Kumar, B. K. Tripathi, and A. Singh, "Attention-based lstm network-assisted time series forecasting models for petroleum production," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106440, Kumar. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623006243

[62] X. Wen and W. Li, "Time series prediction based on lstm-attention-lstm model," *IEEE Access*, vol. 11, pp. 48 322–48 331, 2023.

[63] A. Mohammad-Alikhani, B. Nahid-Mobarakeh, and M.-F. Hsieh, "One-dimensional lstm-regulated deep residual network for data-driven fault detection in electric machines," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 3, pp. 3083–3092, 2024.

[64] O. I. Provotar, Y. M. Linder, and M. M. Veres, "Unsupervised anomaly detection in time series using lstm-based autoencoders," *2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019 - Proceedings*, pp. 513–517, 2019.

[65] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," *2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014*, vol. 2014, pp. 844–848, 2014.

[66] M. Zhang, W. Li, and Q. Du, "Diverse region-based cnn for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 27, pp. 2623–2634, 2018.

[67] S. Mukhopadhyay and M. Litoiu, “Fault detection in sensors using single and multi-channel weighted convolutional neural networks,” *Proceedings of the 10th International Conference on the Internet of Things*, pp. 0–7, 2020.

[68] C. Y. Hsu and W. C. Liu, “Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing,” *Journal of Intelligent Manufacturing*, vol. 32, pp. 823–836, 2021. [Online]. Available: https://doi.org/10.1007/s10845-020-01591-0

[69] T. Y. Kim and S. B. Cho, “Web traffic anomaly detection using c-lstm neural networks,” *Expert Systems with Applications*, vol. 106, pp. 66–76, 9 2018.

[70] A. Borré, L. O. Seman, E. Camponogara, S. F. Stefenon, V. C. Mariani, and L. d. S. Coelho, “Machine fault detection using a hybrid cnn-lstm attention-based model,” *Sensors*, vol. 23, no. 9, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/9/4512

[71] X. Zhang and W. Zhou, “Structural vibration data anomaly detection based on multiple feature information using cnn-lstm model,” *Structural Control and Health Monitoring*, vol. 2023, no. 1, p. 3906180, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/3906180

[72] J. Kim, H. Kang, and P. Kang, “Time-series anomaly detection with stacked transformer representations and 1d convolutional network,” *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105964, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623001483

[73] P. Afshar, A. Mohammadi, and K. N. Plataniotis, “Brain tumor type classification via capsule networks,” *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3129–3133, 2018.

[74] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, and F. Pla, “Capsule networks for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 2145–2160, 2019.

[75] X. Ma, H. Zhong, Y. Li, J. Ma, Z. Cui, and Y. Wang, “Forecasting transportation network speed using deep capsule networks with nested lstm models,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4813–4824, 2021.

[76] R. Huang, J. Li, W. Li, and L. Cui, “Deep ensemble capsule network for intelligent compound fault diagnosis using multisensory data,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, 2020.

[77] S. R. R. Fahim, S. K. Sarker, S. M. Muyeen, M. R. I. Sheikh, S. K. Das, and M. G. Simoes, “A robust self-attentive capsule network for fault diagnosis of series-compensated transmission line,” *IEEE Transactions on Power Delivery*, vol. 8977, 2021.

[78] Z. Wang and T. Oates, “Encoding time series as images for visual inspection and classification using tiled convolutional neural networks,” *AAAI Workshop - Technical Report*, vol. WS-15-14, pp. 40–46, 2015.

[79] L. Shen, Z. Yu, Q. Ma, and J. T. Kwok, "Time series anomaly detection with multiresolution ensemble decoding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, pp. 9567–9575, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17152

[80] T. Kieu, B. Yang, C. Guo, R.-G. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, "Anomaly detection in time series with robust variational quasi-recurrent autoencoders," pp. 1342–1354, 2022.

[81] M. I. Radaideh, C. Pappas, J. Walden, D. Lu, L. Vidyaratne, T. Britton, K. Rajput, M. Schram, and S. Cousineau, "Time series anomaly detection in power electronics signals with recurrent and convlstm autoencoders," *Digital Signal Processing*, vol. 130, p. 103704, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200422003219

[82] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, p. 1, 1 2023.

[83] S. Yan, H. Shao, Y. Xiao, B. Liu, and J. Wan, "Hybrid robust convolutional autoencoder for unsupervised anomaly detection of machine tools under noises," *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102441, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584522001259

[84] S. Mishra and S. Jabin, "Anomaly detection in surveillance videos using deep autoencoder," *International Journal of Information Technology*, vol. 16, pp. 1111–1122, 2 2024.

[85] A.-T. W. Khalid Fahmi, K. Reza Kashyzadeh, and S. Ghorbani, "Fault detection in the gas turbine of the kirkuk power plant: An anomaly detection approach using dlstm-autoencoder," *Engineering Failure Analysis*, vol. 160, p. 108213, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1350630724002590

[86] T. Xie, Q. Xu, and C. Jiang, "Anomaly detection for multivariate times series through the multi-scale convolutional recurrent variational autoencoder," *Expert Systems with Applications*, vol. 231, p. 120725, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423012277

[87] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder-based anomaly detection for indoor air quality time-series data," *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3787–3800, 2023.

[88] N. Chen, H. Tu, X. Duan, L. Hu, and C. Guo, "Semisupervised anomaly detection of multivariate time series based on a variational autoencoder," *Applied Intelligence*, 7 2022.

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting,"

Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[90] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 1135–1143, 2015.

[91] X. Qian and D. Klabjan, "A probabilistic approach to neural network pruning," 2021. [Online]. Available: http://arxiv.org/abs/2105.10065

[92] S. Urolagin, P. K.V., and N. V. S. Reddy, "Generalization capability of artificial neural network incorporated with pruning method," P. S. Thilagam, A. R. Pais, K. Chandrasekaran, and N. Balakrishnan, Eds. Springer Berlin Heidelberg, 2012, pp. 171–178.

[93] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," 2020. [Online]. Available: http://arxiv.org/abs/2012.09243

[94] H. Drucker and Y. Le Cun, "Improving generalization performance using double backpropagation," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 991–997, 1992.

[95] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," 2004.

[96] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 8 1999.

[97] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 757–774, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157823000228

[98] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 8 1996.

[99] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890540185711364

[100] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[101] R. Hoque, S. Das, M. Hoque, and M. Hoque, "Breast cancer classification using xgboost," *World Journal of Advanced Research and Reviews*, vol. 21, pp. 1985–1994, 2 2024.

[102] M. K. Amal Asselman and S. Aammou, "Enhancing the prediction of student performance based on the machine learning xgboost algorithm," *Interactive Learning Environments*, vol. 31, no. 6, pp. 3360–3379, 2023. [Online]. Available: https://doi.org/10.1080/10494820.2021.1928235

[103] A. Dezhkam and M. T. Manzuri, "Forecasting stock market for an efficient portfolio by combining xgboost and hilbert–huang transform," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105626, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622006169

[104] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance," 2022.

[105] Z. C. Lipton, Y. X. Wang, and A. J. Smola, "Detecting and correcting for label shift with black box predictors," *35th International Conference on Machine Learning, ICML 2018*, vol. 7, pp. 4887–4897, 2018.

[106] S. Rabanser, S. Günnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[107] R. Wu, C. Guo, Y. Su, and K. Q. Weinberger, "Online adaptation to label distribution shift," pp. 1–20, 2021. [Online]. Available: http://arxiv.org/abs/2107.04520

[108] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, "A modern take on the bias-variance tradeoff in neural networks," 2019.

[109] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[110] M. A. Tanner and W. H. Wong, "The calculation of posterior distributions by data augmentation," *Journal of the American Statistical Association*, vol. 82, 1987.

[111] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, 2019.

[112] D. Amodei and D. Hernandez, "Ai and compute," May 2018. [Online]. Available: https://openai.com/research/ai-and-compute

[113] D. Yalalov, "Ai model training costs are expected to rise from $100 million to $500 million by 2030," Feb 2023. [Online]. Available: https://mpost.io/ai-model-training-costs-are-expected-to-rise-from-100-million-to-500-million-by-2030/

[114] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica (Slovenia)*, vol. 44, pp. 291–302, 2020.

[115] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, pp. 43–76, 1 2021.

[116] Y. Xian and H. Hu, "Enhanced multi-dataset transfer learning method for unsupervised person re-identification using co-training strategy," *IET Computer Vision*, vol. 12, pp. 1219–1227, 12 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2018.5103https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-cvi.2018.5103https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-cvi.2018.5103

[117] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 1623–1637, 3 2022.

[118] Y. Yao, Y. Wang, Y. Guo, J. Lin, H. Qin, and Y. Sensetime, "Cross-dataset training for class increasing object detection," 1 2020. [Online]. Available: https://arxiv.org/abs/2001.04621v1

[119] X. Zhou, V. Koltun, and P. Krähenbühl, "Simple multi-dataset detection," 2022.

[120] X. Jin, Y. Park, D. C. Maddix, H. Wang, and Y. Wang, "Domain adaptation for time series forecasting via attention sharing," 2022.

[121] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2016.

[122] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster r-cnn for object detection in the wild," 2018.

[123] J. Cheng, K. Zhang, and Y. Yang, "An order tracking technique for the gear fault diagnosis using local mean decomposition method," *Mechanism and Machine Theory*, vol. 55, pp. 67–76, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X12001000

[124] H. Li, Y. Zhang, and H. Zheng, "Bearing fault detection and diagnosis based on order tracking and teager-huang transform," *Journal of Mechanical Science and Technology*, vol. 24, pp. 811–822, 3 2010.

[125] A. Sapena-Bano, J. Burriel-Valencia, M. Pineda-Sanchez, R. Puche-Panadero, and M. Riera-Guasp, "The harmonic order tracking analysis method for the fault diagnosis in induction motors under time-varying conditions," *IEEE Transactions on Energy Conversion*, vol. 32, no. 1, pp. 244–256, 2017.

[126] M. Akar, "Detection of a static eccentricity fault in a closed loop driven induction motor by using the angular domain order tracking analysis method," *Mechanical Systems and Signal Processing*, vol. 34, no. 1, pp. 173–182, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S088832701200129X

[127] J. Wang, Y. Peng, and W. Qiao, "Current-aided order tracking of vibration signals for bearing fault diagnosis of direct-drive wind turbines," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 10, pp. 6336–6346, 2016.

[128] H. Vold, H. Herlufsen, M. L. Mains, and D. Corwin-Renner, "Multi axle order tracking with the vold-kalman tracking filter," *Sound and Vibration*, vol. 31, pp. 30–34, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:67831803

[129] M. van der Seijs, "Second generation vold-kalman order filtering," 2024, accessed: (June 22nd, 2024).

[130] M.-C. Pan and C.-X. Wu, "Adaptive vold–kalman filtering order tracking," *Mechanical Systems and Signal Processing*, vol. 21, no. 8, pp. 2957–2969, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327007001008

[131] S. Zhao, Q. Song, M. Wang, W. Lai, and Y. Li, "Mechanical faults detection for vehicle motors under nonstationary conditions based on vold-kalman order tracking method," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 237, no. 4, pp. 839–851, 2023. [Online]. Available: https://doi.org/10.1177/09544070221078679

[132] Z. Chen, G. He, J. Li, Y. Liao, K. Gryllias, and W. Li, "Domain adversarial transfer network for cross-domain fault diagnosis of rotary machinery," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 11, pp. 8702–8712, 2020.

[133] X. Li, W. Zhang, H. Ma, Z. Luo, and X. Li, "Partial transfer learning in machinery cross-domain fault diagnostics using class-weighted adversarial networks," *Neural Networks*, vol. 129, pp. 313–322, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360802030229X

[134] H. Zheng, Y. Yang, J. Yin, Y. Li, R. Wang, and M. Xu, "Deep domain generalization combining a priori diagnosis knowledge toward cross-domain fault diagnosis of rolling bearing," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[135] D. Xiao, Y. Huang, L. Zhao, C. Qin, H. Shi, and C. Liu, "Domain adaptive motor fault diagnosis using deep transfer learning," *IEEE Access*, vol. 7, pp. 80 937–80 949, 2019.

[136] S. Altaf, S. Ahmad, M. Zaindin, S. Huda, S. Iqbal, and M. W. Soomro, "Multiple industrial induction motors fault diagnosis model within powerline system based on wireless sensor network," *Sustainability*, vol. 14, no. 16, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/16/10079

[137] Z.-H. Liu, B.-L. Lu, H.-L. Wei, L. Chen, X.-H. Li, and M. Rätsch, "Deep adversarial domain adaptation model for bearing fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 7, pp. 4217–4226, 2021.

[138] K.-C. Chao, C.-B. Chou, and C.-H. Lee, "Online domain adaptation for rolling bearings fault diagnosis with imbalanced cross-domain data," *Sensors*, vol. 22, no. 12, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/12/4540

[139] X. Zhang, J. Tang, Y. Qu, G. Qin, L. Guo, J. Xie, and Z. Long, "Few-shot fault diagnosis based on heterogeneous information fusion and meta learning," *IEEE Sensors Journal*, vol. 23, no. 18, pp. 21 433–21 442, 2023.

[140] E. Dubrova, *Hardware Redundancy.* Springer New York, 2013. [Online]. Available: https://doi.org/10.1007/978-1-4614-2113-9{_}4

[141] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Artificial Neural Networks and Machine Learning – ICANN 2011*, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 44–51.

[142] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2227-7080/9/3/52

[143] C. Van Rijsbergen and C. Van Rijsbergen, *Information Retrieval.* Butterworths, 1979. [Online]. Available: https://books.google.co.uk/books?id=t-pTAAAAMAAJ

[144] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[145] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark," *CoRR*, vol. abs/1510.03336, 2015. [Online]. Available: http://arxiv.org/abs/1510.03336

[146] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *ArXiv*, 4 2019. [Online]. Available: http://arxiv.org/abs/1904.09237

[147] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pp. 1409–1416, 2019.

[148] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *ArXiv*, pp. 1–8, 2016. [Online]. Available: http://arxiv.org/abs/1612.06676

[149] F. Chollet, "Building autoencoders in keras," *The Keras Blog*, 2016. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html

[150] K. Gross, R. Singer, S. Wegerich, J. Herzog, R. VanAlstine, and F. Bockhorst, "Application of a model-based fault detection system to nuclear plant signals," *International conference on intelligent systems applications to power systems*, p. 6, 1997. [Online]. Available: http://www.osti.gov/bridge/product.biblio.jsp?osti_id=481606

[151] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," *Icdm*, 2008. [Online]. Available: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf%0Ahttps://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest

[152] P. Vijay, "Timeseries anomaly detection using an autoencoder," *Keras*, 2020. [Online]. Available: https://keras.io/examples/timeseries/timeseries_anomaly_detection/

[153] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, pp. 10–21, 2016.

[154] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," *Proceedings of the 17th SIAM International Conference on Data Mining, SDM 2017*, pp. 90–98, 2017.

[155] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Comput. Surv.*, vol. 54, no. 3, apr 2021. [Online]. Available: https://doi.org/10.1145/3444690

[156] G. Jin, X. Yi, L. Zhang, L. Zhang, S. Schewe, and X. Huang, "How does weight correlation affect the generalisation ability of deep neural networks?" vol. 2020-Decem, 2020.

[157] D. Partridge and N. Griffith, "Strategies for improving neural net generalisation," *Neural Computing & Applications*, vol. 3, pp. 27–37, 1995.

[158] S. Lawrence and C. L. Giles, "Overfitting and neural networks: Conjugate gradient and backpropagation," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 114–119, 2000.

[159] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in Neural Information Processing Systems*, 2001.

[160] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Computation*, vol. 8, pp. 643–674, 4 1996. [Online]. Available: https://direct.mit.edu/neco/article/8/3/643/5975/The-Effects-of-Adding-Noise-During-Backpropagation

[161] J. Chen, X. Feng, L. Jiang, and Q. Zhu, "State of charge estimation of lithium-ion battery using denoising autoencoder and gated recurrent unit recurrent neural network," *Energy*, vol. 227, p. 120451, 7 2021.

[162] G. Salimi-Khorshidi, G. Douaud, C. F. Beckmann, M. F. Glasser, L. Griffanti, and S. M. Smith, "Automatic denoising of functional mri data: Combining independent component analysis and hierarchical fusion of classifiers," *NeuroImage*, vol. 90, pp. 449–468, 4 2014.

[163] A. Byerly and T. Kalganova, "Towards an analytical definition of sufficient data," 2022. [Online]. Available: http://arxiv.org/abs/2202.03238

[164] ——, "Class density and dataset quality in high-dimensional, unstructured data," 2022. [Online]. Available: http://arxiv.org/abs/2202.03856

[165] H. Raza, G. Prasad, and Y. Li, "Dataset shift detection in non-stationary environments using ewma charts," *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 3151–3156, 2013.

[166] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*, pp. 816–825, 12 2016.

[167] H. Raza, G. Prasad, and Y. Li, "Adaptive learning with covariate shift-detection for non-stationary environments," *2014 14th UK Workshop on Computational Intelligence, UKCI 2014 - Proceedings*, 10 2014.

[168] Z. Cai, O. Sener, and V. Koltun, "Online continual learning with natural distribution shifts: An empirical study with visual data," 2021. [Online]. Available: https://github.com/http://arxiv.org/abs/2108.09020

[169] S. Ahmad, K. Styp-Rekowski, S. Nedelkoski, and O. Kao, "Autoencoder-based condition monitoring and anomaly detection method for rotating machines," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 4093–4102.

[170] B. Maschler, T. Knodel, and M. Weyrich, "Towards deep industrial transfer learning for anomaly detection on time series data," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 01–08.

[171] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[172] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[173] N. Chinchor, "Muc-4 evaluation metrics," in *Proceedings of the 4th Conference on Message Understanding*, ser. MUC4 '92. USA: Association for Computational Linguistics, 1992, p. 22–29. [Online]. Available: https://doi.org/10.3115/1072064.1072067

[174] J. Sevilla, L. Heim, M. Hobbhahn, T. Besiroglu, A. Ho, and P. Villalobos, “Estimating training compute of deep learning models,” 2022, accessed: 2023-4-23. [Online]. Available: https://epochai.org/blog/estimating-training-compute

[175] P. Welch, “The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms,” *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[176] E. Rajaby and S. M. Sayedi, “A structured review of sparse fast fourier transform algorithms,” *Digital Signal Processing*, vol. 123, p. 103403, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200422000203

[177] Z. Xu, C. Dan, J. Khim, and P. Ravikumar, “Class-weighted classification: Trade-offs and robust approaches,” 2020.