UNIVERSITY OF CALIFORNIA SAN DIEGO
SAN DIEGO STATE UNIVERSITY

Towards Intelligent, Secure, and Efficient Industrial Internet of Things

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Engineering Science (Electrical & Computer Engineering)

by

Onat Gungor

Committee in charge:

University of California San Diego
- Professor Tajana Šimunić Rosing, Co-Chair
- Professor Ryan Kastner
- Professor Piya Pal

San Diego State University
- Professor Baris Aksanli, Co-Chair
- Professor Shangping Ren

2023

The dissertation of Onat Gungor is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

Co-Chair

Co-Chair

University of California San Diego

San Diego State University

2023

DEDICATION

To my family

For their endless support and encouragement to go on and complete this journey

TABLE OF CONTENTS

Dissertation Approval Page iii

Dedication iv

Table of Contents v

List of Figures viii

List of Tables x

Acknowledgements xi

Vita xiii

Abstract of the Dissertation xv

Chapter 1 Introduction 1
1.1 Intelligent Learning 3
1.2 Secure Learning 4
1.3 Efficient Learning 7
1.4 Thesis Contributions 9

Chapter 2 Diversity-induced Optimally Weighted Ensemble Learner 12
2.1 Introduction 12
2.2 Related Work 14
2.3 Proposed Framework 15
2.3.1 Data pre-processing Module 16
2.3.2 Deep Learning Module 16
2.3.3 Ensemble Module 18
2.4 Experimental Analysis 22
2.4.1 Experimental Setup 22
2.4.2 Data pre-processing 24
2.4.3 Deep Learning Models Performance 25
2.4.4 Ensemble Learner Performance 27
2.4.5 Analysis on Retraining 30
2.5 Conclusion 33
2.6 Acknowledgements 33

Chapter 3 Ensemble Few-Shot Learning under Limited Labeled Data 34
3.1 Introduction 34
3.2 Related Work 35
3.3 Proposed Framework 36

3.3.1 Siamese Neural Network . . . 36
3.3.2 Majority Voting Classifier . . . 39
3.4 Experimental Analysis . . . 40
3.4.1 Dataset Description . . . 40
3.4.2 Experimental Setup . . . 40
3.4.3 Results . . . 41
3.5 Conclusion . . . 43
3.6 Acknowledgements . . . 43

Chapter 4 Resilient Stacking Ensemble Learner Against Adversarial Attacks . . . 45
4.1 Introduction . . . 45
4.2 Related Work . . . 46
4.2.1 IIoT Adversarial Attack Workflow . . . 46
4.2.2 Adversarial Attack Formulation for RUL prediction . . . 47
4.2.3 Adversarial Attacks in Predictive Maintenance . . . 50
4.2.4 Ensemble Methods . . . 51
4.3 Proposed Framework . . . 52
4.3.1 Deep Learning Methods Compromise Calculation . . . 52
4.3.2 Stacking Ensemble Learner . . . 53
4.4 Experimental Analysis . . . 59
4.4.1 Dataset Description . . . 59
4.4.2 Experimental Setup . . . 60
4.4.3 Single DL Models Resiliency . . . 61
4.4.4 Proposed Stacking Ensemble Learner Resiliency . . . 61
4.5 Conclusion . . . 66
4.6 Acknowledgements . . . 67

Chapter 5 Diversity Promoting Ensemble
Adversarial Training . . . 68
5.1 Introduction . . . 68
5.2 Related Work . . . 69
5.3 Proposed Framework . . . 70
5.3.1 Pre-trained Models . . . 70
5.3.2 Loss Gradient Similarity Calculation . . . 71
5.3.3 Augmented Training Data Generation and Training . . . 71
5.3.4 Testing Framework . . . 72
5.3.5 Compared Training Settings . . . 72
5.4 Experimental Analysis . . . 73
5.4.1 Dataset Description . . . 73
5.4.2 Experimental Setup . . . 73
5.4.3 Impact of Number of Base Learners in Resiliency . . . 73
5.4.4 Mean Compromise Comparison . . . 74
5.5 Conclusion . . . 75
5.6 Acknowledgements . . . 76

Chapter 6 Hyperdimensional Computing for Resilient IIoT Predictive Analytics . . . 77
6.1 Introduction . . . 77
6.2 Background and Related Work . . . 78
6.2.1 HDC Background . . . 78
6.3 Proposed Framework . . . 81
6.4 Experimental Analysis . . . 83
6.4.1 Dataset Description . . . 83
6.4.2 Experimental Setup . . . 83
6.4.3 Resiliency Analysis . . . 84
6.5 Conclusion . . . 88
6.6 Acknowledgements . . . 88

Chapter 7 Hyperdimensional Computing Novel Adversarial Attack Design . . . 89
7.1 Introduction . . . 89
7.2 Related Work . . . 91
7.3 Adversarial Attack Design Framework . . . 92
7.3.1 Perturbation Creation . . . 93
7.3.2 Diversity Inclusion . . . 95
7.3.3 Real-time Attack Selection . . . 95
7.4 Experimental Analysis . . . 97
7.4.1 Dataset Description . . . 97
7.4.2 Experimental Setup . . . 97
7.4.3 Experimental Results . . . 98
7.5 Conclusion . . . 101
7.6 Acknowledgements . . . 101

Chapter 8 Summary and Future Work . . . 102
8.1 Thesis Summary . . . 102
8.1.1 Diversity-induced Optimally Weighted Ensemble Learner . . . 103
8.1.2 Ensemble Few-Shot Learning under Limited Labeled Data . . . 103
8.1.3 Resilient Stacking Ensemble Learner Against Adversarial Attacks . . . 103
8.1.4 Diversity Promoting Ensemble Adversarial Training . . . 104
8.1.5 Hyperdimensional Computing for Resilient IIoT Predictive Analytics . . . 104
8.1.6 Hyperdimensional Computing Novel Adversarial Attack Design . . . 104
8.2 Future Work . . . 105
8.2.1 Intelligent Learning . . . 105
8.2.2 Secure Learning . . . 105
8.2.3 Efficient Learning . . . 106

Bibliography . . . 107

LIST OF FIGURES

Figure 1.1. ML-enabled IIoT Architecture 2

Figure 1.2. ML prediction performance under adversarial attack 5

Figure 2.1. Proposed Framework for RUL Prediction 16

Figure 2.2. Dot Product Operation 20

Figure 2.3. Aircraft Engine Diagram 23

Figure 2.4. Random Forest Feature Importance 24

Figure 2.5. Deep Learning Models Prediction Performance 25

Figure 2.6. Base Learner Optimal Weights 26

Figure 2.7. State-of-the-art Ensemble Comparison 28

Figure 2.8. *OPTDIV* Performance for Different υ 30

Figure 2.9. Effect of δ on υ 31

Figure 2.10. RMSE Comparison 32

Figure 2.11. Normalized Retraining Time 32

Figure 3.1. Our Proposed Framework 36

Figure 3.2. Siamese Neural Network (SNN) Structure 37

Figure 3.3. Majority Voting Classifier 39

Figure 3.4. Transfer Learning Methods Comparison 41

Figure 4.1. IIoT Adversarial Attack Workflow 46

Figure 4.2. Multivariate time-series data 47

Figure 4.3. Framework for DL Methods Compromise Calculation 52

Figure 4.4. Framework for Stacking Ensemble Training 54

Figure 4.5. Framework for Stacking Ensemble Testing 55

Figure 4.6. Most Resilient Stacking Ensemble Selection 57

Figure 4.7. Calculate Compromise 58
Figure 4.8. Meta-learner Compromise Analysis 62
Figure 4.9. Stacking Ensemble Compromise Analysis 63
Figure 4.10. Adversarial Attacks Compromise Analysis 65
Figure 5.1. Our Proposed Framework 70
Figure 5.2. Testing Framework 72
Figure 5.3. Impact of Number of Base Learners in Resiliency 74
Figure 6.1. HDC Learning Framework 78
Figure 6.2. Black-box Attack Framework 81
Figure 6.3. CWRU Experimental Test Apparatus 83
Figure 6.4. Mean Compromise Analysis (Drive End Dataset) 85
Figure 6.5. Mean Compromise Analysis (Fan End Dataset) 85
Figure 7.1. Our Proposed Attack Design Framework 91
Figure 7.2. Perturbation Framework 93
Figure 7.3. Diversity Calculation Framework 94
Figure 7.4. Attack Selection Framework 96
Figure 7.5. Attack Performance Comparison ($\varepsilon = 0.1$) 98
Figure 7.6. Our Attack Design Performance 99

LIST OF TABLES

Table 2.1. Selected Deep Learning Models 17

Table 2.2. C-MAPSS Data Set Summary 23

Table 3.1. Single Method Fault Type Predictions 38

Table 3.2. Our Method's Improvement over the Best Algorithm (%) 42

Table 4.1. Single DL Models Mean Compromise 61

Table 4.2. Most Resilient Stacking Ensemble Configuration 66

Table 4.3. Proposed Stacking Ensemble Compromise Improvement Over the Most Resilient DL Method (%) 66

Table 5.1. Mean Compromise Comparison 75

Table 5.2. *DENSE-DEFENSE* Resiliency Improvement 75

Table 6.1. Average Compromise Comparison 86

Table 6.2. Average and Maximum Resiliency Improvement of HD over DL Methods . 86

Table 6.3. Target Models Training Time Comparison 87

Table 7.1. Improvement over the best single attack (FGSM) 100

Table 7.2. Attack Selection Overhead 100

ACKNOWLEDGEMENTS

I would like to first thank my advisors, Professor Tajana Rosing and Professor Baris Aksanli for their guidance and support during my Ph.D. Prof. Rosing's enthusiasm to explore a wide range of research directions pushed me to never leave a stone unturned. I'd like to thank Prof. Aksanli for his support in all aspects of life throughout the years, giving me the motivation I needed to continue on and for bringing new perspectives through his experience to my research. I would also like to thank my dissertation committee members, Professors Shangping Ren, Piya Pal, and Ryan Kastner, for their feedback and discussions related to my Ph.D. work. I would like to thank all my colleagues in SEELab for their active collaboration and continuous support. My research was made possible by funding from the National Science Foundation (NSF) Grant 1527034, 1730158, 1826967, 1830331, 1911095, 2003279.

I owe so much to all my family for their understanding, patience, and encouragement that helped me navigate through the difficult moments during the past few years.

The material in this dissertation is based on the following publications.

Chapter 2, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "DOWELL: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices". *IEEE Internet of Things Journal*, 2021. The dissertation author was the primary investigator and author of this material.

Chapter 3, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "ENFES: Ensemble Few-Shot Learning For Intelligent Fault Diagnosis with Limited Data". *IEEE SENSORS*, 2021. The dissertation author was the primary investigator and author of this material.

Chapter 4, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "STEWART: stacking ensemble for white-box adversarial attacks towards more resilient data-driven predictive maintenance". *Computers in Industry*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 5, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and

B. Aksanli, “DENSE-DEFENSE: Diversity Promoting Ensemble Adversarial Training Towards Effective Defense”. *IEEE SENSORS*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 6, in part, is a reprint of material as it appears in O. Gungor, T. Rosing, and B. Aksanli, “HD-I-IoT: Hyperdimensional Computing for Resilient Industrial Internet of Things Analytics”. IEEE/ACM Design, Automation and Test in Europe Conference and Exhibition (DATE), 2023. The dissertation author was the primary investigator and author of this material.

Chapter 7, in part, is a reprint of material as it appears in O. Gungor, T. Rosing, and B. Aksanli, “Adversarial-HD: Hyperdimensional Computing Adversarial Attack Design for Secure Industrial Internet of Things”. Cyber-Physical Systems and Internet of Things Week, 2023. The dissertation author was the primary investigator and author of this material.

My co-authors (Prof. Baris Aksanli, Tajana Rosing, listed in alphabetical order) have all kindly approved the inclusion of the aforementioned publications in my dissertation.

VITA

2018	Bachelor of Science in Industrial Engineering, Ozyegin University, Istanbul, Turkey
2019	Bachelor of Science in Computer Science, Ozyegin University, Istanbul, Turkey
2019	Teaching Assistant, San Diego State University, CA, USA
2020	Research Scholar, San Diego State University Research Foundation, CA, USA
2021	Course Instructor, University of California, San Diego, CA, USA
2022	Research Fellow, San Diego State University, CA, USA
2023	Graduate Student Researcher, Energy Sciences Network (ESnet), Lawrence Berkeley National Laboratory (LBL), Berkeley, CA, USA
2023	Doctor of Philosophy in Engineering Science (Electrical & Computer Engineering), University of California, San Diego, USA and San Diego State University, USA

PUBLICATIONS

Xiaofan Yu, Minxuan Zhou, Fatemeh Asgarinejad, Onat Gungor, Baris Aksanli, Tajana Rosing. "Lightning Talk: Private and Secure Edge AI with Hyperdimensional Computing", *Design Automation Conference (DAC)*, 2023.

Mitchell Timken, Onat Gungor, Tajana Rosing, Baris Aksanli. "Analysis of Machine Learning Algorithms for Cyber Attack Detection in SCADA Power Systems", *IEEE International Conference on Smart Communications and Networking (SmartNets)*, 2023.

Onat Gungor, Tajana Rosing, Baris Aksanli. "Adversarial-HD: Hyperdimensional Computing Adversarial Attack Design for Secure Industrial Internet of Things", *Cyber-Physical Systems and Internet of Things Week*, 2023.

Onat Gungor, Tajana Rosing, Baris Aksanli. "HD-I-IoT: Hyperdimensional Computing for Resilient Industrial Internet of Things Analytics", *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

Onat Gungor, Tajana Rosing, Baris Aksanli. "DODEM: Double Defense Mechanism Against Adversarial Attacks Towards Secure Industrial Internet of Things Analytics", *Arxiv*, 2023.

Onat Gungor, Tajana Rosing, Baris Aksanli. "STEWART: Stacking Ensemble for White-Box Adversarial Attacks Towards more resilient data-driven predictive maintenance", *Computers in Industry*, 2022.

Onat Gungor, Tajana Rosing, Baris Aksanli. "RES-HD: Resilient Intelligent Fault Diagnosis Against Adversarial Attacks Using Hyper-Dimensional Computing", *Arxiv*, 2022.

Onat Gungor, Tajana Rosing, Baris Aksanli. "DENSE-DEFENSE: Diversity Promoting Ensemble Adversarial Training Towards Effective Defense", *IEEE Sensors*, 2022.

Onat Gungor, Tajana Rosing, Baris Aksanli. "DOWELL: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices", *IEEE Internet of Things Journal*, 2021.

Onat Gungor, Tajana Rosing, Baris Aksanli. "Respire++: Robust indoor sensor placement optimization under distance uncertainty", *IEEE Sensors Journal*, 2021.

Onat Gungor, Tajana Rosing, Baris Aksanli. "CAHEROS: Constraint-Aware HEuristic Approach for RObust Sensor Placement", *IEEE Sensors*, 2021.

Onat Gungor, Tajana Rosing, Baris Aksanli. "ENFES: ENsemble FEw-Shot Learning For Intelligent Fault Diagnosis with Limited Data", *IEEE Sensors*, 2021.

Onat Gungor, Tajana Rosing, Baris Aksanli. "OPELRUL: OPtimally Weighted Ensemble Learner for Remaining Useful Life Prediction", *IEEE International Conference on Prognostics and Health Management (PHM)*, 2021.

Onat Gungor, Jake Garnier, Tajana Rosing, Baris Aksanli. "LENARD: Lightweight ENsemble LeARner for MeDium-term Electricity Consumption Prediction", *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2020.

Onat Gungor, Tajana Rosing, Baris Aksanli. "RESPIRE: Robust Sensor Placement Optimization in Probabilistic Environments", *IEEE Sensors*, 2020.

Onat Gungor, Baris Aksanli, Reyhan Aydogan "Algorithm selection and combining multiple learners for residential energy prediction", *Future Generation Computer Systems*, 2019.

Onat Gungor, Umut Cakan, Reyhan Aydogan, Pinar Ozturk. "Effect of awareness of other side's gain on negotiation outcome, emotion, argument, and bidding behavior", *International Workshop on Agent-Based Complex Automated Negotiation*, 2019.

ABSTRACT OF THE DISSERTATION

Towards Intelligent, Secure, and Efficient Industrial Internet of Things

by

Onat Gungor

Doctor of Philosophy in Engineering Science (Electrical & Computer Engineering)

University of California San Diego, 2023
San Diego State University, 2023

Professor Tajana Šimunić Rosing, Co-Chair
Professor Baris Aksanli, Co-Chair

Industrial Internet of Things (IIoT) is an adaptation of traditional IoT for industrial environments enabling full automation, remote monitoring, and smart maintenance. Predictive analytics aims to utilize the collected system data and provide meaningful insight into business decisions using Machine Learning (ML). However, there are numerous challenges that should be addressed to fully benefit from IIoT predictive analytics. These include performance, securtiy, and efficiency of ML algorithms. This dissertation provides solutions to each of these challenges.

Dynamic IIoT settings and conditions can significantly impact individual ML prediction performance. As a solution to this problem, ensemble learning systematically combines multiple

ML methods to increase prediction performance and robustness. However, in order to deploy ensemble learning solutions in IIoT systems, additional training overhead and learning ability under limited supervision should be addressed. To address the first challenge, we propose a diversity-induced optimally-weighted ensemble learner. Our solution provides 39.2% faster retraining compared to only accuracy included ensemble with 3.4% loss in accuracy. To solve the second problem, we devise a novel few shot ensemble learning framework. It results in up to 16.4% improvement over the best algorithm by only using 0.3% of the training data.

IIoT security is challenging due to its increased inter-connectivity, small scale devices, and large attack surface. Among various cyberattacks, adversarial attacks craft perturbed examples to affect ML prediction performance. These attacks can cause serious outcomes on IIoT systems, yet they are not carefully addressed in the IIoT domain. To fill this research gap, we develop (1) a stacking ensemble learning-based framework that stays resilient against various adversarial attacks, and (2) diversity promoting ensemble adversarial training approach as a defense mechanism. Our stacking ensemble improves resiliency against adversarial attacks by up to 60% compared to the most resilient single ML method. Furthermore, our defense improves the resiliency by up to 97% compared to state-of-the-art training settings.

IIoT systems need efficient and robust learning solutions due to their resource constrained devices and the potential for noise and variability. Hyper-dimensional computing (HDC) is a brain-inspired learning solution which is shown to be efficient, accurate, and robust. We first apply HDC for predictive analytics solution and test it against various adversarial attacks. We observe that HDC has up to 67.5% higher resiliency compared to the state-of-the-art deep learning methods while being up to 25.1$\times$ faster to train. Although we showed that HDC is more resilient than DL methods, there is still a need to further investigate its resiliency against adversarial attacks to use HDC for predictive analytics safely. For that purpose, we design a novel HDC adversarial attack. Our approach improves attack success rate by up to 36%, and F1 score by up to 61% compared to the most effective state-of-the-art single adversarial attack.

Chapter 1

Introduction

Industry 4.0 or fourth industrial revolution is an important milestone in production systems which aims to automate decision-making, interconnect machinery, and maximize productivity [1]. This leverages the notion of Industrial Internet of Things (IIoT) focusing on machine-to-machine communication, big data, and machine learning. IIoT is used for various tasks such as remote monitoring, business intelligence, quality control, automation, and predictive maintenance [2]. Its value is increasing, where IIoT market is estimated to worth $7.1 trillion to the United States by 2030 [3]. Huge amounts of data are collected in IIoT environments on a regular basis by connecting every machine to the Internet [4]. Processing and analyzing these data plays a significant role in obtaining more effective IIoT systems. Predictive analytics aims to deploy Machine Learning (ML) solutions in IIoT systems to make predictions about the likelihood of future outcomes, e.g., machinery failure, maintenance schedule.

ML-based solutions can be impactful in high-level decision making, bringing significant cost gains. To illustrate, one robotic manufacturing company reduced downtime by 50% and increased performance by 25% by utilizing an ML algorithm for failure prediction [5]. It is crucial to understand how ML can be integrated into IIoT systems. Figure 1.1 presents the ML-enabled IIoT architecture which consists of four main layers: physical, edge, cloud, and visualization [6]. The physical layer consists of IIoT assets such as machinery, sensors, and actuators. The edge layer collects data from physical layer, pre-processes the collected data,

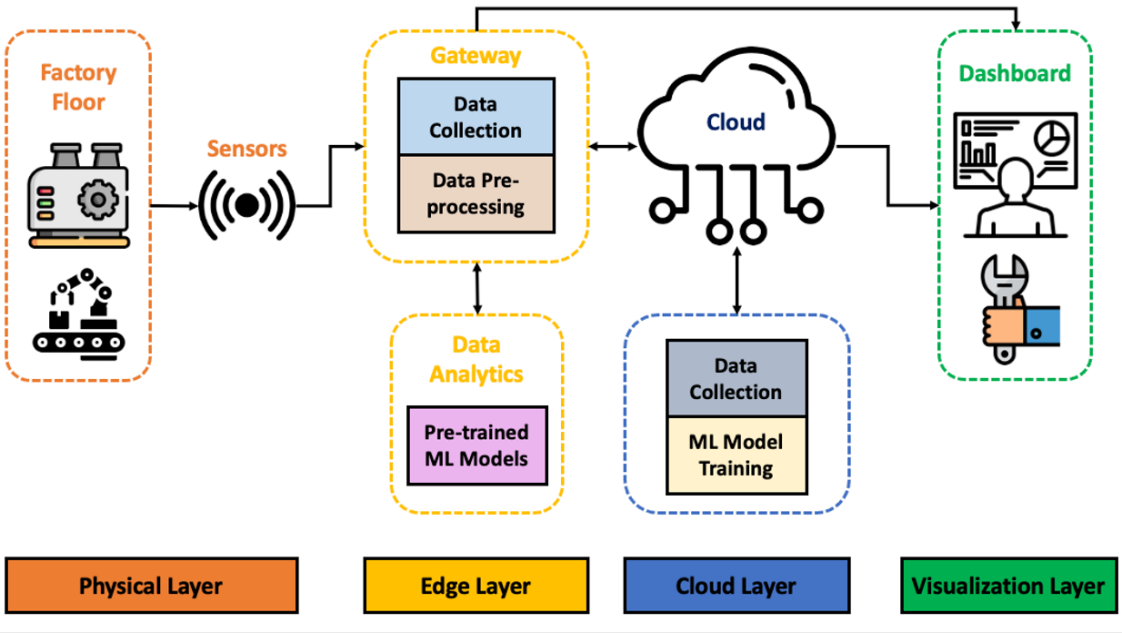


Figure 1.1. ML-enabled IIoT Architecture

establishes the cloud connection, and performs data analytics via pre-trained ML methods. The cloud layer is responsible for storing the collected system data, and training ML models. It is important to note that the trained ML models might require retraining when new data arrive and those models are sent back to the edge layer to keep the prediction performance at a certain level. Finally, the visualization layer utilizes data from both edge and cloud layers and provides a visual representation of actionable insights, e.g., maintenance schedule, failure prediction.

This dissertation focuses on two main predictive analytics applications: predictive maintenance and intrusion detection. Predictive maintenance (PdM) finds an optimum time to schedule a maintenance before any failure occurs [7]. Its market size is expected to grow from $4.0 billion in 2020 to $12.3 billion by 2025 [8]. Data-driven PdM utilizes historical data and deploy ML models in IIoT environments. Data-driven PdM has multiple applications such as remaining useful life (RUL) estimation, and intelligent fault diagnosis (IFD) [9]. RUL is defined as the remaining time of a machine to perform its functions until it fails [9]. RUL prediction is a crucial PdM task where even the slightest improvement in the RUL prediction performance can cause huge cost gains via increased production efficiency [7]. Due to abundance of available IIoT data,

ML-based RUL prediction methods have become extremely popular. These methods aim to find a mapping between historical data and corresponding RUL values. As another important PdM application, IFD utilizes ML algorithms to detect and classify different fault types. Other than predictive maintenance, intrusion detection aims to continuously monitor the IIoT network data to detect incoming cyberattacks [10]. It provides timely detection and alerts to prevent the spread of infection throughout the IIoT system. ML methods have been recently adopted for intrusion detection due to their successful performance in detecting cyberattacks effectively.

In order to fully benefit from IIoT predictive analytics, there are three main challenges that should be addressed. The first challenge is the difficulty of selecting a single ML method that can perform well across different IIoT settings. Ensemble learning aims to solve this problem by combining multiple learners, yet it also brings additional training overhead. Another challenge is IIoT security due to long lifetime of industrial devices, large-scale networks, and increased interconnectivity. Specifically, ML security plays a critical role in IIoT systems since ML methods are heavily used in high level decision making. Adversarial attacks create slight but carefully-crafted examples to affect the ML model prediction performance. These attacks can lead to serious outcomes for IIoT systems such as undetected failures. The last challenge is creating efficient ML solutions to be used at the edge, in resource-constrained devices. Hyperdimensional computing (HDC) is one possible solution approach for robust and efficient learning. However, HDC can also be vulnerable to adversarial attacks. To cope up with these challenges, this dissertation proposes intelligent, secure, and efficient IIoT predictive analytics solutions. Next, we go into the details of these challenges, provide a background knowledge, and outline our solutions.

1.1 Intelligent Learning

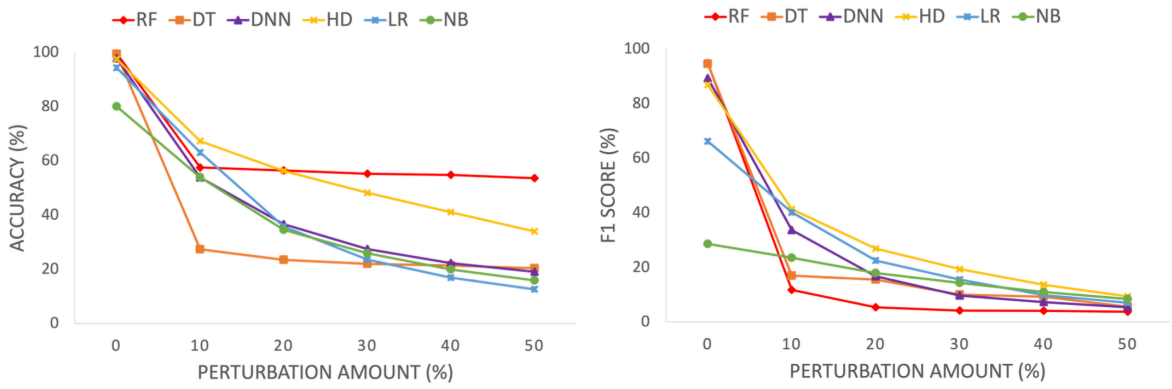
Although there are numerous ML methods proposed for different data-driven PdM applications, it is difficult to deploy a single ML method that works well across various IIoT settings. Since industrial systems include very large number of individual systems and components, single

ML model predictive performance might change drastically. This implies that deploying single ML method might cause misleading actions for IIoT systems, e.g., wrong maintenance decision. We can obtain more accurate and robust predictions by combining different ML methods. **Ensemble learning (EL)** is one solution approach to this problem which combines various ML algorithms, i.e., base learners, in a systematic manner. EL provides higher a generalization ability, improves robustness, and reduces the likelihood of a wrong selection of a poor individual learner. This brings more successful predictive analytics, increasing system robustness and decreasing maintenance costs of IIoT systems. There are three main ensemble learning approaches in the literature [11]: (i) bagging, (ii) boosting, and (iii) stacking. Bagging combines similar type of learners by using different subsamples of the data, e.g., random forest. Boosting also combines similar learners while fixing the prediction error iteratively, e.g., AdaBoost. Lastly, stacking combines different learners via a second-level learner, i.e., a meta learner.

In order for ensemble learning to be used in IIoT predictive analytics, there are two main challenges that need to be addressed: (i) additional training overhead, and (ii) learning under limited supervision. Additional training overhead is an important issue in ensemble learning where multiple models need to be trained initially and might require retraining under new data to maintain the overall prediction performance. To address this challenge, we propose a diversity-induced optimally-weighted ensemble learner. For IIoT systems, it might be infeasible to have labeled data for different working conditions. This can impact the prediction performance of ensemble learning solutions. To solve this problem, we propose ensemble few-shot learning.

1.2 Secure Learning

IIoT Security: IIoT systems are often designed without security in mind or use communication protocols that are not sufficiently secure and vulnerable off-the-shelf commercial products [12, 13]. Increased inter-connectivity, small scale devices, large attack surface, and poorly implemented security features make IIoT an easy target for cybercriminals [14]. Wu



(a) **Accuracy** (b) F_1 **Score**

Figure 1.2. ML prediction performance under adversarial attack

et al. [13] summarize the IIoT assets that are vulnerable to cyberattacks under 4 categories: operating systems, application software, communication protocols, and smart devices. Sophisticated attackers can easily gain access to an entire IIoT system and damage its functionality and production for a lengthy period [15]. For instance, the average estimated losses were $10.7 million per breach of data among manufacturing organizations in Asia Pacific in 2019 [16]. To cope with this challenge, cyber security measures should be taken such as cyber-security awareness training, keeping software programs up-to-date, and installing a firewall [13]. Tuptuk and Hailes [12] summarize common IIoT attacks under 13 different classes: denial of service, eavesdropping, man-in-the-middle, false data injection, time delay, data tampering, replay, spoofing, side channel, covert channel, zero day, physical, and adversarial attacks.

IIoT Adversarial Attack Motivation: Adversarial attacks aim to create slight but carefully-crafted examples to impact ML model prediction performance. These attacks became extremely important since ML methods have great success in IIoT [17]. Adversarial attacks can cause serious consequences on IIoT systems such as undetected failures [6]. Figure 1.2 demonstrates the impact of an adversarial attack (under varying perturbation amounts) on ML prediction performance for IIoT intrusion detection. Here, 0% perturbation refers to no adversarial attack. Higher perturbation amount implies a stronger adversarial attack. In this

figure, different colors denote distinct ML methods, e.g., random forest (RF), hyperdimensional computing (HD), deep neural network (DNN). Under an adversarial attack, accuracy and F_1 score can be impacted significantly irrespective of the underlying ML model, causing up to 25$\times$ performance loss.

Adversarial Attacks: There are three types of adversarial attacks in the literature: evasion, poisoning, and exploratory [18]. Evasion attacks target compromising the test data, poisoning attacks contaminate the training data, and exploratory attacks gain knowledge about the learning algorithm. We can further divide evasion attacks into two groups: white-box and black-box. While white-box attacks have detailed knowledge about the model, black-box attacks assume no knowledge about the underlying model. Other than attack generation methods, an adversary also needs to determine how to conduct an adversarial attack for a real-world system based on access level to the target ML model. We can categorize real-world attack patterns under 3 classes [19]: (i) direct attacks allow an adversary to submit inputs to the actual target and receive corresponding results, (ii) replica attacks use an exact replica of the target model to refine the adversarial input, (iii) transfer attacks select a substitute model which is a good-enough approximation of the target and use this model to craft adversarial examples. Among these, transfer attack is the most realistic attack strategy where an adversary may not have access to the target model.

Adversarial Defenses: Adversarial defenses aim to protect ML models against adversarial attacks based on three main classes [20]: (i) input defense, (ii) adversarial attack detection, and (iii) model defense. Input defense pre-processes input data to remove any adversarial component before the system back-end processes it. Data compression [21], data coding [22], data decomposition [23], and adding noise [24] are some example input defense strategies. Adversarial attack detection aims to distinguish attacked data from normal ones before model training and inference. Although it has been shown that these examples are not easy to detect [25], there are two different methods for adversarial attack detection: (i) data feature analysis utilizes data statistics analysis methods such as generalized likelihood ratio test [26], maximum mean

discrepancy [27], and temporal consistency [28], (ii) ML-based detection first extracts features and then train a decision model to determine if the input is an adversarial or not. Different ML methods have been used so far, e.g., one-class classifiers [29], extreme learning machine [30], and reinforcement learning [31]. Model defense is the last category of defenses which strengthen the model itself against adversarial attacks, i.e., increasing the robustness. This is the most heavily studied defense approach where there are numerous approaches such as gradient masking [32], defensive distillation [33], generative adversarial network [34], certified defenses [35], and adversarial training [36].

There are numerous IIoT security studies focusing on cryptography, authentication, patch management, and intrusion detection [37]. However, the adversarial attack and defense focus is lacking in majority of these. Hence, we fill this research gap by proposing resilient learning solutions and novel defense mechanisms for IIoT predictive analytics. We have two main contributions. First, we develop a stacking ensemble learning-based framework that stays resilient against various white-box adversarial attacks. Second, we propose a diversity promoting ensemble adversarial training approach as a defense mechanism against adversarial attacks.

1.3 Efficient Learning

Machine learning (ML) is fundamental to extract useful information from collected IIoT data. IIoT systems consist of resource constrained devices where conventional ML models cannot be directly used. There might be numerous IIoT learning scenarios where these devices need to perform predictive analytics [38]. To enable this, there are three main ML challenges that need to be addressed towards more efficient IIoT predictive analytics:

- **Computational Complexity:** State-of-the-art ML algorithms are computationally intensive, and consume great amount of energy. The computing and resource capabilities of IIoT's resource constrained devices do not match up with the computing and power requirements of these ML models.

- **Real-time Learning:** IIoT is a dynamic system which requires streaming input, where the input is continuously processed as it arrives. This creates real-time decision making, requiring ML solutions to be extremely fast and capable of real-time learning.
- **Noise Robustness:** IIoT sensor data is shown to be noisy [39]. This can significantly affect the prediction performance of ML models, risking their wide deployment.

To solve these challenges, there is a need for an efficient and robust learning solution. Hyperdimensional Computing (HDC) was introduced as a brain-inspired learning solution for robust and efficient learning [40]. HDC models data using points of a high-dimensional space, called hypervectors. These points are manipulated with formal algebra operations to represent relationships between objects. Compared to deep learning, HDC has shown advantages such as smaller model size, reduced computation cost, one-shot learning capability, and robustness to noise. These features make HDC a promising solution for IIoT's resource constrained devices, and the potential for noise and variability. HDC has three main stages [41]: 1) encoding: mapping data into high dimensional vectors, called hypervectors (HVs), 2) training: combining encoded HVs to create a model representing each class with a HV, and 3) inference: comparing the test sample with the trained model to find the most similar class.

HDC has been used in a range of applications such as activity recognition [42], speech recognition [43], and biomedical signal processing [44]. However, there is no prior work utilizing HDC for IIoT predictive analytics. To fill this research gap, we first propose non-linear encoding-based HDC for intelligent fault diagnosis. Furthermore, HDC resiliency against adversarial attacks has not been completely understood. Although HDC can provide higher resiliency compared to the state-of-the-art ML models, its prediction performance can still be impacted by adversarial attacks. To show its resiliency thoroughly, we design a novel HDC specific adversarial attack which is more effective than state-of-the-art adversarial attacks.

1.4 Thesis Contributions

This thesis focuses on creating intelligent, secure, and efficient ML solutions for IIoT predictive analytics. It shows methods to increase prediction performance, secure ML algorithms against adversarial attacks, and deploying lightweight learning approaches for IIoT environments. We propose novel ensemble learning solutions that can perform well in many IIoT settings. To get sufficient data for learning, these IIoT environments deploy numerous small sensors and edge computing devices, creating security vulnerabilities, such as adversarial attacks that impact the prediction performance. We develop secure learning solutions to enhance the resiliency against adversarial attacks. These intelligent and secure learning solutions increase the computational burden of IIoT systems, which rely heavily on smaller computing devices. To solve this problem, we design efficient learning solutions for IIoT predictive analytics. The following discussion summarizes the contributions and outlines the rest of the thesis:

- Chapter 2 presents a novel ensemble learning solution that introduces diversity to minimize retraining overhead while keeping accuracy at a certain level. To ensure accuracy, we create a quadratic programming model that combines single learners effectively. For diversity, we measure similarity among different learner predictions and select the most diverse ones iteratively. The experimental results show that our approach provides 39.2% faster retraining with only 3.4% loss in accuracy compared to only accuracy included ensemble approach. This shows that our approach can adapt to changing conditions faster with high accuracy.

- Chapter 3 presents a novel few shot ensemble learning framework for intelligent fault diagnosis. Our approach combines five different Siamese neural network architectures using an iterative majority voting classifier. Our transfer learning-oriented experiments show that we can improve the best algorithm significantly while using very limited labeled data. We obtain up to 16.4% improvement over the best algorithm by only using 0.3% of

the training data.

- Chapter 4 proposes a resilient stacking ensemble learning-based framework against white-box adversarial attacks. We first train DL methods from three different architectures: recurrent, convolutional, and hybrid. Our ensemble learner then combines the most resilient DL method predictions based on our iterative selection procedure. Our experiments show that adversarial attacks can impact the performance of DL-method considerably, leading up to 120$\times$ prediction performance loss. We use this performance loss to quantify method resiliency, where more resilient methods would lead to smaller performance loss. The proposed stacking ensemble improves resiliency against adversarial attacks by up to 60% compared to the most resilient single learner.

- Chapter 5 presents a novel defense framework against adversarial attacks. The idea is based on the fact that diversity among the ensemble learners plays a crucial role in reaching overall ensemble robustness. We first measure the loss gradient similarity among pre-trained ML models and select the most dissimilar ones to promote diversity. Then, we create perturbed training instances using the selected diverse base learners and augment those examples into our training data. In testing, we measure the performance change after adversarial attacks are introduced. Our experiments show that we can improve the resiliency by up to 97% (43% on average) compared to state-of-the-art training settings.

- Chapter 6 presents hyper-dimensional computing (HDC) as an efficient, robust, and resilient ML method. We first apply non-linear encoding based HDC for intelligent fault diagnosis. We then test HDC against various adversarial attacks. Our black-box adversarial attacks first train a substitute model and create perturbed test instances using this trained model. These examples are then transferred to the target models. The change in the classification accuracy is measured as the difference before and after the attacks. This change measures the resiliency of a learning method. Our experiments show that HDC leads to a more resilient and lightweight learning solution than the state-of-the-art deep

learning methods. HDC has up to 67.5% higher resiliency compared to the state-of-the-art methods while being up to 25.1× faster to train.

- Chapter 7 proposes a novel adversarial attack design for HDC. Although HDC is shown to be more resilient than state-of-the-art deep learning models, it is still vulnerable to adversarial attacks. To show its resiliency, we devise a novel adversarial attack for HDC. We first select the most diverse set of attacks to minimize overhead, and eliminate adversarial redundancy. Then, we perform a real-time attack selection which finds out the most effective attack. Our experiments on a realistic IIoT intrusion data set show the effectiveness of our attack design. Compared to the most effective single attack, our design strategy can improve attack success rate by up to 36%, and F_1 score by up to 61%.

- Chapter 8 summarizes our dissertation and discusses possible future research directions in the field of IIoT predictive analytics.

Chapter 2

Diversity-induced Optimally Weighted Ensemble Learner

2.1 Introduction

Industry 4.0 is an important milestone in production systems where smart manufacturing has become an essential component. This leverages the adaptation of traditional Internet of Things (IoT) for manufacturing, creating the Industrial IoT (IIoT) notion. IIoT enables the interconnection of anything, anywhere, and at any time in the context of industrial applications [1]. This allows better automation, and more fine-grained control, utilizing computer networks to collect enormous amount of data from the connected machines and convert this data into actionable information [45]. To optimize the performance of these smart and automated systems, timely and correct maintenance decisions are necessary, eliminating unplanned downtime, and better inventory management. The focus of IIoT is the appropriate management of industrial assets along with predictive maintenance (PdM) [46]. For example, BOSCH maintains quality management through PdM based on big data analysis, smart sensors, and artificial intelligence [47]. PdM predicts time-to-failure of a machine by employing copious mathematical approaches where it finds an optimum time to schedule a maintenance before any failure occurs.

Remaining useful life (RUL) is defined as the remaining time of a machine to perform its functions until it fails. RUL prediction is an important PDM application [9], which can significantly improve the overall system performance. Even the slightest RUL prediction improvement

can lead to enormous cost gains. Recently, data-driven RUL prediction methods have become popular with the abundance of available data. These machine learning (ML) oriented methods find a mapping between historical data and RUL. Among these, deep learning (DL) methods are more preferable due to their superior performance. However, industrial systems consist of large number of individual systems and components, and finding a single method that works best in these various settings is a difficult task. Also, since PdM data might show extreme variations in early vs. late phases (e.g., machines do not fail early), these ML models may need retraining with the new incoming data. The retraining phase is the performance bottleneck of these models, with significant computation time overhead. To achieve the vision of IIoT and utilizing from the huge data generated by IIoT devices, there is a need for lightweight prediction approaches [46].

Ensemble learning combines multiple algorithms, i.e., base learners, and improves base learner performance. This results in more successful data analytics, increasing system robustness and decreasing maintenance costs of IIoT systems. Ensemble learning eliminates single method selection but may create additional overhead as multiple methods need to be trained (and retrained with new data). Previous works generally use a small number of base learners, around 5-7, to avoid this issue [48]. However, there may be other methods performing better. Thus, the main issues with ensemble learning-based RUL prediction become 1) how many learners to consider for best performance and 2) how to select from them to minimize the retraining overhead.

The contribution of this chapter is to develop a diversity-induced optimally-weighted ensemble learner for RUL prediction (*OPTDIV*). To address the first issue, we dramatically increase the set of base learners to 20 methods, more than $2\times$ compared to the state-of-the-art. We show single method performance on NASA C-MAPSS data set [49]. Some of these methods, e.g., Wavenet are used for RUL prediction for the first time. We observe that the best performing algorithm changes with data set. Next, our ensemble learner uses a quadratic programming model to find the base learner optimal weights. This optimizes the accuracy of our ensemble learner. Finally, we consider model diversity to reduce base learner set, by calculating the *Euclidean* distance among base learner predictions. This step selectively reduces the base learner

set, reducing the computation overhead of the retraining phase. We compare our proposed model *OPTDIV* against optimally weighted ensemble learner *OPT* (without diversity) and see that *OPTDIV* has 39.2% faster retraining than *OPT* with only 3.4% loss in accuracy. This enables *OPTDIV* to adapt to changing conditions faster with high accuracy.

2.2 Related Work

The research on improving RUL prediction performance gained popularity due to advances in condition and health monitoring techniques. We can categorize the RUL prediction approaches under three main categories: experience based, model-driven (physical) and data-driven. Experience based models rely solely on expert knowledge and they are specific to a machine, that is why they are really hard to generalize. Physical models incorporate the physics of failure into RUL estimation [50]. The failure mechanism, e.g., fatigue, wear, is included in a mathematical model, establishing a relationship between the usage of a system to a lifetime prediction. It is difficult to create these models and they are machine specific.

Data-driven models (the ML approach) use historical data to learn a model of system behavior [51]. ML methods play a vital role in IIoT with use cases in maintenance cost optimization, and manufacturing process improvement [52]. Since traditional ML models require extensive feature extraction, DL recently became more popular to provide end-to-end RUL prediction [53]. For instance, Hyundai Motors Co., a car manufacturer, recently developed a DL-based car fault diagnosis system that is superior to expert analysis [47]. DL also provides better prediction performance than traditional ML methods. There are numerous DL methods adopted for RUL prediction: recurrent neural network [54], convolutional neural network [55], long short-term memory [56, 57], auto-encoders [58, 59], and etc. There are also hybrid models which combine two or more models for better prediction such as CNN and LSTM [60].

Best algorithm may change across different systems (and thus data sets) [61]. An ensemble learner combines predictions from multiple models and it performs better than the

single methods. To fulfill the theoretical requirements of good ensembles, base learners should be both accurate and diverse [62]. In this chapter, we incorporate these two components into our ensemble learner by proposing an iterative framework. For RUL prediction, there are different ensemble learning approaches. Li et al. [63] combine multiple traditional ML-based learners by using particle swarm optimization and sequential quadratic programming to determine their weights. Shi et al. [64] utilize ensemble learning to predict RUL of bearings. Li et al. [65] assign an optimized, degradation-dependent weight to each learner to obtain better prediction accuracy. All these works do not reach great prediction performance since they did not use DL methods (i.e., they solely consider traditional ML methods). They also did not consider the diversity aspect of the base learners. In this chapter, we utilize various DL methods and improve their single prediction performance by proposing a diversity-induced ensemble learner framework.

2.3 Proposed Framework

DL is particularly useful for IIoT applications since it can model a complex production environment with high prediction accuracy. However, a single DL model may not provide accurate RUL predictions. Alternatively, distinctive models can be combined (ensemble learner) for better prediction. Similarly, it is onerous to choose the base learners. Mistakenly selected base learners may lead to deficient performance. To solve this problem, we create a diversity-induced optimally weighted ensemble learner in this chapter. We present our framework in Figure 2.1. First, we start with data pre-processing module where we normalize the data and select the most useful features. In DL module, we create a DL library where we consider 20 state-of-the-art models ranging from vanilla recurrent neural networks to temporal convolutional networks. In our last module, we create our ensemble model by following two steps: accuracy and diversity. In accuracy, we formulate an optimization problem to find the optimal weights of the base learners, then we eliminate the models which have zero weight. In diversity, we first create a diversity matrix to measure similarity among selected base learner predictions. Afterwards, we select the

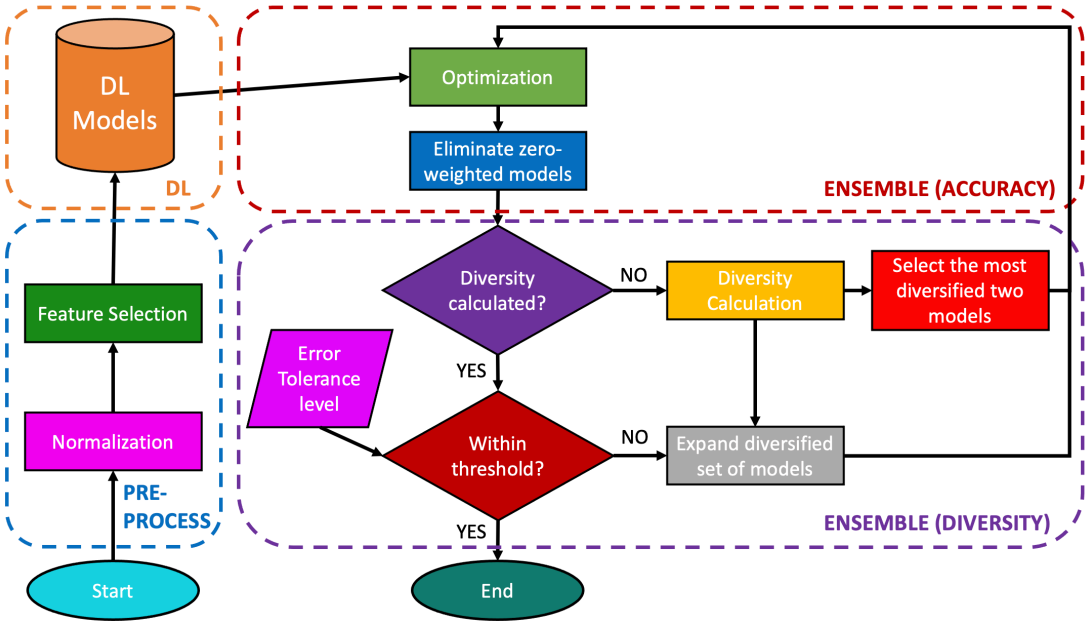


Figure 2.1. Proposed Framework for RUL Prediction

most diverse models and discover their optimal weights while keeping ensemble accuracy at a certain level by introducing error tolerance level.

2.3.1 Data pre-processing Module

We normalize the input sensor data using min-max normalization. For the feature selection, we utilize the Random Forest (RF) algorithm to discover variable importance which is calculated based on the reduction in residual sum of squares. Based on the calculated feature importance values, we keep the features that take positive importance values. Feature selection is crucial to increase model prediction performance.

2.3.2 Deep Learning Module

The deep learning module outputs single prediction model RUL values using the input sensor data. We implement 20 DL prediction methods. We employ sliding time window approach to convert time series sensor data into a regression problem. Table 2.1 presents the selected deep learning models. We categorize those models under three different architectures.

Table 2.1. Selected Deep Learning Models

Architecture	Category	Sub-category	Abbreviation	Explanation
Recurrent	Recurrent Neural Network(RNN)[66]	Vanilla	RNN	Simple RNN
		Layer Normalized	LNRNN	Normalization across the features dimension
	Long Short-Term Memory (LSTM) [67] Gated Recurrent Unit (GRU) [68]	Vanilla	LSTM GRU	Special memory cells
		Bi-directional	BLSTM BGRU	Backward direction (future and past data)
		Parallel	PLSTM PGRU	Two parallel paths
Convolutional	1-D Convolutional Neural Network (CNN) [55, 69, 70, 71]	Vanilla	CNN	Simple 1D convolution
		Depthwise separable	DSEPCNN	Single input channel convolution
		Wavenet	WAVE	Dilated and causal convolutions, residual and parameterized skip connections
		Temporal Convolutional Networks	TCN	Dilated and causal convolutions
Hybrid	CNN-RNN [72]	Vanilla Parallel	CRNN PCRNN	Serially and in parallel connected CNN and RNN layers
	CNN-LSTM [60]	Vanilla Parallel	CLSTM PCLSTM	Serially and in parallel connected CNN and LSTM layers
	CNN-GRU [73]	Vanilla Parallel	CGRU PCGRU	Serially and in parallel connected CNN and GRU layers
	CNN-LSTM-GRU [74]	Parallel	CLG	In parallel connected CNN, LSTM, and GRU layers
	CNN-RNN-LSTM-GRU	Parallel	CRLG	In parallel connected CNN, RNN, LSTM, and GRU layers

1) Recurrent Architectures: For recurrent models, hidden state represent everything that has been seen so far. Recently, they become popular in RUL prediction and demonstrated good performance [56]. All models contain three recurrent layers (e.g. RNN, LSTM) having 64, 32, and 16 units. After recurrent layers, all models are connected to two fully connected feed forward neural networks (each with 8 units) and final one-dimensional output layer.

2) Convolutional Architectures: Convolutional neural networks (CNN) use multiple feature extraction stages that can automatically learn hidden representations. Particularly, for RUL prediction, many CNN models have shown a great success [55]. Especially, 1-D CNN is common for time series applications, making it a suitable model for RUL prediction. The considered CNN network structures contain five consecutive CNN layers, Flatten (Dropout) layer, one fully connected layer (with 100 nodes) and an output layer with one node.

3) Hybrid Architectures: Hybrid models blend two or more methods to integrate strong aspects of different methods and to create more powerful estimator. For instance, combining CNN

and LSTM can be useful since CNN can handle feature extraction, and LSTM can build long term time dependencies. Based on the previously selected vanilla recurrent and convolutional models, we construct hybrid models. We consider both in series and in parallel connection.

2.3.3 Ensemble Module

A theoretically good ensemble method should include both accurate and diverse base learners [62]. In our ensemble module, we satisfy this condition by considering accuracy and diversity steps consecutively. In *accuracy*, we find the optimal weights for our base learners by solving an optimization problem. Then we eliminate the zero-weighted models to keep the most important ones. In *diversity*, we measure similarity among remaining models based on Euclidean distance and select the smallest model subset that is able to meet the desired threshold criteria. The threshold level measures how much worse performance our diversified subset can tolerate than the optimally weighted ensemble learner.

1) Accuracy: The goal of the accuracy step is to discover the most important methods in RUL prediction. The importance is measured by the optimal weights assigned to the base learners. To find those weights, we formulate a mathematical optimization problem where we minimize the mean squared error (MSE). Mathematically, MSE is formulated using the variance and bias of an estimator $\hat{\theta}$:

$$MSE(\hat{\theta}) = Variance(\hat{\theta}) + Bias^2(\hat{\theta}) \tag{2.1}$$

We specifically consider MSE since minimization of MSE is equivalent to minimizing bias and variance simultaneously, as shown in Equation 2.1. Accordingly, we formulate the following mathematical optimization model:

$$\text{minimize} \quad \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \Sigma_{j=1}^{M} w_j \hat{y_{ij}}\right)^2 \tag{2.2}$$

$$\text{subject to} \quad \sum_{j=1}^{M} w_j = 1 \tag{2.3}$$

$$w_j \geq 0 \quad \forall j = 1, \ldots, M \tag{2.4}$$

where N is the number of observations, M is the number of base learners, y_i is the true values for an observation i ($i = 1, \ldots, N$), $\hat{y_{ij}}$ is the predicted values for an observation i by the base learner j ($j = 1, \ldots, M$). w_j is the weight corresponding to the base learner j. The objective function (2.2) minimizes the MSE, constraint (2.3) ensures that weights sum up to 1, and constraints (2.4) restrict all weights to be non-negative. The constructed model has a convex quadratic objective function. We prove this by showing that Hessian (i.e. matrix that organizes all the second partial derivatives of a function) of our objective function is positive semidefinite (PSD). Without loss of generality, objective function (2.2) can be reformulated using L2 norm and matrix notation:

$$\|y - \hat{Y}w\|_2^2 \tag{2.5}$$

where y is an N dimensional vector storing real values, $\hat{Y}$ is an $N \times M$ prediction matrix and w is an M dimensional weight vector. For simpler notation, let ψ be a function that maps w to $\|y - \hat{Y}w\|_2^2$:

$$\psi : w \mapsto \|y - \hat{Y}w\|_2^2 = \|y\|_2^2 - 2y^T\hat{Y}w + \|\hat{Y}w\|_2^2 \tag{2.6}$$

Note that ψ is twice differentiable. The first and second partial derivatives of ψ with respect to w and w^T are:

$$\frac{\partial \psi}{\partial w} = -2y^T\hat{Y} + 2w^T\hat{Y}^T\hat{Y} \tag{2.7}$$

$$\frac{\partial^2 \psi}{\partial w \partial w^T} = 2\hat{Y}^T\hat{Y} \tag{2.8}$$

We also need to show $\hat{Y}^T\hat{Y}$ is a PSD matrix. Let ξ be an M dimensional vector. We prove

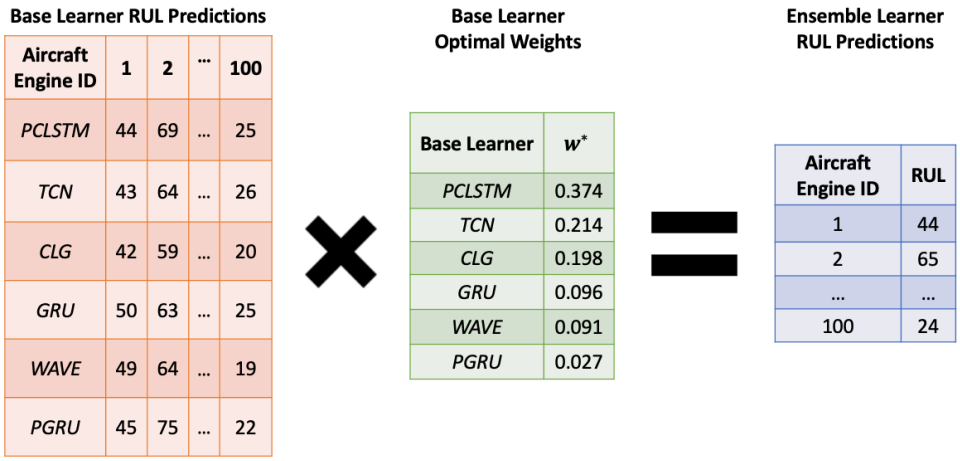


Figure 2.2. Dot Product Operation

that $\hat{Y}^T\hat{Y}$ is PSD:

$$\xi^T(\hat{Y}^T\hat{Y})\xi = (\hat{Y}\xi)^T(\hat{Y}\xi) = \|\hat{Y}\xi\|_2^2 \geq 0 \tag{2.9}$$

We proved that our objective function is convex and it is also quadratic. We also have affine constraints. Thus, above model is a quadratic program (QP) for which there is a guarantee that a local minimum is also the global minimum [75]. Solving this QP yields the optimal weights w^* for each DL model. Based on discovered w^*, we eliminate the models which have 0 weight. This means that unimportant models are eliminated from the larger set. Most importantly, we can achieve the optimal ensemble prediction performance using the base learner predictions and their corresponding weights. Figure 2.2 demonstrates our approach to combine the base learner predictions with their optimal weights. Here, we perform a dot product operation which takes single algorithm RUL predictions and optimal weights, and outputs RUL predictions. To exemplify, consider the first row of the ensemble learner RUL predictions (blue table) in Figure 2.2. In order to obtain the RUL prediction value as 44 (corresponding to the first aircraft engine), we multiply the first column of base learner RUL predictions (orange table) with the base learner optimal weights (green table) and round to the nearest integer. That is, $(44 \times 0.374) + (43 \times 0.214) + (42 \times 0.198) + (50 \times 0.096) + (49 \times 0.091) + (45 \times 0.027) \approx 44$. This optimization procedure considers solely accuracy of the base learners. We call this ensemble

learner *OPT* for the rest of this paper and we denote the number of models in *OPT* as τ. We shrink the number of models by including diversity in our ensemble learner.

2) Diversity: There is a trade-off between accuracy and diversity in ensemble learners. Adding diversity leads to worse prediction performance and vice versa. Our main goal of adding diversity is to obtain the smallest subset of DL models while preserving the accuracy at a certain level. We call this diversity-aware optimally weighted ensemble learner *OPTDIV*. The main motivation behind adding diversity is the reduced retraining overhead. This overhead becomes crucial when new data arrives. In that case, there is a need to retrain the selected models. Since *OPTDIV* has smaller number of models, it would be faster to retrain than *OPT* approach.

To measure diversity, there is no generally accepted metric to be used [62]. In a regression problem, covariance between individual estimators' outputs can be used for instance. We choose to utilize Minkowski distance to measure the similarity among methods' predictions of *OPT*. Minkowski distance measures the similarity between two points in the normed vector space. Consider an N dimensional vector space and two points: $X = (x_1, x_2, ..., x_N)$ and $Y = (y_1, y_2, ..., y_N)$. Minkowski distance $D(X,Y)$ of order p is formulated as:

$$D(X,Y) = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p} \tag{2.10}$$

When $p = 2$, Euclidean distance is obtained:

$$\sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{2.11}$$

Here, any p value can be used without loss of generality. We calculate the Euclidean distance among all possible pair of models and create a distance matrix $\Xi_{\tau \times \tau}$. This matrix demonstrates diversity among selected model predictions. For instance, $\Xi_{(TCN,GRU)}$ denotes the distance between prediction vectors of temporal convolutional network and vanilla GRU. If the distance is bigger, we expect those two models to be more diverse (since their predictions are

less similar).

First, we create the smallest subset of models by selecting the two most diverse models (*OPTDIV*). This is achieved by finding the maximum value in Ξ and its corresponding models. Afterwards, we solve previously formulated QP to calculate the optimal weights for those base learners and measure prediction performance. At this point, we check whether *OPTDIV* prediction performance is within desired threshold. We determine the threshold based on error tolerance level (δ) which is a parameter in [0,1]. Based on selected δ, we determine allowable performance difference between *OPT* and *OPTDIV*. To put more concretely, assume that in *OPT*, we found optimal ensemble RMSE to be 20. If we select error tolerance level δ to be 1%, then our proposed approach iterates until we reach an RMSE of 20.2 or less. At each iteration, we expand *OPTDIV* by selecting the next most diverse model (i.e. select the model that has the second highest element in Ξ and so on). Overall, our approach starts by selecting the two most diverse models, and increases the number of models until it meets the desired threshold level.

Let υ denote the number of models in *OPTDIV*. Observe that for any δ, we have $\upsilon \leq \tau$. Without loss of generality, we can assume that this inequality is strict. This assumption reveals the trade-off between computational overhead and accuracy. Since *OPTDIV* has smaller number of models, when the new data comes, retraining will be less costly than *OPT*. However, its accuracy would be worse (but not worse than δ). If we need much faster training and sub-optimal accuracy, *OPTDIV* would be a better choice.

2.4 Experimental Analysis

2.4.1 Experimental Setup

Dataset Description: NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) is a commonly used data set for RUL prediction. The simplified engine diagram in Figure 2.3 demonstrates the major components, e.g., fan, combustor of an aircraft engine [49]. Input data is created by different sensors, e.g., temperature, pressure placed on these components.

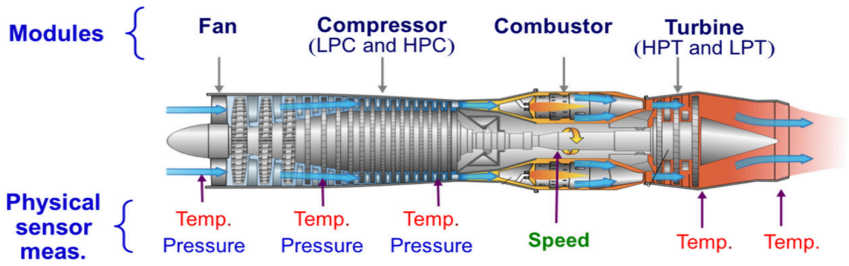


Figure 2.3. Aircraft Engine Diagram

Table 2.2. C-MAPSS Data Set Summary

Data Set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Max/Min cycles for train	362/128	378/128	525/145	543/128
Max/Min cycles for test	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

NASA C-MAPSS has four sets of data: FD001∼FD004. Each data set has different complexity levels, e.g., operating conditions, fault conditions as shown in Table 2.2. While FD001 is the simplest data set, FD004 is the most complex since it has the highest number of operating and fault conditions. Each data set has separate training and testing sets. The training data contains the entire lifetime of an engine, yet test data is terminated at some point before engine failure. Each row corresponds to a single operating time cycle with 26 columns: the engine ID, cycle index, three operational settings, and 21 sensor measurements. At the beginning, the engine is operating normally and develops a fault at some point in the future. The real RUL values are provided for the test data, and the goal is to estimate the RUL before a failure occurs.

RUL Target Function: The easiest and the most common approach to model RUL is linear where its value decreases with time. Nonetheless, it is hard to observe an apparent degradation behavior of an engine in the beginning of its lifetime. In general, an engine degrades more as it approaches its end of life. For the C-MAPSS data set, it is shown that piece-wise linear degradation model is more appropriate than linear model [76]. Thus, we adopt a piece-wise linear RUL target function where we set the maximum RUL limit constant (the break point) to

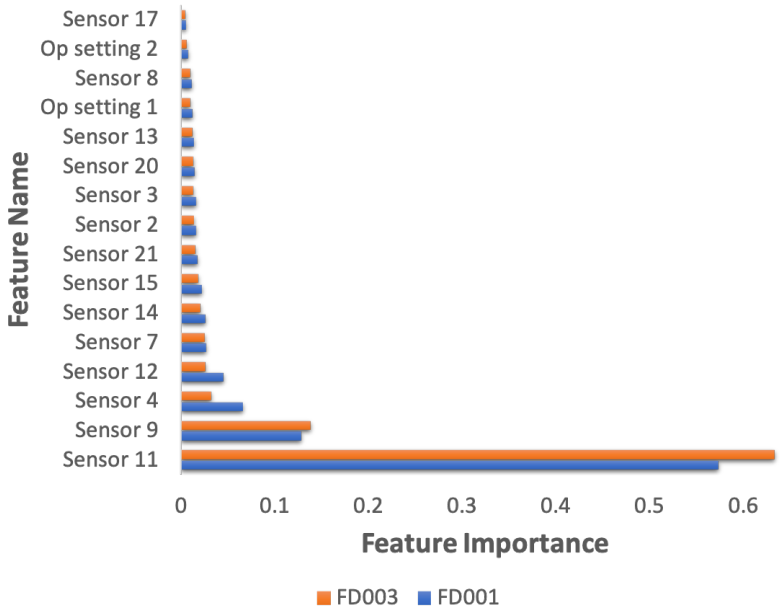


Figure 2.4. Random Forest Feature Importance

125-time cycles as in [77].

Performance Evaluation Metric: Prediction error (ε) is the difference between the estimated RUL (RUL_{est}) and the true RUL (RUL_{true}), i.e., $\varepsilon = RUL_{est} - RUL_{true}$. We use Root Mean Square Error (RMSE) for evaluation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^2} \tag{2.12}$$

2.4.2 Data pre-processing

We select the most important features in predicting RUL for (1) better prediction performance, (2) decreased computational complexity, and (3) preventing overfitting. We perform feature selection for FD001 and FD003 because these two demonstrate explicit health degradation behaviors. We employ the Random Forest (RF) algorithm to evaluate the feature importance. Figure 2.4 illustrates nonzero feature importance values for the FD001 and FD003 training data sets. While x-axis has feature importance values in [0,1], y-axis shows different features.

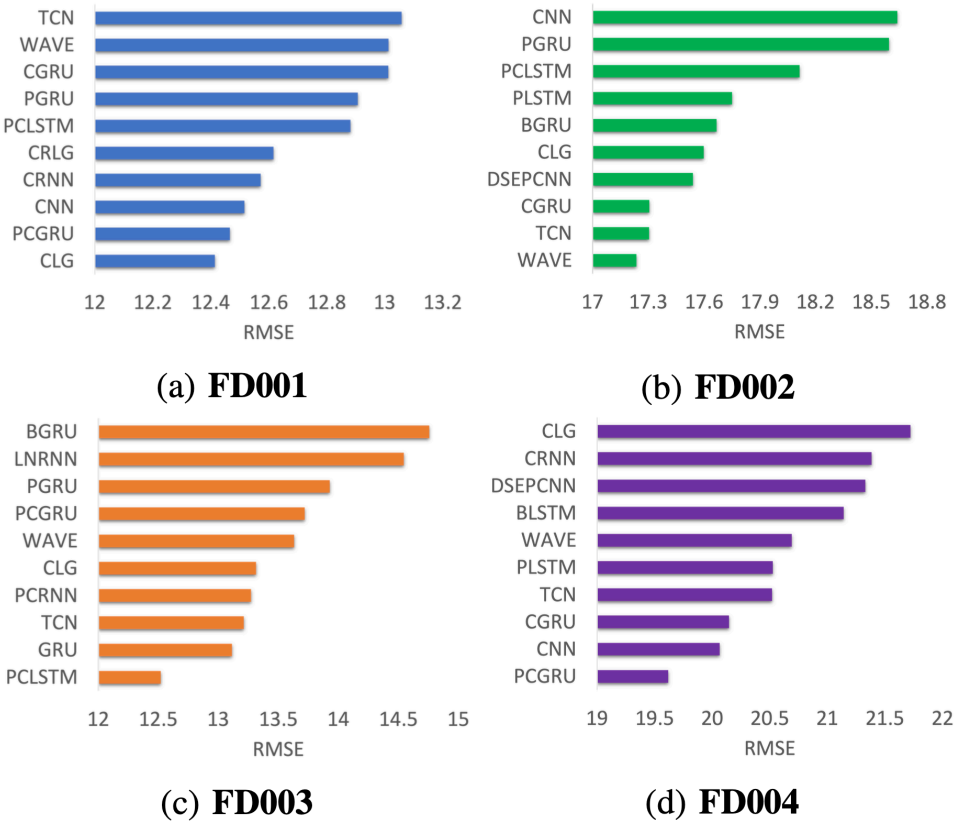


Figure 2.5. Deep Learning Models Prediction Performance

We observe that the most informative feature comes from sensor 11 for both data sets. It is important to note that this process is fully automatized, meaning that we do not need to visually inspect incoming data. When the new data comes, RF will update itself to calculate new feature importance values. Based on that, the most useful features will be updated and join the ensemble learning process.

2.4.3 Deep Learning Models Performance

All experiments are run on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. We run all models with the same parameter configuration: *Adam* optimizer with learning rate 0.001, *elu* activation function, *He* initialization, batch size of 512, and a max number of epochs of 250 where callback is activated (patience is set to 10 for validation data). We replicate each experiment 10 times and report average values. We adopt a sliding time

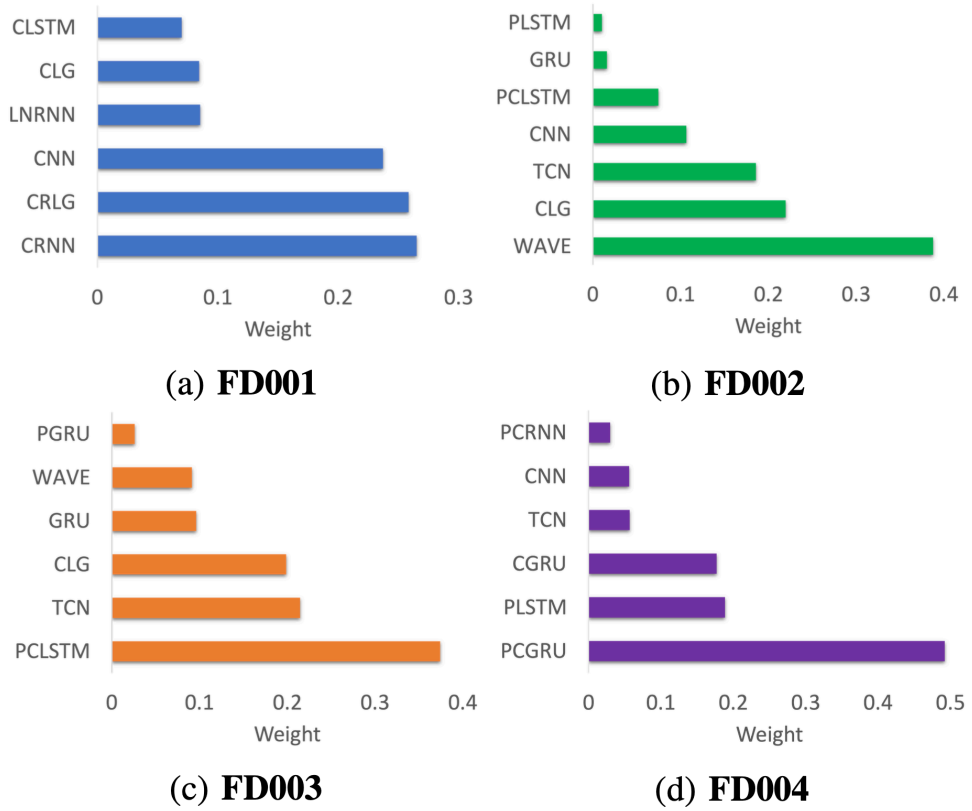


Figure 2.6. Base Learner Optimal Weights

window approach for better prediction performance. Here, we select the time window based on the minimum number of cycles for the test data since only sequences that meet the window-length can be considered. To use the entire test data, the window size is selected as 30, 20, 35, and 15 respectively from FD001 to FD004. Figure 2.5 depicts the best performing ten models across the data sets. At each sub-figure, horizontal axis represents RMSE value, and vertical axis provides specific DL model. The smaller the RMSE value, the better the model is. One key observation is that the best performing algorithm varies depending on data set. Specifically, CLG, WAVE, PCLSTM, and PCGRU are the best algorithms with RMSE values 12.42, 17.24, 12.52, and 19.62 respectively. We can also observe that hybrid and convolutional models perform better than recurrent ones.

2.4.4 Ensemble Learner Performance

Accuracy: To discover the most important models, we formulate and solve the previously formulated quadratic optimization problem in YALMIP [78] using the solver MOSEK [79]. As an output, we obtain the optimal weights w^* corresponding to each base learner. In Figure 2.6, we show the models that take non-zero weights. In each sub-figure, on the x-axis, we have optimal weights w^* while on the y-axis we have the corresponding DL models. CRNN, WAVE, PCLSTM, and PCGRU takes the greatest weights with 0.27, 0.39, 0.37, and 0.49 respectively. The number of models that have positive weights is different at FD002 where we have 7 models ($\tau = 7$) as opposed to 6 ($\tau = 6$) as in the remaining data sets. We also examine the relationship between base learner RMSEs and their optimal weights. Intuitively, we expect the base learner with the smallest RMSE to take the largest weight. However, this cannot be observed at FD001. Besides, even though some methods do not perform really well by themselves, they can still take a positive weight, e.g., PCRNN at FD004. In terms of optimal weights, we again cannot see one algorithm to be the best across all data sets, i.e., no dominant algorithm). When we multiply w^* with the corresponding learner predictions, we obtain *OPT* predictions. RMSE values of *OPT* are 11.95, 16.08, 12.04, and 19.17 respectively. Compared to the best single estimators, *OPT* brings 3.7%, 6.7%, 3.9%, and 2.3% accuracy improvement. Note that *OPT* has the most accurate predictions since we solve an optimization problem. Adding diversity degrades the estimation accuracy (trade-off between accuracy and diversity), yet it decreases the number of models which leads to lower computational overhead.

Comparison with State-of-the-Art Ensemble Methods: To verify the performance of our ensemble approach (*OPT*), we compare it with the state-of-the-art ensemble methods where we select Bagging, Gradient Boosting, AdaBoost, and Stacking [80]. We select these methods since these are the most well-known and successful ensemble approaches. As an input, 20 single method predictions are provided to these ensemble methods. We implement these methods using the scikit-learn library [81]. For the optimal hyper-parameter selection, we utilize GridSearch,

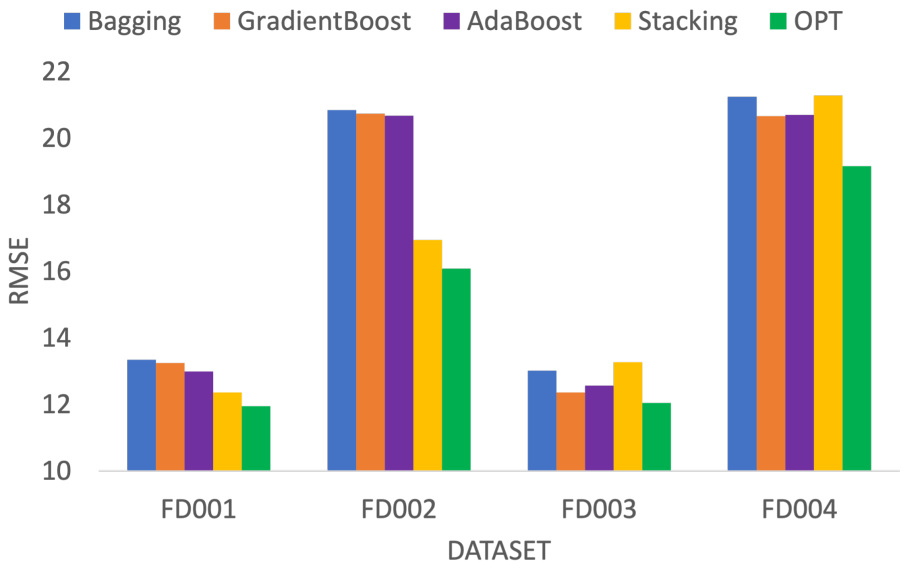


Figure 2.7. State-of-the-art Ensemble Comparison

which builds model on each possible parameter combination and finds the best hyper-parameter configuration [82]. Accordingly, we use the following hyper-parameters: AdaBoost $\mapsto$ number of estimators: 30, learning rate: 1, loss: linear; Gradient Boosting $\mapsto$ number of estimators: 90, learning rate: 0.1, loss: least squares; Bagging $\mapsto$ number of estimators: 20, max samples: 2, max features: 1; Stacking $\mapsto$ averaging the predictions of all base learners. We make a prediction performance comparison based on test data error. Figure 2.7 shows this comparative analysis where our method *OPT* (represented with green color) outperforms selected ensemble methods in all data sets. Specifically, *OPT* improves the prediction performance of the best ensemble method by up to 7.3% and 4.6% on average.

Diversity: We create the diversity matrix $\Xi_{\tau\times\tau}$ based on Euclidean distance between *OPT* model predictions. For demonstration purposes, we share a sub-matrix of $\Xi_{3\times3}$:

$$\Xi_{3\times 3} = \begin{array}{c} \begin{matrix} \textit{CRNN} & \textit{CRLG} & \textit{CNN} \end{matrix} \\ \begin{bmatrix} 0 & 7 & 3 \\ 7 & 0 & 1 \\ 3 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \textit{CRNN} \\ \textit{CRLG} \\ \textit{CNN} \end{matrix} \tag{2.13}$$

where we have a symmetric matrix with zero diagonal. Here, each cell represents a normalized diversity score. For instance, the most two diverse models are CRNN, and CRLG since the distance between those two points is the maximum. We start with the 2 most diverse models and expand this set until we reach desired error tolerance level δ. To provide better understanding, we consider all the models in *OPT* and create *OPTDIV* until $\upsilon = \tau - 1$. To exemplify, for FD001, we have $\tau = 6$, so we consider the most diverse models until $\upsilon = 5$. For each υ value, we calculate its RMSE and its performance degradation compared to *OPT*. Figure 2.8 demonstrates *OPTDIV* performance under varying υ values.

At each sub-figure, on the x-axis, we have υ values. On the y-axis we depict two different measurements: on the left we have RMSE values and on the right we have performance degradation measurements. Performance degradation is calculated by measuring how much accuracy difference there are between *OPTDIV* and *OPT*. For example, when we only use the two most diverse models in FD001, *OPTDIV* is 3.1% less accurate than *OPT*. Naturally, the more models we add, the more accurate *OPTDIV* becomes. However, in some cases, increasing υ does not bring significant advantage, e.g. in FD003, going from $\upsilon = 4$ to $\upsilon = 5$ increases accuracy only by 0.05%. Hence, we bring the idea of error tolerance level δ to select the smallest model subset for *OPTDIV* while not losing a significant accuracy. For example, given $\delta = 0.01$, *OPTDIV* has 4 models (υ = 4) at all data sets except FD004. If we select a bigger δ, then we end up with smaller υ values and vice versa. Figure 2.9 demonstrates this relationship. This figure shows selected υ values based on varying $\delta \in [0.04, 0.02, 0.01]$. In general, we can observe that as δ value gets larger, υ values become smaller. It means that *OPTDIV* has to select more

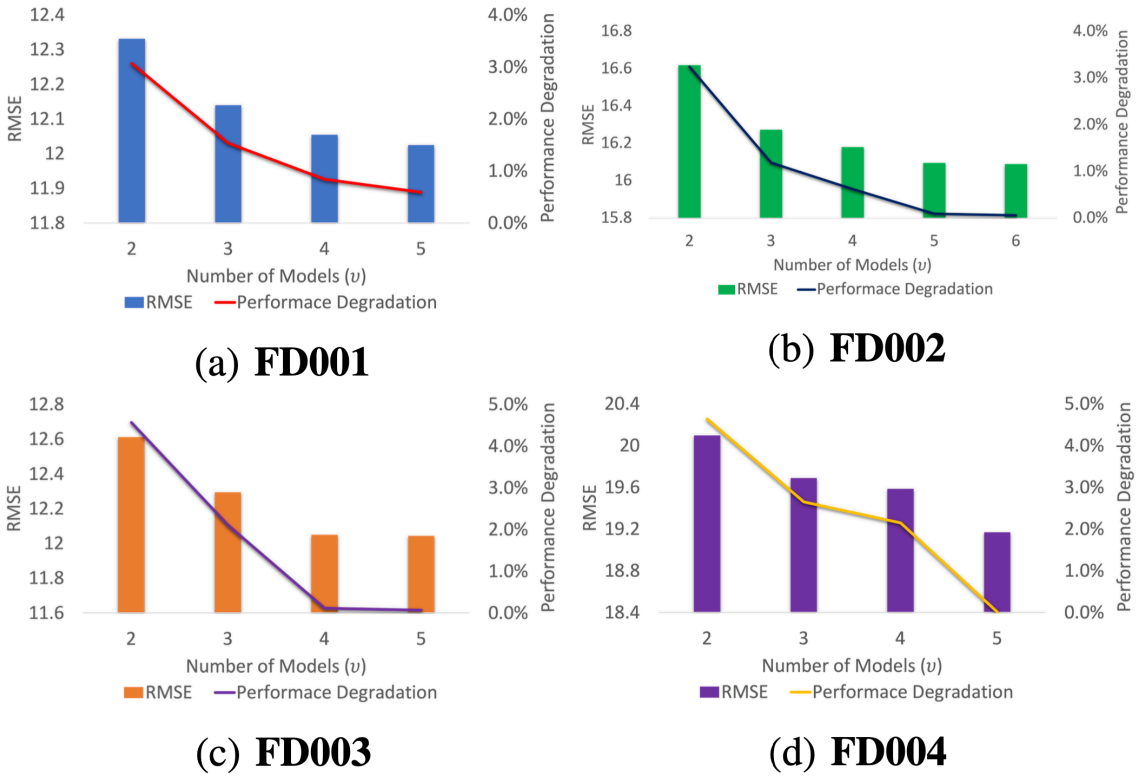


Figure 2.8. *OPTDIV* Performance for Different υ

models to obtain desired accuracy level. As we observed previously, adding more models may not bring a significant advantage in some cases.

2.4.5 Analysis on Retraining

This section analyzes the retraining performance and overhead of *OPT*, *OPTDIV*, and single ML methods when there is new data and retraining is required. For *OPT* and *OPTDIV*, we retrain only τ and υ number of models respectively. For single ML method, we have two approaches: *BEST* and *REBEST*. Both methods use a single ML method but the former uses the previously-identified best method, whereas *REBEST* retrains all methods to find whether the best method has changed with new incoming data. *BEST* eliminates the high retraining overhead, but has the risk to perform worse with new data. *REBEST* can update the best single model but with a much greater computation cost. We compare *OPT*, *OPTDIV*, *BEST*, and *REBEST* in terms of retraining time and prediction accuracy. To make that comparison, we use trained

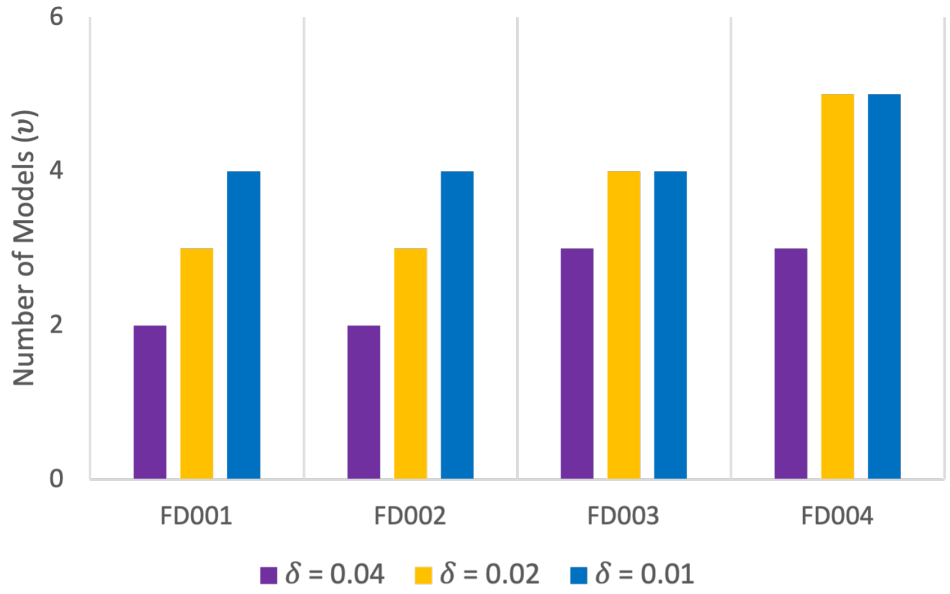


Figure 2.9. Effect of δ on υ

models of FD001, and treat FD003 as newly arrived data. We retrain the selected methods (pre-trained on FD001) based on FD003 train data, and test performance using FD003 test data. This represents that newly coming data can be different from the old data, i.e. FD003 has $2\times$ more fault conditions than FD001. Figure 2.10 and Figure 2.11 show the RMSE and retraining time (normalized against *BEST*) results for this analysis respectively. We consider three *OPTDIV* configurations with varying δ values.

Accuracy comparison: When we compare *OPT* and *OPTDIV* with $\delta = 0.01$, we observe that they perform equally. However, this was not the case on FD001 training where we observed 0.84% accuracy difference (see Figure 2.8). This means that performance difference has disappeared when we retrain these two approaches. Similarly, there is a 0.1% difference between *OPTDIV* ($\delta = 0.02$) and *OPT* which was previously 1.54%. This proves that adding diversity leads to better generalization, which is helpful when the data change. We should also note that *OPTDIV* with any δ value outperforms *BEST* and *REBEST* approaches significantly.

Overhead comparison: Between *OPT* and *OPTDIV*, retraining overhead difference becomes apparent when we compare *OPTDIV* ($\delta = 0.04$), and *OPT* approaches. As *OPTDIV*'s

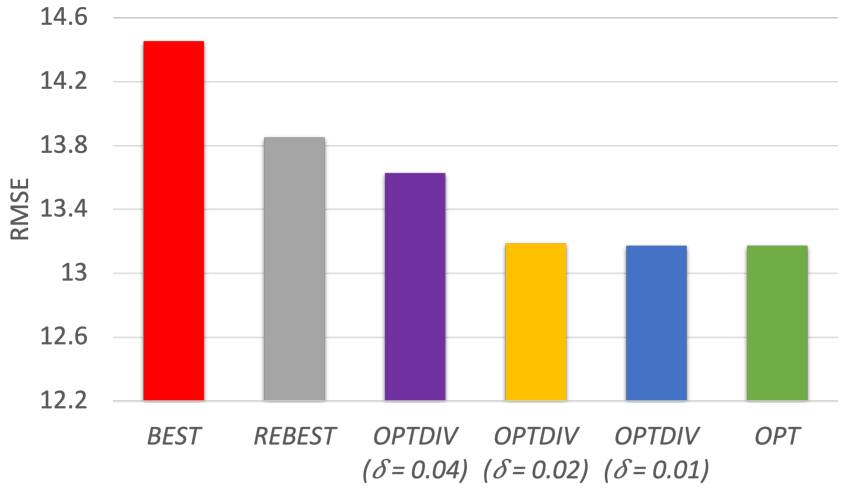


Figure 2.10. RMSE Comparison

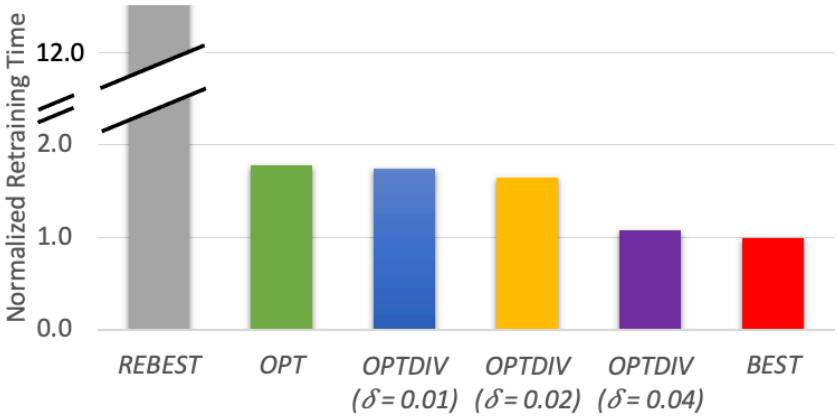


Figure 2.11. Normalized Retraining Time

error tolerance increases, it becomes much faster than *OPT*. *OPTDIV* ($\delta = 0.04$) is 39.2% faster with only a 3.4% loss in accuracy. We also discover that *OPTDIV* is 92% faster than *REBEST*. Compared to *BEST*, *OPTDIV* brings 5.7% performance improvement while only needing 8.5% slower retraining. In summary, *OPTDIV* can adapt to changing data much faster, with small loss in accuracy.

2.5 Conclusion

PdM requires accurate RUL prediction to obtain the best performance from production systems. DL-based RUL prediction methods become prevalent due to their high accuracy and easy implementation. However, choosing one algorithm may not provide accurate predictions across different settings and data sets. Hence, we propose diversity included accurate ensemble learner. We discover the smallest subset of learners out of 20 different DL methods while keeping accuracy at a certain level. We include accuracy by finding optimal weights of the base learners. We then measure the similarity among base learner predictions and select the most diversified set of models. We show that our approach can have 39.2% faster retraining compared to an accuracy-based ensemble learner with only 3.4% loss in accuracy.

In this chapter, we assumed that our data is fully labeled. However, in IIoT systems, labeling might be expensive due to continuously changing operating conditions. In the next chapter, we will address this challenge and propose a novel ensemble learning solution that can work under limited supervision.

2.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1527034, #1730158, #1826967, #1830331, #1911095, and #1952225.

Chapter 2, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "DOWELL: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices *IEEE Internet of Things Journal (IoT-J)*, 2021. The dissertation author was the primary investigator and author of this material.

Chapter 3

Ensemble Few-Shot Learning under Limited Labeled Data

3.1 Introduction

Fault diagnosis helps establishing more efficient predictive maintenance systems by finding and classifying different fault types [83]. Due to advancements in machine learning, numerous intelligent fault diagnosis methods have been developed [84, 85, 86]. These methods require huge amounts of labeled training data to work well. However, it might not be feasible to obtain sufficient training samples for multiple fault types under all working conditions due to: (1) slowly occurring failure processes, (2) frequently changing working conditions, and (3) that some critical industrial systems are not allowed to run into faulty states [87]. Thus, intelligent fault diagnosis methods should work with limited data and changing operating conditions.

This chapter proposes an ensemble few-shot learning framework (*ENFES*) for intelligent fault diagnosis. In contrast to the state-of-the-art, we use multiple few-shot learning (FSL) methods and combine them via our majority voting framework to improve the prediction performance. We focus on Siamese neural network owing to its high accuracy in fault diagnosis [87]. Given pairs of input data, we first train five different convolutional neural network-based Siamese neural networks. We combine the individual predictions using our iterative majority voting ensemble learner where the fault type voted the most by the classifiers is assigned as the final output. In case of a tie, the least accurate two models are eliminated based on validation set accuracy until

a single method remains. We evaluate our approach using Case Western Reserve University (CWRU) Bearing Datasets [88], a widely used benchmark for fault diagnosis. Since the working conditions can change frequently, we consider different transfer learning (TL) scenarios, where we train the model in one setting and test it in another. Here, the goal is to train and test model under different working conditions. We directly transfer the learned model (in training phase) for testing purposes. We observe that the best algorithm changes based on the underlying transfer learning scenario indicating that one algorithm would not work the best in all cases. *ENFES* can bring a significant performance improvement over the best algorithm with very limited training data, up to 16.4% and 10.7% improvement over the best method with only 60 and 90 samples (out of 19,800 samples), respectively.

3.2 Related Work

Few-shot learning (FSL) performs well with limited labeled data [89, 90, 91]. FSL-based fault diagnosis learns classifiers given only a few labeled examples of each fault type where the goal is to find similarity among input pairs. There are a variety of FSL approaches such as Siamese neural networks [92], matching networks [93], model-agnostic meta learning [94], and memory augmented neural network [95], and FSL-based fault diagnosis works [87, 96, 97, 98, 99]. These studies focus on using a single method for fault diagnosis. Previous studies found that the best performing algorithm for a problem changes with respect to the dataset [48, 100]. Thus, we need a systematic methodology to combine different methods. To achieve this, we utilize ensemble learning which is previously used for intelligent fault diagnosis [101, 102]. We also utilize transfer learning to adapt to changing working conditions. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. Transfer learning can enable transferring an already-learned model in an existing operating condition to a new condition that was not observed before [103].

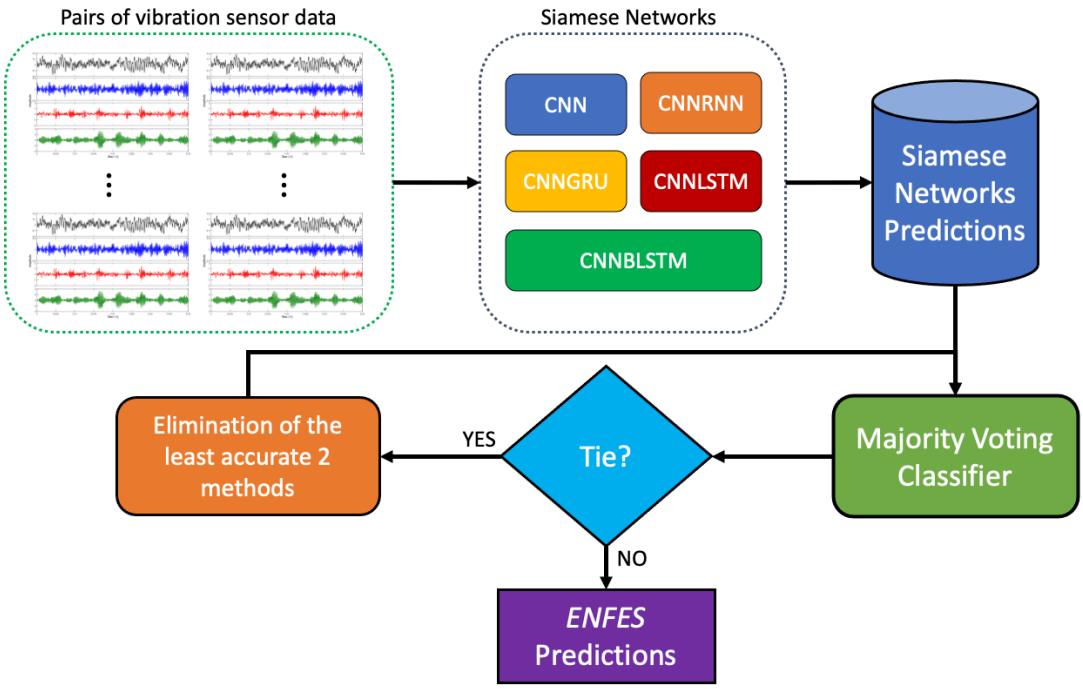


Figure 3.1. Our Proposed Framework

3.3 Proposed Framework

Figure 3.1 presents our proposed framework. Given pairs of input sensor data, we first train various Siamese neural networks. We collect the predictions of individual methods and then combine them using our majority voting classifier approach where the most voted fault type by the classifiers becomes the *ENFES* prediction. In case of a tie, the least accurate two models are eliminated from the models based on validation set performance. We explain two aspects of our framework in detail: Siamese neural networks, and majority voting classifier.

3.3.1 Siamese Neural Network

Although there are multiple sensors collecting data, it may not be feasible to label every failure point. Few-shot learning (FSL) aims to solve this problem by using limited labeled data. The goal of training an FSL model is to discover similarity between inputs coming from different fault types. To train an FSL model, a support set is given containing a small set of labeled fault

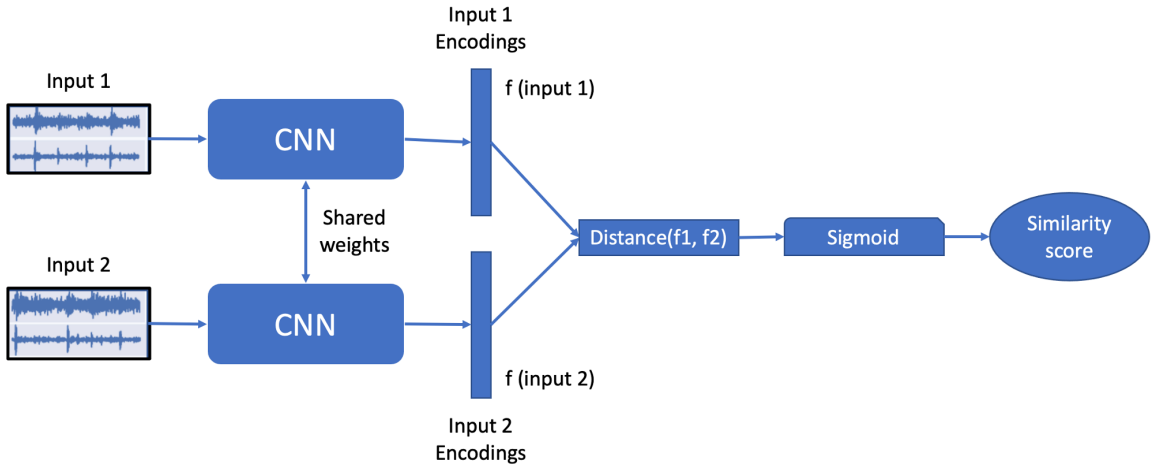


Figure 3.2. Siamese Neural Network (SNN) Structure

types. Given pairs of time series sensor data, training phase outputs the probability P that two input samples are the same. For testing, there are two different approaches: one-shot k-way testing and n-shot k-way testing. In one-shot k-way testing the support set Ψ contains k samples $x_1,\ldots,x_k$ with distinct labels $y_1,\ldots,y_k$: $\Psi = \{(x_1,y_1),...,(x_k,y_k)\}$. Given the test sample $\tilde{x}$, the goal is to output the class $\tilde{c}$ that maximizes the similarity probability:

$$\tilde{c} = \underset{c}{\operatorname{argmax}}(P(\tilde{x},x_c)),\ x_c \in \Psi \tag{3.1}$$

In n-shot k-way testing, support set has n samples from k classes $\Psi_1,\ldots,\Psi_n$. Given test sample $\tilde{x}$, we obtain class $\tilde{c}$ that maximizes the similarity probability over n samples:

$$\tilde{c} = \underset{c}{\operatorname{argmax}}(\sum_{j=1}^{n} P(\tilde{x},x_{cj})),\ x_{cj} \in \Psi_j \tag{3.2}$$

We focus on Siamese Neural Network (SNN) due to its high accuracy in fault diagnosis [87]. We illustrate an example of SNN structure in Figure 3.2. SNNs contain two or more identical sub-networks (same network architecture and shared weights). Outputs of these networks are mapped to a high-dimensional feature vector to calculate the distance between the inputs. By connecting feature vector to a distance function and a fully connected neural network,

Table 3.1. Single Method Fault Type Predictions

Bearing ID	*CNN*	*CNNRNN*	*CNNGRU*	*CNNLSTM*	*CNNBLSTM*
1	Ball (0.014 in)	Ball (0.007 in)	Ball (0.014 in)	Ball (0.014 in)	Ball (0.021 in)
2	Outer Race (0.021 in)	Outer Race (0.014 in)	Outer Race (0.021 in)	Outer Race (0.014 in)	Outer Race (0.021 in)
...	...	...	...	...	...
250	Inner Race (0.007 in)	Inner Race (0.014 in)	Inner Race (0.007 in)	Inner Race (0.021 in)	Inner Race (0.007 in)

we find the similarity probability between input pairs. Higher similarity probability indicates that two inputs are likely in the same category.

There are many alternatives for the sub-network selection, such as wide deep convolutional neural network (WDCNN) [104] and long short-term memory (LSTM) [105]. However, a single prediction method may not perform the best across different systems [61]. Ensemble learning solves this issue where predictions from multiple models are strategically combined. We create a CNN-based ensemble learning framework by considering five different architectures: convolutional neural network (*CNN*), CNN recurrent neural network (*CNNRNN*), CNN gated recurrent unit (*CNNGRU*), CNN long short-term memory (*CNNLSTM*), and CNN bi-directional long short-term memory (*CNNBLSTM*). We use the same CNN architecture across all methods as in [104, 87]. For the hybrid models (where two models are combined, e.g., *CNNRNN*), we connect CNN with the selected network structure consecutively, e.g., for CNNRNN, we connect CNN with 2 RNN layers with 32 and 16 nodes. We replace RNN layers with GRU, LSTM, and BLSTM for the *CNNGRU*, *CNNLSTM*, and *CNNBLSTM* respectively. These SNNs provide single model fault type predictions for our test data, shown in Table 3.1, where each model outputs its fault prediction for a given bearing. Then, we combine these predictions using our majority voting classifier. Notably, fault type predictions change with respect to the algorithm.

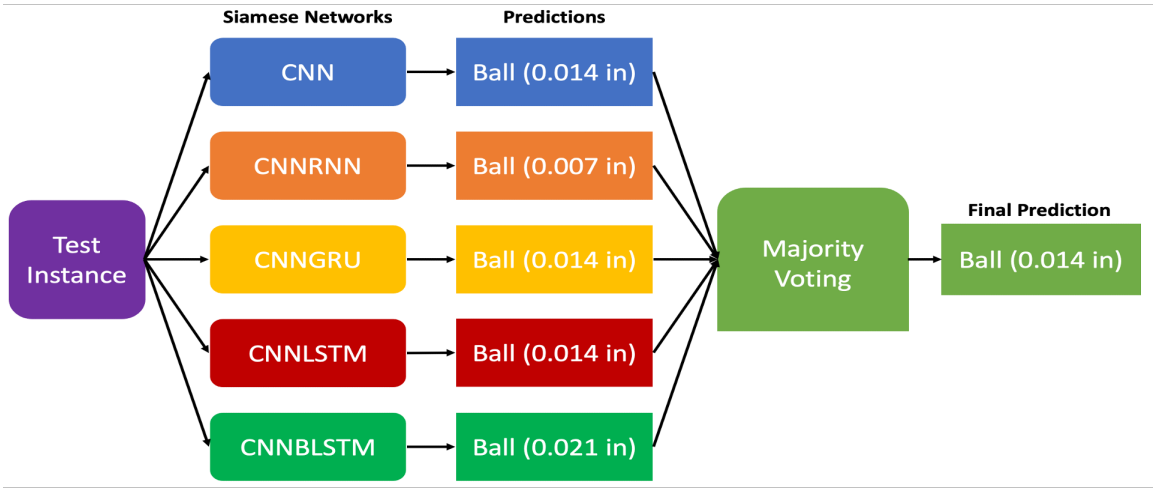


Figure 3.3. Majority Voting Classifier

3.3.2 Majority Voting Classifier

Majority voting classifier (MVC) is an ensemble learner that combines the class predictions of different methods. Assume that we have n different classifiers $\xi_1, \xi_2 \ldots \xi_n$ that map input data X to class $c_1, c_2 \ldots c_n$. MVC Ξ finds the class $\bar{c}$ that maximizes the weighted sum of correct class predictions [106]:

$$\Xi(X) = \bar{c} = \underset{c}{\operatorname{argmax}} \sum_{j=1}^{n} \omega_j I(\xi_j(X) = c) \tag{3.3}$$

where $\omega_1, \omega_2 \ldots \omega_n$ are the classifier weights summing up to 1. I is the indicator function which is 1 if the classifier prediction j is class c, and 0 otherwise. If we set all weights equal to each other (i.e. $\omega_j = \frac{1}{n}$), this formulation becomes a *mode* function which outputs the class most voted by the classifiers. The main problem occurs when there is a tie among n classifiers. To illustrate, say we have three classifiers ξ_1, ξ_2, ξ_3, with their fault type predictions c_1, c_2, c_3 which are all distinct from each other. In this case, there is a tie among three methods. To break a tie, we propose a method that iteratively eliminates the least accurate $n-1$ methods based on the validation set accuracy. We continue classifier elimination until there is one method left to guarantee tie-breaking. We specifically start with an odd number of methods (five) to reduce

possible tie scenarios among classifiers. Figure 3.3 presents majority voting classifier example. Given a test input sample, SNNs first predict the failure type, e.g., Ball (0.014 in). The majority voting classifier then selects the prediction that is mostly voted by the SNNs and mark this as the final prediction.

3.4 Experimental Analysis

3.4.1 Dataset Description

We use Case Western Reserve University (CWRU) Bearing Datasets [88], a widely used benchmark for fault diagnosis This data was collected at 12k samples/second and at 48k samples/second for drive end bearing experiments. We use the former data set that contains 19,800 training and 750 test samples. Bearing used in this experiment has three components: rolling element, inner race, and outer race. 9 different fault types are provided in the dataset based on the fault diameter (0.007, 0.014, and 0.021 inches) and the component (plus the normal bearing condition). Besides, three datasets (A, B, and C) are given representing different working conditions (based on motor speeds). We use transfer learning (TL) to transfer models across possible working conditions. For instance, transfer from dataset A to B trains model using dataset A, and tests this model on dataset B.

3.4.2 Experimental Setup

We use the same setup as in [87] because it led to great fault diagnosis performance: sliding window of size 2048 points sliding with 80 points shift step, L1 distance to calculate distance between feature vectors, regularized cross-entropy loss function, Adam optimizer, batch size of 32, 15000 as number of epochs, and five-shot learning scenario. We repeat each experiment 10 times and report average values. All experiments are run on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor.

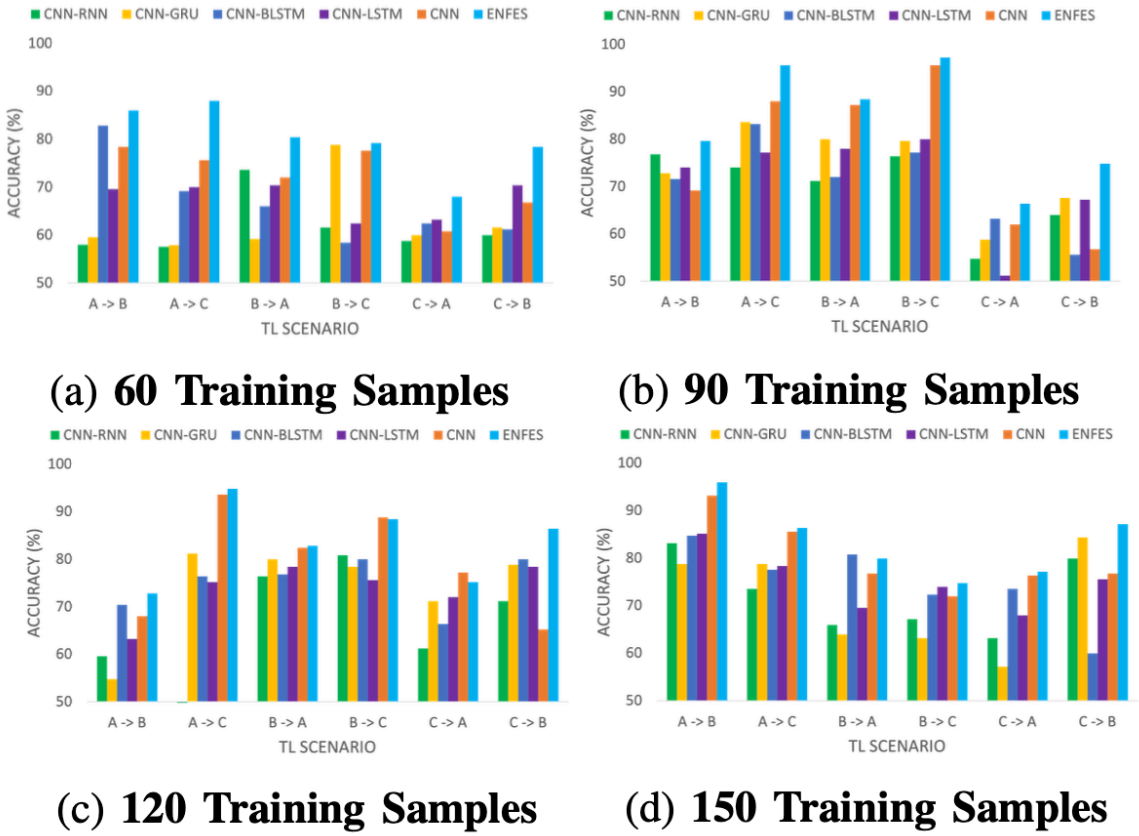


Figure 3.4. Transfer Learning Methods Comparison

3.4.3 Results

We analyze the performance of our method under a transfer learning (TL) setting to represent changing working conditions. In any production system, working condition of an equipment can alter frequently, that is why it is crucial for data-driven methods to generalize to unseen working conditions. We have three different datasets corresponding to three working conditions: A, B, and C. We experiment with pairwise transfer learning scenarios: transfer model from A to B, A to C, B to A, B to C, C to A, and C to B. Since our focus is on fault type prediction using limited amount of data, we only use 60, 90, 120, and 150 samples from the training data (out of 19,800 samples) while using the entire test data. Figure 3.4 presents the model performance comparison for different number of training samples and TL scenarios. In each sub-figure, the x-axis represents the transfer learning scenario (e.g., $A \mapsto B$ denotes

Table 3.2. Our Method's Improvement over the Best Algorithm (%)

	Number of Training Samples			
TL Scenario	**60**	**90**	**120**	**150**
A->B	3.86	3.65	3.41	3
A->C	16.4	8.64	1.28	0.93
B->A	9.24	1.38	0.5	-0.99
B->C	0.51	1.67	-0.45	1.08
C->A	7.59	5.06	-2.59	1.05
C->B	11.38	10.65	8	3.33
Average	**8.17**	**5.18**	**1.69**	**1.4**

transfer from dataset A to B), and y-axis is the accuracy of the methods. Each color denotes a different method where our method (*ENFES*) is shown with light blue color. The performance of a single method changes with respect to the TL scenario. For example, using 60 training samples, *CNNBLSTM* is the best method for $A \mapsto B$ while *CNNRNN* being best for the opposite transfer configuration (i.e. $B \mapsto A$) when single SNNs are compared. *ENFES* utilizes all 5 methods to improve the prediction performance. Under 60 and 90 samples, *ENFES* outperforms other methods consistently. Table 3.2 presents our proposed approach improvement over the best method for all TL scenarios. At 60 and 90 training samples, *ENFES* improves the best method by up to 16.4% and 10.7% respectively (8.2% and 5.2% average). As we have more training samples, the improvement over the best algorithm decreases, yet we still obtain improvement up to 8% and 3.3% under 120 and 150 training samples (1.7% and 1.4% average).

The reasoning behind decreasing *ENFES* improvement with increasing data is due to the performance of the best algorithm. For instance, at 60 samples $C \mapsto B$ scenario, the best algorithm *CNNLSTM* provides 70% accuracy. As we reach 150 samples, the best algorithm *CNNGRU* can reach 85% accuracy. In this case, the room for improvement is limited by this method's accuracy. Hence, improvement for *ENFES* decreases from 11.4% to 3.3%. The performance of a single algorithm changes with respect to TL scenario, e.g., *CNNGRU* is the worst algorithm at $C \mapsto A$

scenario under 150 samples. Our method provides consistent accurate predictions at all instances (it is either the best or the second best approach). *ENFES* provides an average improvement over the best method if we use less than 1% of the entire training data. This result aligns with our claim where *ENFES* has highly efficient prediction under limited labeled data.

3.5 Conclusion

Fault diagnosis determines which fault occurred in a production system [83]. By using historical sensor data and ML models, intelligent fault diagnosis methods can be created. Nevertheless, these methods require huge amount of labeled data to perform well. Few-shot learning (FSL) eliminates this restriction by discovering similarity among input pairs. However, selection of a single FSL method may not provide an optimal prediction performance. Different methods can be combined by using an ensemble learner. In this chapter, we proposed ensemble few-shot learning approach for fault diagnosis. We consider five different Siamese neural networks and combine their fault type predictions via our majority voting classifier. We show that our approach can improve the classifier accuracy significantly under different transfer learning scenarios.

In this chapter, we addressed the limited supervision challenge in IIoT systems. So far, we assumed that machine learning models do not receive any perturbed data, i.e., no adversarial attack. In the following chapter, we will design a novel ensemble learning solution that can stay resilient against different adversarial attacks.

3.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1527034, #1830331, #1911095, and #1952225.

Chapter 3, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, “ENFES: Ensemble Few-Shot Learning For Intelligent Fault Diagnosis with Limited Data”. *IEEE SENSORS*, 2021. The dissertation author was the primary investigator and author

of this material.

Chapter 4

Resilient Stacking Ensemble Learner Against Adversarial Attacks

4.1 Introduction

Recently, data-driven RUL prediction methods became popular since IIoT-based instrumentation has led to an abundance of system monitoring data. However, the performance of ML methods relies heavily on input data quality. Thus, these methods are quite vulnerable to adversarial attacks where an attacker can alter input data to impact ML prediction performance significantly. Since ML is a core element for data-driven RUL prediction, these attacks may have serious consequences such as wrong maintenance decisions causing undetected failures in a system [6]. We need novel ML solutions that can stay resilient against adversarial attacks.

The performance of an ML-based application also depends on the specific ML algorithm used. Selecting a single ML method is a difficult process since its performance may change drastically based on the underlying dataset [61]. For adversarial attacks, it is easier to decode ML model parameters for a single method, reducing the system resiliency against attacks [107]. Alternatively, ensemble learning combines multiple individual algorithms and it usually improves base learner prediction performance [64, 100]. Against adversarial attacks, many ensemble learning studies are proposed that are more resilient than a single method [108, 109, 110]. To the best of our knowledge, ensemble learning has not been used previously to show its superior resiliency in a realistic data-driven IIoT setting.

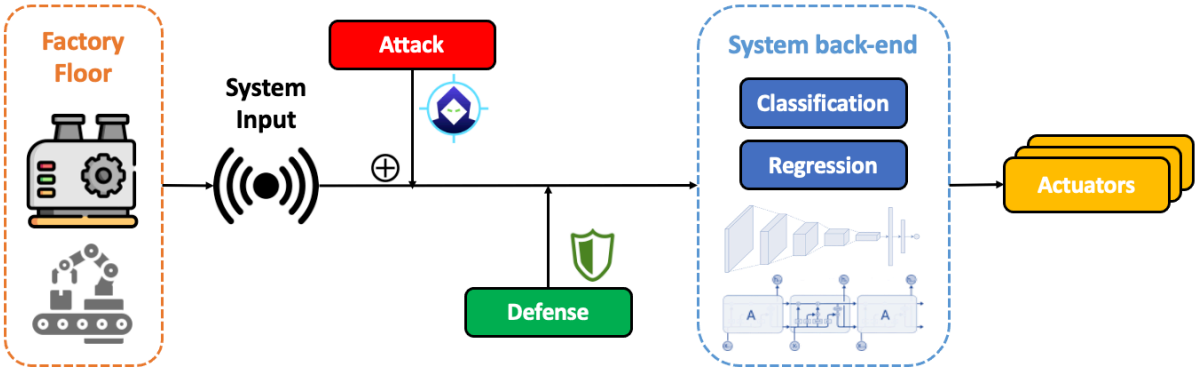


Figure 4.1. IIoT Adversarial Attack Workflow

In this chapter, we propose a stacking ensemble learning framework which can stay resilient against different adversarial attacks. We first train ten different deep learning (DL) methods from three different architectures: recurrent, convolutional, and hybrid. Our ensemble learner then combines the most resilient DL method predictions based on our iterative selection procedure. Using NASA C-MAPSS [49], and UNIBO Powertools [111] dataset, we demonstrate that adversarial attacks can impact the performance of DL-method considerably, leading up to 120× prediction performance loss which can lead to premature replacements or completely missed maintenance decisions. We use this performance loss to quantify method resiliency, where more resilient methods would lead to smaller performance loss. Our experiments show that proposed stacking ensemble approach improves resiliency against adversarial attacks by up to 60% (48% on average) compared to the most resilient single method.

4.2 Related Work

4.2.1 IIoT Adversarial Attack Workflow

To understand how adversarial attacks can take place in IIoT environments, we present the IIoT adversarial attack workflow in Figure 4.1. The workflow consists of 4 main components: (i) System input is the data collected from sensors in a factory environment. (ii) Adversarial attack begins with the attacker exploring any vulnerability to compromise one common system

Time stamp	$Sensor_1$	$Sensor_2$	...	$Sensor_S$	RUL
1	$Reading_{1,1}$	$Reading_{2,1}$	...	$Reading_{S,1}$	RUL_1
2	$Reading_{1,2}$	$Reading_{2,2}$	...	$Reading_{S,2}$	RUL_2
...	...	...	...	...	...
N	$Reading_{1,N}$	$Reading_{2,N}$	...	$Reading_{S,N}$	RUL_N

Figure 4.2. Multivariate time-series data

input data as victim. The attacker then induces a model to craft perturbations. After tampering with the data inputted into the system back-end, the attacker can add malicious perturbations into the input data and complete the attack. (iii) Defense refers to protection and mitigation strategies against attacks. It plays a key role in minimizing the impact of adversarial attacks. (iv) System back-end applies different ML methods to process the collected input sensor data to generate the desired information to control the actuators.

4.2.2 Adversarial Attack Formulation for RUL prediction

The multivariate time-series input data with its corresponding RUL values is illustrated in Figure 4.2. In this figure, we have S sensor data from N consecutive time stamps where each cell represents individual sensor readings. We make the following mathematical definitions:

1. $\tau_i \in \mathbb{R}^S : [Reading_{1,i}, Reading_{2,i}, \ldots, Reading_{S,i}]$ is the vector containing all sensor readings for the time stamp i, $\forall i = 1, \ldots, N$.

2. $T \in \mathbb{R}^{S \times N} : [\tau_1, \tau_2, \ldots, \tau_N]$ represents the multivariate time-series data.

3. $D \in \mathbb{R}^{(S+1) \times N} : [(\tau_1, RUL_1), (\tau_2, RUL_2), ..., (\tau_N, RUL_N)]$ denotes the supervised training data.

4. $f(.) \in F : \mathbb{R}^{S \times N} \mapsto \mathbb{R}^N$ is DL model which maps all sensor readings to the remaining useful life prediction values $\hat{RUL}$.

5. $L_f(.,.)$ denotes the loss function of the model f.

6. $\ddot{T} = T + \delta T$ is the crafted adversarial example. $\ddot{T}$ is obtained by adding a perturbation δT with the sample T such that $R\ddot{U}L \neq R\hat{U}L$ and $\|\ddot{T} - T\| \leq \varepsilon$ where $\varepsilon \geq 0 \in \mathbb{R}$ is a maximum perturbation magnitude, $\|.\|$ is any norm w.l.o.g (e.g., L_∞), $f(T) = R\hat{U}L$, and $f(\ddot{T}) = R\ddot{U}L$.

7. Given a trained DL model f and original data T, adversarial example $\ddot{T}$ is found as a solution to the following box-constrained optimization problem:

$$\ddot{T} = T + \underset{\delta T}{\operatorname{argmin}}\{\|\delta T\| : f(T + \delta T) \neq f(T)\} \tag{4.1}$$

   This problem yields the minimum perturbation amount δT while ensuring that RUL prediction is altered. Most DL models make this formulation (Equation 4.1) non-linear and non-convex, making it hard to find a closed-form solution [18]. Hence, we implement different techniques to find an approximate solution to this optimization problem.

Fast Gradient Sign Method (FGSM)

FGSM was suggested as an efficient attack method to fool the GoogLeNet model [112]. This method initially calculates the gradient of the cost function with respect to the input of the neural network. Adversarial examples are created based on a gradient direction:

$$\ddot{T} = T + \varepsilon * sign(\nabla_\tau L_f(T, R\hat{U}L)) \tag{4.2}$$

where ε denotes the amount of the perturbation.

Basic Iterative Method (BIM)

BIM is an extension of FGSM where FGSM is applied multiple times with really small step size [113]. At each iteration of the algorithm, BIM perturbs the original data in the direction

of the gradient multiplied by the step size α:

$$\ddot{T} = T + \alpha * sign(\nabla_{\tau} L_f(\ddot{T}, R\hat{U}L)) \tag{4.3}$$

where α is calculated by dividing the amount of perturbation by the number of iterations: $\alpha = \varepsilon/I$. Then, BIM clips the obtained time series elements to make sure that they are in the ε-neighborhood of the original time series:

$$\ddot{T} = min\{T + \varepsilon, max\{T - \varepsilon, \ddot{T}\}\} \tag{4.4}$$

Momentum Iterative Mehod (MIM)

MIM integrates momentum into the BIM to stabilize the update directions and to escape from poor local maxima [114]. At each iteration i, the variable g_i gathers the gradients with a decay factor μ:

$$g_{i+1} = \mu * g_i + \frac{\nabla_{\tau} L_f(\ddot{T}_i, R\hat{U}L)}{\|\nabla_{\tau} L_f(\ddot{T}_i, R\hat{U}L)\|_1} \tag{4.5}$$

where the gradient is normalized by the L_1 distance. Then, the perturbed data is generated in the direction of the sign of g_{i+1} with a step size α:

$$\ddot{T}_{i+1} = \ddot{T}_i + \alpha * sign(g_{i+1}) \tag{4.6}$$

In MIM, the algorithm also ensures that the crafted adversarial examples $\ddot{T}$ satisfy the L_∞ norm bound constraint:

$$\|\ddot{T} - T\|_\infty \leq \varepsilon$$

Robust Optimization Method (ROM)

The general goal in a supervised learning problem is to find model parameters θ that minimize the empirical risk

$\mathbb{E}_{(T,RUL)\sim\Xi}[L(T,RUL,\theta)]$ where Ξ is the underlying supervised data distribution. However, this formulation cannot handle data adversary properly. To solve that problem, set of allowed perturbations Δ is introduced initially.

Then, we modify the empirical risk formulation by feeding samples from the distribution Ξ directly into the loss L which leads to the following min-max optimization formulation [115]:

$$\min_{\theta} \zeta(\theta), \quad \textit{where } \zeta(\theta) = \mathbb{E}_{(T,RUL)\sim\Xi}[\max_{\delta\in\Delta} L(T+\delta,RUL,\theta)]. \tag{4.7}$$

Here, while inner maximization finds an adversarial version of a given data point T that achieves a high loss, outer minimization discovers model parameters to minimize the adversarial loss given by the inner attack problem. ROM replaces every instance with its FGSM-perturbed counterpart to solve this problem.

While all these four methods use the gradient information of the loss function, they modify the test data by adding different amounts of perturbation representing separate attack scenarios. An attacker, who is able to access the trained DL methods, can implement these methods and harm the prediction performance without being detected.

4.2.3 Adversarial Attacks in Predictive Maintenance

Adversarial attacks targeting predictive maintenance (PdM) applications can bring serious outcomes such as delayed maintenance/replacement of a machine [6]. There are few studies that analyze the impact of adversarial attacks on data-driven PdM. Mode et al. [107] focus on false data injection attack (FDIA) on PdM systems which alters the collected sensor data by a very small margin. They demonstrate the impact of different FDIA techniques on various DL methods e.g., gated recurrent unit (GRU), and convolutional neural network (CNN). Their results show that CNN is extremely sensitive to attacks while GRUs is the most resilient method. Their further work [6] analyze the effect of adversarial attacks against ML methods. Specifically, they utilize fast gradient sign method (FGSM) and basic iterative method (BIM) to create adversarial

examples and compare DL model performances under those attacks. They show that these attacks can cause up to 5× worse prediction performance. These two similar works consider limited number of DL methods and attack scenarios. Besides, their experimental analysis includes the simplest and most predictable data set from C-MAPSS. They also did not propose a novel ML solution to increase system resiliency against adversarial attacks.

4.2.4 Ensemble Methods

In PdM domain, there are multiple ensemble learning approaches towards more accurate predictions. Li et al. [63] combine multiple traditional ML methods using particle swarm optimization and sequential quadratic programming. Shi et al. [64] combine multiple traditional ML methods by using the most diverse base learners and features from different degradation stages. Gungor et al. [7] combine different DL methods by discovering the most accurate and diverse base learners iteratively (see Chapter 2 for details). Our ensemble approach combine different DL methods using a stacking ensemble to find the most resilient base learners.

Ensemble learners are especially useful to provide additional security against cyber-attacks since they can learn more robust features [116, 117]. Against adversarial attacks, different ensemble learners are proposed for image classification. Pang et al. [108] present a diversity promoting ensemble improving adversarial robustness while maintaining state-of-the-art accuracy. Mirzaeian et al. [109] propose a resilient ensemble where each member learns a radically distinct latent space through diverse knowledge distillation. To the best of our knowledge, our work is the first to use ensemble learning towards more resilient PdM. Our ensemble results are also more generalizable since we increase the number of attack scenarios, deep learning models, and experimental dataset significantly compared to the state-of-the-art.

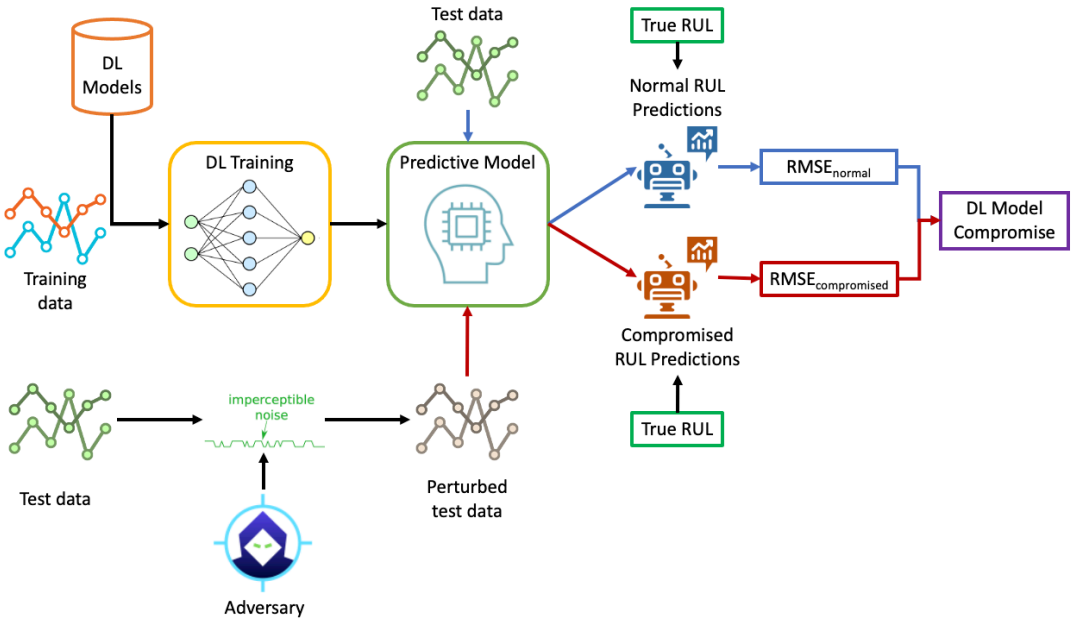


Figure 4.3. Framework for DL Methods Compromise Calculation

4.3 Proposed Framework

4.3.1 Deep Learning Methods Compromise Calculation

We select 10 different DL models from recurrent (RNN, LSTM, BLSTM, GRU, BGRU), convolutional (CNN, WAVE), and hybrid architectures (CLSTM, CGRU, GLSTM). With these 10 models, we cover a good range of DL methods from different architectures, increasing the generalizability of our study. To quantify the resiliency of DL models, we use our framework presented in Figure 4.3. The process starts with training 10 DL algorithms fed with training data. The trained models are then evaluated under two different test data sets: 1) normal, and 2) perturbed data. Adversary creates the perturbed data by adding imperceptible noise to the normal test data. This noise generation process is obtained by using one of the selected adversarial methods. Predictive models output two different remaining useful life (RUL) estimations: normal RUL predictions, and compromised RUL predictions. Given true RUL values, our error metric root mean squared error (RMSE) is calculated for both normal and compromised prediction scenarios based on the following formulation: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^2}$ where N is the number of

samples, ε is the difference between the estimated RUL (RUL_{est}) and the true RUL (RUL_{true}). Using these error values, we calculate the DL model compromise which is formulated as:

$$Compromise = \frac{RMSE_{compromised}}{RMSE_{normal}} \tag{4.8}$$

where $Compromise > 1$ (under the assumption that attacks lead to worse prediction performance). The smaller the compromise value, the more resilient the model is against the adversarial attack. For instance, given two methods CNN and LSTM, and their compromise values 8, and 5 respectively, we can conclude that LSTM is more resilient against the adversarial attack. If we have M number of adversarial attacks (where $M > 1$), then we need to calculate the mean compromise value for each DL method as follows:

$$Compromise_{mean} = \left(\sum_{i=1}^{M} \frac{RMSE^{i}_{compromised}}{RMSE_{normal}} \right) / M \tag{4.9}$$

Since we have multiple attack techniques, this metric gives a more accurate idea about single model resilience. Overall, we obtain mean compromise values for each DL model.

4.3.2 Stacking Ensemble Learner

Stacking (short for stacked generalization) is one of the most-used ensemble learning methods where single method predictions are aggregated using a second-level learner, or meta-learner [118]. Since our ensemble learner combines different DL model predictions, we select stacking as the most suitable ensemble approach.

Ensemble Learner Training

We present the general framework for stacking ensemble training in Figure 4.4. We start the training process by splitting training data into two subsets. We use the first subset (subset 1) to train the DL models. Here, for the sake of simplicity, the figure shows only two methods, namely CNN and RNN. After model training is completed, we obtain our predictive

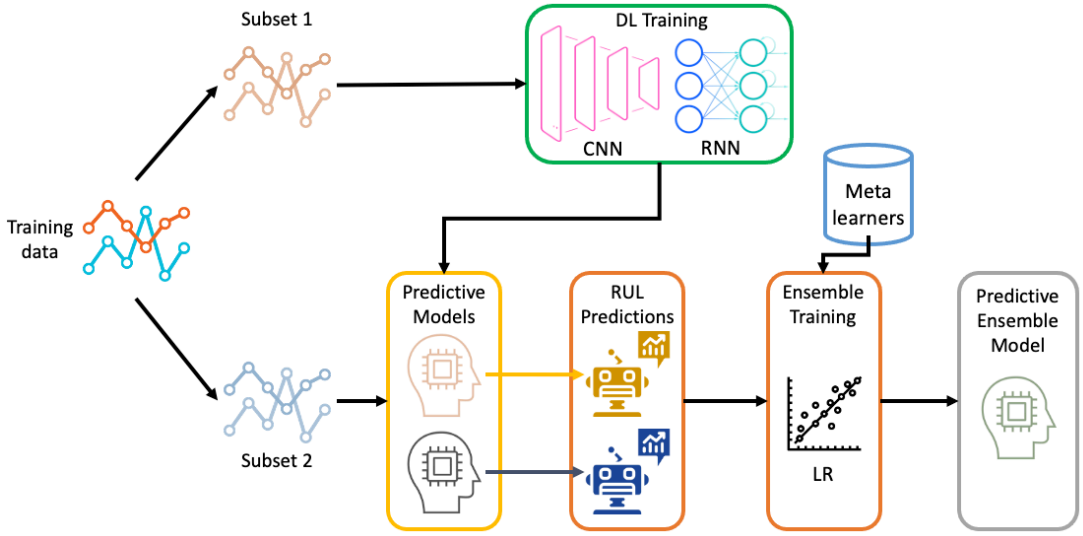


Figure 4.4. Framework for Stacking Ensemble Training

models. These models are used to make prediction on the subset 2 where each DL method outputs RUL predictions. These RUL predictions are given to the stacking ensemble training for which different meta-learners are trained. This training part is different from DL model training. In DL training, as an input we have time series data, yet in ensemble training, we have the RUL prediction values obtained from different DL methods. As an output of ensemble learner training, we obtain our predictive ensemble models. For illustration purposes, linear regression (LR) is used as the meta-learner to map single model RUL predictions to real RUL values in Figure 4.4. Overall, we train 4 meta-learners to find out the most resilient one against adversarial attacks:

1) Linear Regression (LR): This linear model makes a prediction by calculating a weighted sum of the input features, plus a bias term [118]: $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ where $\hat{y}$ indicates the predicted ensemble RUL value, n is the number of DL models, x_i is the ith DL model RUL prediction, θ_i $(\forall i = 1, \ldots, n)$ is the ith DL model weight, and θ_0 is the bias term.

2) Random Forest (RF): RF is an ensemble of decision trees trained by the bagging method [118]. The algorithm constructs multiple decision trees at training time and outputs the mean prediction of the individual trees.

3) AdaBoost: AdaBoost is one of the most famous boosting approaches where focus

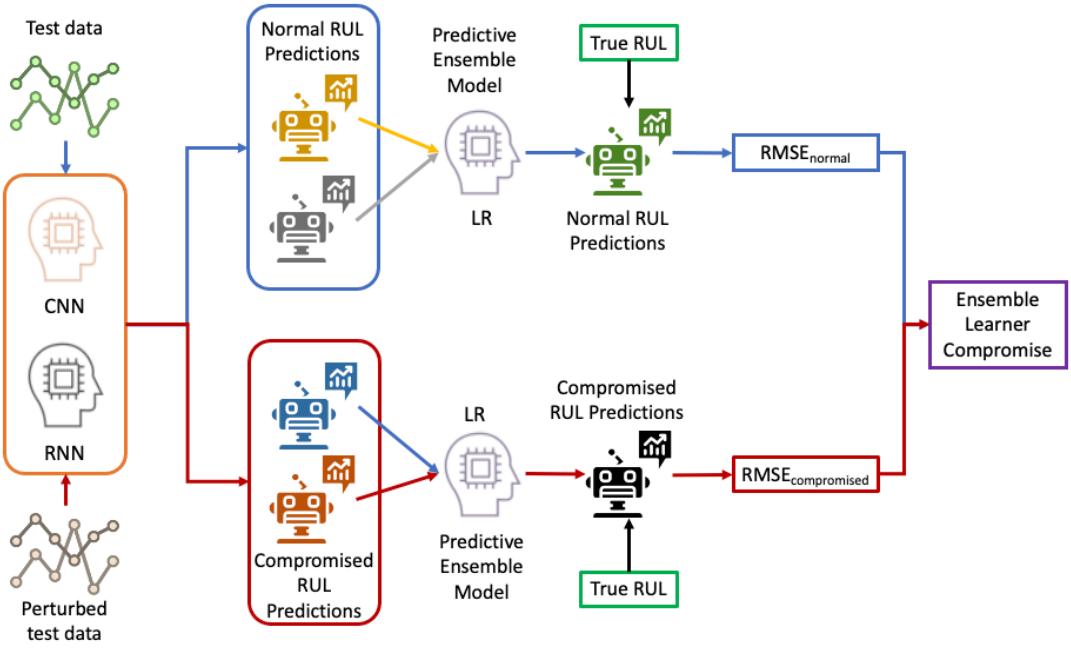


Figure 4.5. Framework for Stacking Ensemble Testing

is given to the training instances that the predecessor underfitted. The weights of instances are adjusted according to the error of the current prediction [118]. That is, subsequent estimators focus more on difficult cases.

4) Extreme Gradient Boosting (XGBoost): XGBoost is an an efficient and effective implementation of the gradient boosting algorithm. Gradient boosting differs from AdaBoost since it fits the new predictor to the residual errors made by the previous predictor [118]. The two main reasons why XGBoost is heavily used are execution speed and model performance.

Ensemble Learner Test

We test our stacking ensemble learner based on the framework provided in Figure 4.5. Similar to single DL model testing, we obtain the compromise value for our ensemble learner as an output. Given normal test data and perturbed test data (crafted by the adversary using adversarial attack methods described previously), pre-trained (predictive) DL models (e.g. CNN, RNN) make normal and compromised RUL predictions. These single method predictions are then given to our pre-trained (predictive) ensemble model (e.g. LR) to generate ensemble normal

and compromised RUL predictions. Similarly, the compromise value is calculated by dividing the compromised RMSE by the normal RMSE. Since we have multiple attack scenarios, we need to calculate mean compromise for our ensemble learner formulated in Equation 6.6. To show the benefit of our ensemble learner, we calculate the ensemble improvement over single method based on the following formulation:

$$Improvement = \left(\frac{Compromise_{single} - Compromise_{ensemble}}{Compromise_{single}} \right) \tag{4.10}$$

where $Compromise_{single}$ denotes the individual DL model mean compromise value, and ensemble mean compromise value is $Compromise_{ensemble}$. We report the improvement in percentage (%). Here, improvement demonstrates the resiliency of our ensemble learner against adversarial attacks compared to a single learner. The higher the improvement is, the more resilient our ensemble learner is compared to single DL model.

Most Resilient Stacking Ensemble Selection

To determine the most resilient ensemble learner configuration, we follow our procedure presented in Figure 4.6. This algorithm increases the number of base learners iteratively, and finds the most resilient ensemble configuration where resiliency can no longer be improved. Given single method mean compromise values C, the algorithm first sorts C in an ascending order. We launch the ensemble search with the 2 most resilient methods. We train the ensemble, test it, and calculate the ensemble mean compromise value using these two methods. The function that calculates ensemble mean compromise is also provided in Figure 4.7. This function first trains the ensemble learner given true RUL values and base learner RUL predictions. Then, it makes ensemble RUL predictions for both normal and perturbed test data. As an input, it uses single DL method normal and perturbed test data RUL predictions. The algorithm then calculates the RMSE for both normal and compromised scenarios using real RUL values and ensemble RUL predictions. It finally finds out the ensemble mean compromise. After we obtain ensemble

```
Input  : C = [C_1, C_2, ..., C_N] (single method mean compromise values)
Output: bestcompromise
ensemblecompromise = ∞, i = 2, worsenedcounter = 0, worsenedtolerance =
  2;
[C_sorted, I] = sort(C, ascend);
bestcompromise = C_sorted(1);
while i ≤ N do
    if i == 2 then
        model_indexes = I(1 : i);
        ensemblecompromise = calculateCompromise(..., model_indexes);
        if ensemblecompromise < bestcompromise then
            bestcompromise = ensemblecompromise
        end
    end
    else
        model_indexes = [model_indexes, I(i)];
        ensemblecompromise = calculateCompromise(..., model_indexes);
        if  ensemblecompromise < bestcompromise then
            bestcompromise = ensemblecompromise;
        end
        else
            worsenedcounter++;
            if worsenedcounter == worsenedtolerance then
                break;
            end
        end
    end
    i + + ;
end
return bestcompromise;
```

Figure 4.6. Most Resilient Stacking Ensemble Selection

compromise value, we check if this value is smaller than the single best method compromise and update the best compromise accordingly. We then continue with the next most resilient method selection and add this method to our base learner subset. For this new ensemble configuration,

Input : $RUL_{subset2}$ (true RUL values for training subset 2), RUL_{test} (true RUL values for test data), $\hat{RUL} = [\hat{RUL}_1, \hat{RUL}_2, \ldots, \hat{RUL}_N]$ (single method RUL predictions for training subset 2), $\tilde{RUL} = [\tilde{RUL}_1, \tilde{RUL}_2, \ldots, \tilde{RUL}_N]$ (single method RUL predictions for normal test data), $\ddot{RUL} = [\ddot{RUL}_1, \ddot{RUL}_2, \ldots, \ddot{RUL}_N]$ (single method RUL predictions for perturbed test data), N (number of base learners), M (number of attack methods), $model_{indexes}$

Output: meancompromise

meancompromisc, i = 0;
while $i < M$ **do**
    $ensemble = train_{ensemble}(\hat{RUL}(model_{indexes}), RUL_{subset2})$;
    $\tilde{RUL}_{ensemble} = test_{ensemble}(\tilde{RUL})$;
    $\ddot{RUL}_{ensemble} = test_{ensemble}(\ddot{RUL})$;
    $RMSE_{normal} = calculate_{ensemble}(RUL_{test}, \tilde{RUL}_{ensemble})$;
    $RMSE_{compromised} = calculate_{ensemble}(RUL_{test}, \ddot{RUL}_{ensemble})$;
    ensemblecompromise = $\frac{RMSE_{compromised}}{RMSE_{normal}}$;
    meancompromise += ensemblecompromise;
    $i++$;
end
meancompromise /= M;
return meancompromise;

Figure 4.7. Calculate Compromise

we calculate its compromise value and update the best compromise if it improves the best compromise value. If there is no improvement, we increment the variable *worsenedcounter*. This variable controls whether we should continue or terminate the ensemble search process. We allow only a fixed number of iterations with performance decrease, *worsenedtolerance*, after which we terminate the search process and return the best compromise value.

4.4 Experimental Analysis

4.4.1 Dataset Description

To validate the resiliency of our proposed ensemble learner framework against adversarial attacks, we use two different datasets: NASA C-MAPSS [49], and UNIBO Powertools [111].

NASA C-MAPSS is a benchmark dataset for RUL estimation. It includes multiple aircraft engines simulated under various operating and fault conditions. It includes 4 different datasets in increasing complexity: FD001∼FD004. For each dataset, we have separate training and test data where the goal is to predict RUL for the test data. Our feature columns include the engine ID, cycle index, three operational settings, and 21 sensor measurements.

UNIBO Powertools is a lithium-ion (Li-Ion) battery dataset collected in a laboratory test by an Italian Equipment producer [111]. It contains 27 batteries which are run until their end of life. We use 17 of these batteries for training and 10 of them for testing. These batteries have different nominal capacities and they are tested under different conditions: 1) standard test: battery was discharged at 5A current in main cycles, 2) high current test: battery was discharged at 8A current in main cycles, 3) preconditioned test: battery cells are stored at 45°C environment for 90 days before conducting the test. The following procedure is used to create the dataset where during discharge, the sampling period is set to 10 seconds [111]: 1) Charge cycle: Constant Current-Constant Voltage (CC-CV) at 1.8A and 4.2V (100mA cut-off), 2) Discharge cycle: Constant Current until cut-off voltage (2.5V), 3) Repeat steps 1 and 2 (main cycle) 100 times, 4) Capacity measurement: charge CC-CV 1A 4.2V (100mA cut-off) and discharge CC 0.1A 2.5V, 5) Repeat the previous steps until the battery cell end of life. We have different columns in this dataset: battery id, time, voltage, current, charging capacity, discharging capacity, watt hour (wh) measurements during charge and discharge, temperature, and cycle count.

4.4.2 Experimental Setup

Adversarial Attack Methods

We use the following parameters for the selected adversarial methods [119, 114, 115]: amount of perturbation(ε)=0.1, step size(α)=0.001, number of iterations(I)=100.

Deep Learning Methods

Although we use the same model structures for both datasets, we select different hyper-parameters to run the models so as to obtain the best possible performance. We replicate each experiment 10 times and report average **compromise** values where we run all experiments on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor.

NASA C-MAPSS: *Adam* optimizer with learning rate 0.001, *elu* activation function, batch size of 128, and a max number of epochs of 150 where callback is activated (patience is set to 10 for validation data), and sliding time window size of 80.

UNIBO: *Adam* optimizer with learning rate 0.0001, *selu* activation function, batch size of 256, and a max number of epochs of 100 where callback is activated (patience is set to 10 for validation data), and sliding time window size of 500.

Stacking Ensemble

For the select meta-learners, we perform hyper-parameter optimization using a grid search [120]. This gives us optimal hyper-parameters to combine predictions from different DL models using stacking ensemble. For the ensemble training, we split training data into two subsets using the ratio 70% (subset 1) to 30% (subset 2). We use subset 1 for DL model training, and subset 2 for ensemble training. We set *worsenedtolerance* to 2 since it leads to the selection of optimal ensemble configuration.

Table 4.1. Single DL Models Mean Compromise

DL Model / Dataset	FD001	FD002	FD003	FD004	UNIBO
CLSTM	8.0	120.3	6.8	86.2	27.2
CNN	20.6	72.0	13.5	12.5	22.6
WAVE	17.6	25.6	14.4	5.6	7.2
CGRU	9.0	13.6	7.8	6.5	32.8
BLSTM	6.7	8.4	8.7	6.0	10.5
GLSTM	6.4	8.4	7.9	6.2	6.8
BGRU	6.1	7.4	7.5	5.9	7.5
LSTM	5.7	7.9	7.2	5.4	6.3
GRU	**5.2**	7.7	6.5	7.1	**4.5**
RNN	5.3	**4.3**	**5.0**	**4.6**	7.1

4.4.3 Single DL Models Resiliency

Table 4.1 presents mean compromise values for each DL method. In this table, each row represents a different DL model and each column corresponds to a distinct dataset. We first observe that DL model prediction performance is impacted poorly by the adversarial attacks where there is up-to 120$\times$ compromise. We also notice that the resiliency of a DL method changes with respect to the dataset. Here, we present the most resilient methods at the bottom of the table. We observe that GRU is the most resilient algorithm at FD001, and UNIBO while RNN is the best at the remaining datasets. We can conclude that recurrent architectures, e.g., GRU, RNN, are superior over others. CNN-based methods are extremely sensitive to the adversarial attacks where the prediction performance degrades by up to 72$\times$. Hybrid methods can be resilient if solely recurrent architectures are combined, e.g., GLSTM. For the most resilient ensemble selection, we utilize the compromise values provided in Table 4.1.

4.4.4 Proposed Stacking Ensemble Learner Resiliency

Meta-learner Resiliency Analysis

We first analyze the meta-learner resiliency of our ensemble learner. Figure 4.8 illustrates the meta-learner mean compromise values where each meta-learner is represented with a distinct

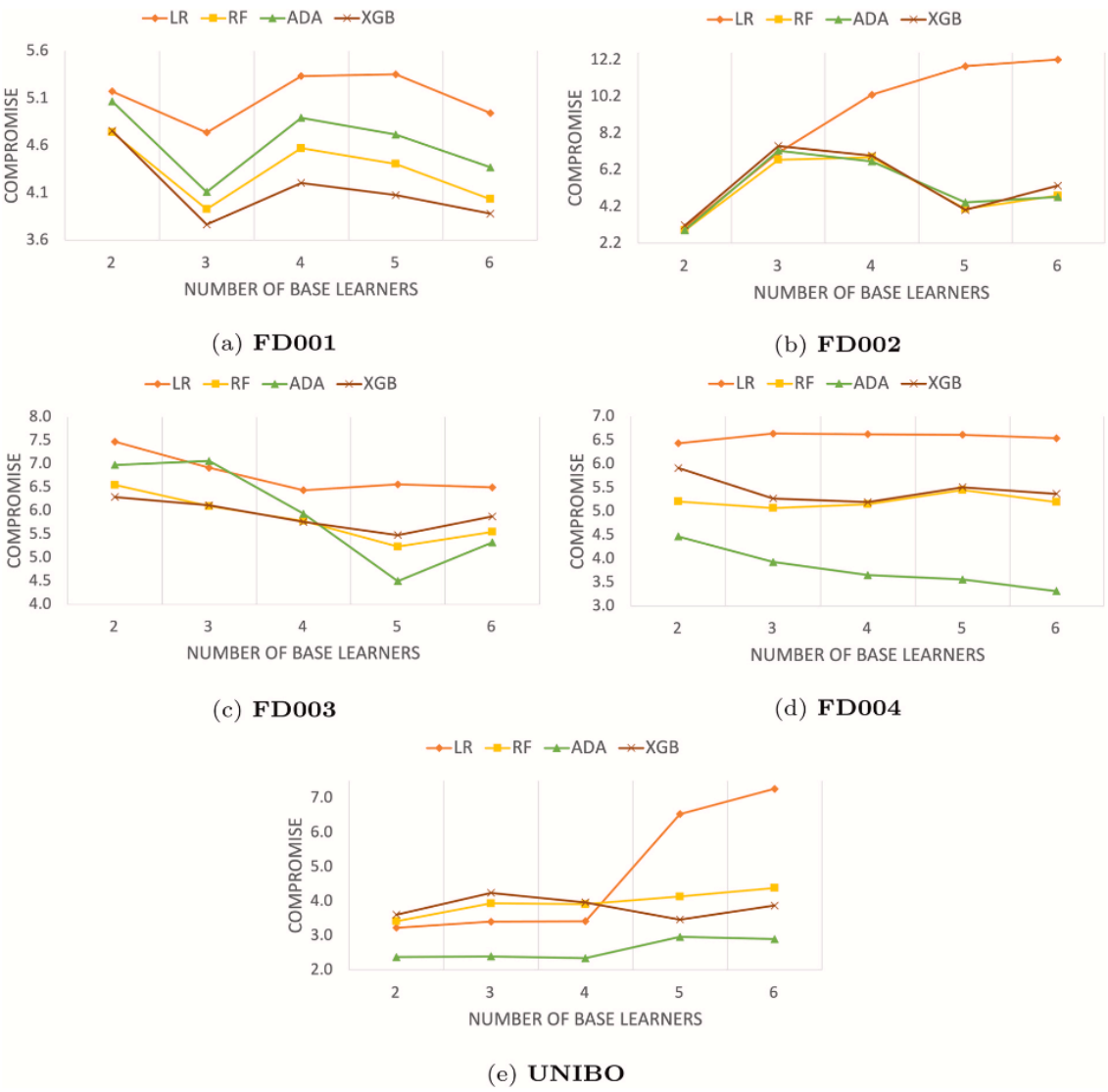


Figure 4.8. Meta-learner Compromise Analysis

color. In each sub-figure, x-axis shows the number of base learners, and the y-axis provides the ensemble compromise. We first note that the meta-learner resiliency fluctuates considerably with respect to the number of base learners. To illustrate, at FD003, AdaBoost (ADA) is the most resilient at 5 and 6 base learner ensemble scenarios, yet it is the worst if we only select 3 base learners. The best performing meta-learner also changes based on the number of base learners. However, this is not the case for all datasets. For instance, ADA is always the most resilient meta-learner at FD004, and UNIBO. When we analyze the average performance of each

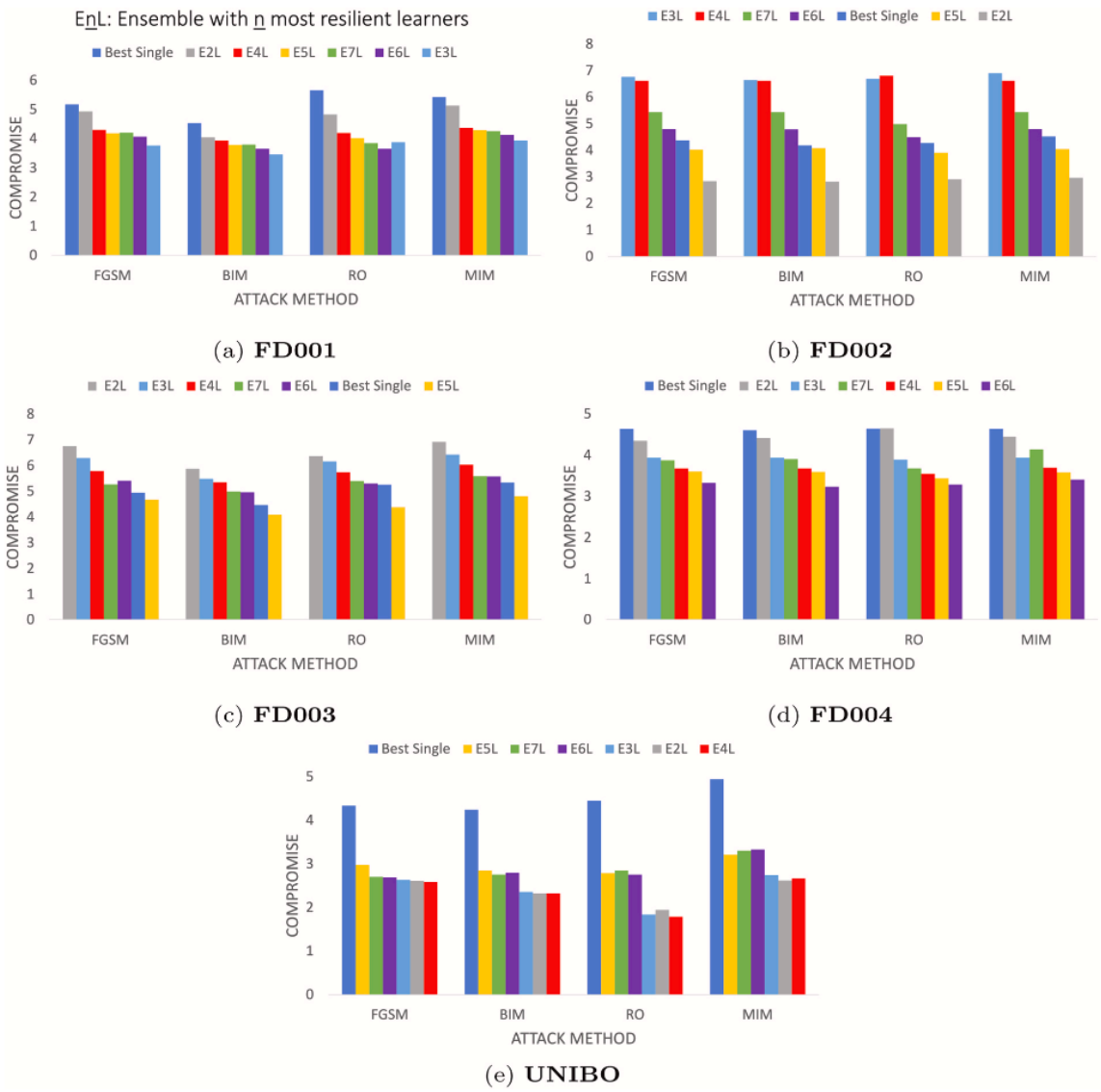


Figure 4.9. Stacking Ensemble Compromise Analysis

meta-learner over all datasets and ensemble learner configurations, we obtain 6.43, 4.91, 4.88, and 4.47 compromise values for LR, RF, XGB, and ADA respectively. This shows that ADA is the most resilient meta-learner while LR being the least resilient (on average). For the rest of our ensemble analysis, we select the best meta-learner for each ensemble configuration and report those measurements. To exemplify, for UNIBO dataset and any ensemble configuration, we present the ADA compromise values since its value is the smallest. However, ADA is not selected for any ensemble configuration at FD001.

Stacking Ensemble Analysis

We first analyze the resiliency of ensemble learners having different number of base learners. Figure 4.9 demonstrates a variety of stacking ensemble learner compromise values. In each sub-figure, x-axis shows the attack method, the y-axis denotes the ensemble compromise. Each figure shows the best single method and multiple ensemble learner configurations for different attack methods. Note that in these figures, EnL represents our ensemble learner with n most resilient learners. We consider different number of base learners (from 2 to 7, e.g., 'E2L' uses 2 most resilient base learners) and the most resilient single method ('Best Single'). All methods in each figure are represented with distinct colors and the legend of each figure shows the order in which these methods are presented. Each ensemble compromise value in this figure corresponds to the compromise value of the best meta-learner. We can find the most resilient ensemble configuration from Figure 4.9, on the right most bar in each sub-figure. For instance, at FD003, E5L (represented with yellow color) is the most resilient configuration. E3L, E2L, E5L, E6L, and E4L are the most resilient ensemble configuration for FD001, FD002, FD003, FD004, and UNIBO respectively. Besides, we observe that increasing the number of base learners does not always lead to more resilient learner. To illustrate, the best performing ensemble at FD002 only uses 2 base learners. This result motivates us for a more clever ensemble method selection approach which can both terminate the search process early (i.e., it might not be necessary to try all base learners) and can find the most resilient ensemble configuration.

Adversarial Attacks Compromise Analysis

Based on the results in Figure 4.9, we also analyze the impact of an adversarial attack on the model compromise. Figure 4.10 shows the average compromise values for each attack method. On the y-axis, we calculate the average compromise over all ensemble and the best single method scenarios, x-axis corresponds to the dataset. Momentum iterative method (MIM) leads to highest compromise (up to 5.8×) whereas BIM is the least strong attack among all.

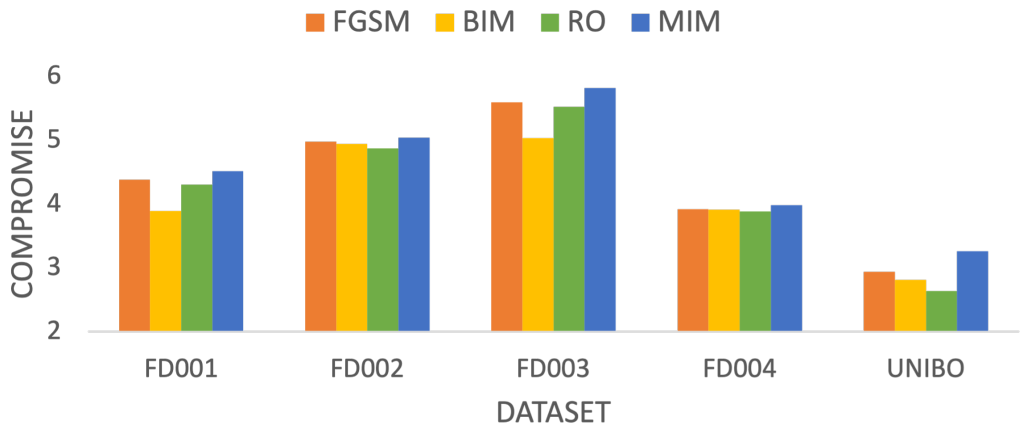


Figure 4.10. Adversarial Attacks Compromise Analysis

Most Resilient Stacking Ensemble Selection

Table 4.2 presents the results of the most resilient ensemble configuration search process. In this table, the columns show the dataset, best ensemble configuration, mean ensemble compromise, best single method mean compromise, and the ensemble resiliency improvement over the single method respectively. We can observe that the best ensemble configuration is different at each dataset, e.g., E3L for FD001, E2L for FD002, and so on. We previously found out the most resilient ensemble configurations. We can validate that our proposed algorithm is able to select those ensemble configurations successfully. While we obtain the best ensemble mean compromise at UNIBO (2.34×), the smallest single method compromise is obtained at FD002 (4.34×). Our stacking ensemble approach achieves up to 47.9% mean compromise improvement. We also analyze the proposed stacking ensemble compromise improvement for the adversarial attack methods individually. Table 4.3 shows our proposed ensemble method's resiliency improvement over the best single method under each attack scenario. We reach up-to 59.9% improvement at UNIBO. For FD001, FD002, FD003, and FD004, the maximum improvements are 31.5%, 35%, 16.5%, and 29.8% respectively.

Table 4.2. Most Resilient Stacking Ensemble Configuration

Dataset	Ensemble Configuration	Compromise	Best Single Compromise	Improvement (%)
FD001	3 Learners (E3L)	3.76	5.21	27.83%
FD002	2 Learners (E2L)	2.88	**4.34**	33.64%
FD003	5 Learners (E5L)	4.49	5.00	10.20%
FD004	6 Learners (E6L)	3.32	4.63	28.29%
UNIBO	4 Learners (E4L)	**2.34**	4.49	**47.88%**

Table 4.3. Proposed Stacking Ensemble Compromise Improvement Over the Most Resilient DL Method (%)

Dataset / Attack Method	FGSM	BIM	RO	MIM
FD001	27.4	23.6	**31.5**	27.5
FD002	**35.0**	32.6	32.0	34.5
FD003	5.3	8.4	**16.5**	10.1
FD004	28.2	**29.8**	29.2	26.6
UNIBO	40.4	45.3	**59.9**	46.0

4.5 Conclusion

We propose a stacking ensemble learning framework which is more resilient against adversarial attacks compared to single DL methods. We use different adversarial attacks and 10 distinct DL models from recurrent, convolutional, and hybrid architectures. We find that recurrent neural network based architectures provide more resilient learning whereas convolutional neural network structures are extremely sensitive to the adversarial attacks. We observe that the most resilient single ML method changes based on the data set or attack method. To address this issue, we propose a framework that finds the most resilient ensemble configuration against multiple attacks. The results show that our proposed ensemble learner framework can improve the resiliency of the most resilient single method by up to 60%. From a research perspective, this means that the proposed ensemble solution can still perform well under adversarial attacks. At the management level, this leads to more accurate replacement and maintenance decisions even under cyber-attacks.

In this chapter, we addressed the decreasing prediction performance of DL models under

adversarial attacks. Although we devised a resilient learning solution, we still need a defense mechanism that can be used for any DL model. In the following chapter, we propose a novel adversarial training defense against adversarial attacks.

4.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1911095, and #1952225.

Chapter 4, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "STEWART: stacking ensemble for white-box adversarial attacks towards more resilient data-driven predictive maintenance". *Computers in Industry*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 5

Diversity Promoting Ensemble Adversarial Training

5.1 Introduction

Data-driven remaining useful life (RUL) estimation utilizes sensor data to build machine learning (ML) models. Recently, this approach became popular with abundance of available sensor data where sensor data collection and processing plays a crucial role to achieving good prediction performance [7, 121]. Furthermore, performance of ML methods relies heavily on input sensor data quality. Thus, these methods are vulnerable to adversarial attacks where an attacker can modify input data or model parameters, significantly worsening ML prediction performance [122]. Since ML is in the center of data-driven RUL prediction, these attacks may lead to wrong maintenance decisions causing undetected failures in a system [6]. Hence, there is a need for effective defense mechanisms that can minimize the impact of adversarial attacks in the RUL prediction domain.

This chapter proposes diversity promoting ensemble adversarial training framework as a defense mechanism. To the best of our knowledge, our work is the first that proposes ensemble adversarial training towards more resilient data-driven predictive maintenance. Given 10 different pre-trained deep learning (DL) methods, we first calculate pairwise loss gradient similarity. Based on the similarity values, we select the most dissimilar subset of methods. We then create perturbed training examples based on the selected methods where we use Fast

Gradient Sign Method [112]. These crafted examples are augmented with the clean, i.e., non-perturbed, training data. Given augmented training data, we train a convolutional neural network (CNN) [55] because of its high accuracy in RUL prediction. In testing, we first create perturbed test instances using our trained CNN based on different adversarial attacks. These instances are then transferred to pre-trained DL methods to measure the performance change after adversarial attacks which we refer as resiliency. The less the performance change is, the more resilient a method is. We compare our approach with two non-adversarial state-of-the-art training settings. Our experiments on NASA C-MAPSS dataset [49] show that the proposed ensemble training approach can improve the learner resiliency by up to 97% (43% on average).

5.2 Related Work

Adversarial training is one of the most effective defense approaches against adversarial attacks [36]. It augments training data with adversarial examples in each training iteration. However, this approach converges to a degenerate global minimum [123]. To solve this problem, ensemble adversarial training (EAT) is introduced by Tramer et al. [123] where training data is augmented with adversarial examples generated from different target models. EAT provides a better defense mechanism since it is harder for the attacker to trick multiple models in the ensemble instead of just a single model. To obtain ensemble robustness against adversarial attacks, the base learners should be diverse [124]. There are different methods proposed in the literature that promote diversity in ensemble adversarial training [108, 125, 124, 126]. Yang et al. [126] theoretically show that promoting the orthogonality between gradients of base models leads to higher robustness. Inspired by this work, we promote diversity based on loss gradient similarity among base learners.

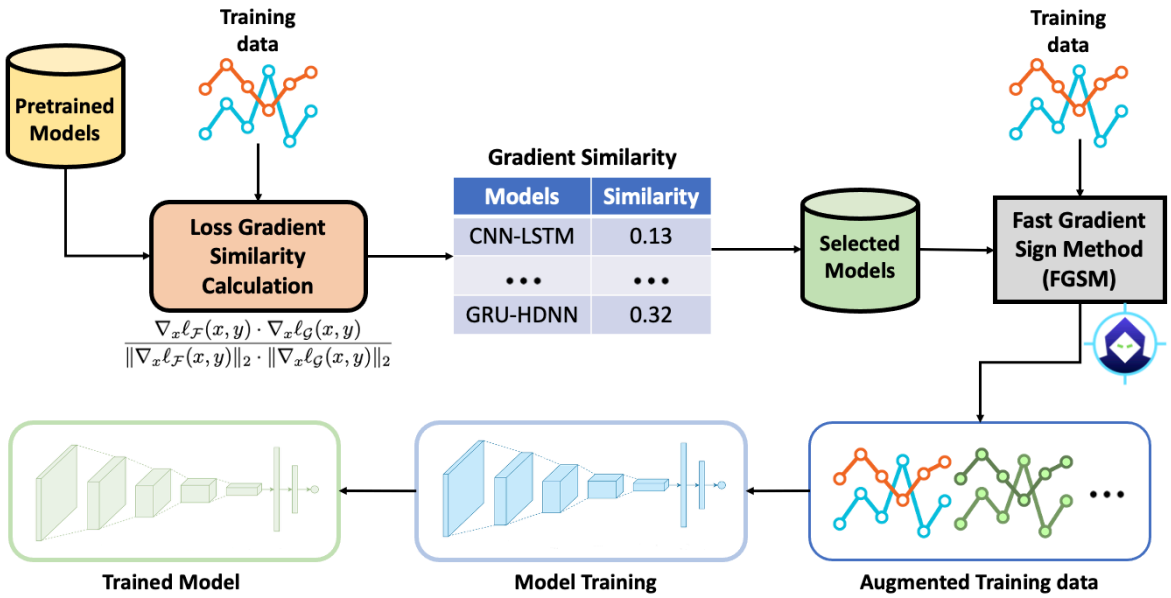


Figure 5.1. Our Proposed Framework

5.3 Proposed Framework

Figure 5.1 depicts our proposed ensemble adversarial training framework. Given pre-trained DL models, we first calculate the loss gradient similarity among learners and select the most dissimilar ones. Using the selected learners and fast gradient sign method, perturbed training examples are generated and augmented to the training data. Then, we train a convolutional neural network (CNN) [55] using the augmented training data. As the output of this framework, we obtain the trained CNN model. Next, we explain the steps of our framework in detail:

5.3.1 Pre-trained Models

We used 10 different pre-trained deep learning models from our previous study [122]: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bi-directional LSTM (BLSTM), Gated Recurrent Unit (GRU), Bi-directional GRU (BGRU), Convolutional Neural Network (CNN), Wavenet (WAVE), CNN-LSTM (CLSTM), CNN-GRU (CGRU), GRU-LSTM (GLSTM). We cover a good range of DL methods, increasing the generalizability of our study.

5.3.2 Loss Gradient Similarity Calculation

To introduce diversity into ensemble adversarial training, we measure pairwise loss gradient similarity among two different pre-trained models (F and G) based on the following formulation [126]:

$$\left| \frac{(\nabla_x L_F)^T (\nabla_x L_G)}{\|(\nabla_x L_F)\|_2 \cdot \|(\nabla_x L_G)\|_2} \right| \tag{5.1}$$

where $\nabla_x L_F$ and $\nabla_x L_G$ denote the loss gradient vectors of base models F and G on input x. Note that Equation 5.1 is the absolute value of cosine similarity between the gradients of the two loss functions. As a result of this step, we obtain the gradient similarity table as illustrated in Figure 5.1. The smaller the similarity is, the more diverse the two models are. We then select the models which are least similar to our pre-trained CNN since this model structure will be used in adversarial training. We increment the number of models (thus augmented data size) until no further resiliency improvement is observed.

5.3.3 Augmented Training Data Generation and Training

Based on the selected models from the previous step, we generate perturbed training examples using fast gradient sign method (FGSM) [112]. FGSM first calculates the gradient of the cost function with respect to the input of the neural network. Adversarial examples are then created based on the gradient direction: $\ddot{x} = x + \varepsilon * sign(\nabla_x L(\theta, x, y))$ where $\ddot{x}$ represents the crafted adversarial examples and ε denotes the amount of the perturbation. We select FGSM since it can create adversarial examples efficiently [127]. These crafted examples are then included in the training data. Given augmented training data, we train convolutional neural network (CNN) [55] due to its high prediction accuracy in RUL prediction. As an output, we obtain the adversarially trained CNN model.

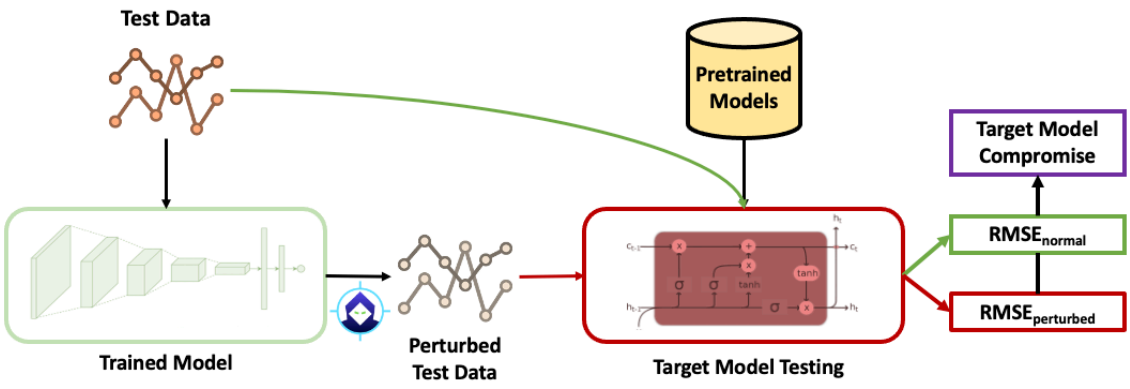


Figure 5.2. Testing Framework

5.3.4 Testing Framework

Figure 5.2 shows our testing framework where we adapt a transfer black-box attack strategy [17]. Given the test data, we first create perturbed samples using our trained model (CNN) based on three different adversarial attacks: fast gradient sign method (FGSM) [112], basic iterative method (BIM) [116], and momentum iterative method (MIM) [114]. We then transfer these instances to our pre-trained models. We measure pre-trained models' prediction performance before ($RMSE_{normal}$) and after ($RMSE_{perturbed}$) the adversarial attacks where $RMSE$ refers to root mean squared error. To measure the prediction performance change, we define a metric called $Compromise_{mean}$ formulated as:

$$Compromise_{mean} = \left(\sum_{i=1}^{M} \frac{RMSE^{i}_{perturbed}}{RMSEnormal} \right) / M \tag{5.2}$$

where $Compromise_{mean} > 1$ (with the assumption that attacks lead to worse prediction performance) and M denotes the number of adversarial attacks (i.e., $M = 3$). The smaller the mean compromise is, the more resilient the model becomes against adversarial attacks.

5.3.5 Compared Training Settings

We compare the resiliency of the proposed method with two different non-adversarial training settings which directly use the pre-trained models: (i) white-box setting creates perturbed

test instances using a pre-trained model and use them in the same model testing, i.e., no test example transfer across different models. For instance, only pre-trained LSTM creates perturbed test instances to be used in LSTM resiliency measurement, (ii) black-box setting creates perturbed test examples only using the pre-trained CNN model and transfers the examples to other pre-trained models at testing time. This setting is similar to our testing strategy, yet it does not include adversarial training.

5.4 Experimental Analysis

5.4.1 Dataset Description

We use NASA C-MAPSS [49] which is a benchmark dataset for RUL estimation. This dataset includes multiple aircraft engines simulated under different operating and fault conditions. We select the FD002 dataset which is one of the most complicated (i.e., the highest number of operating and fault conditions) datasets in C-MAPSS.

5.4.2 Experimental Setup

We use the following parameters for the selected adversarial attacks [119, 114]: amount of perturbation (ε) = 0.1, step size (α) = 0.001, number of iterations (I) = 100, decay factor (μ) = 1. For the DL model training, we use *Adam* optimizer with learning rate 0.001, *elu* activation function, batch size of 128, and a max number of epochs of 150, and sliding time window size of 80. We repeat each experiment 10 times and report average values. All experiments are run on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor.

5.4.3 Impact of Number of Base Learners in Resiliency

For our proposed method, we experiment with different number of diverse base learners and measure their *mean compromise* across all pre-trained DL models to determine *DENSE-DEFENSE* optimal configuration. Figure 5.3 shows mean compromise values (y-axis) across each DL method (x-axis). We can observe that switching from 2 to 3 learners increase the

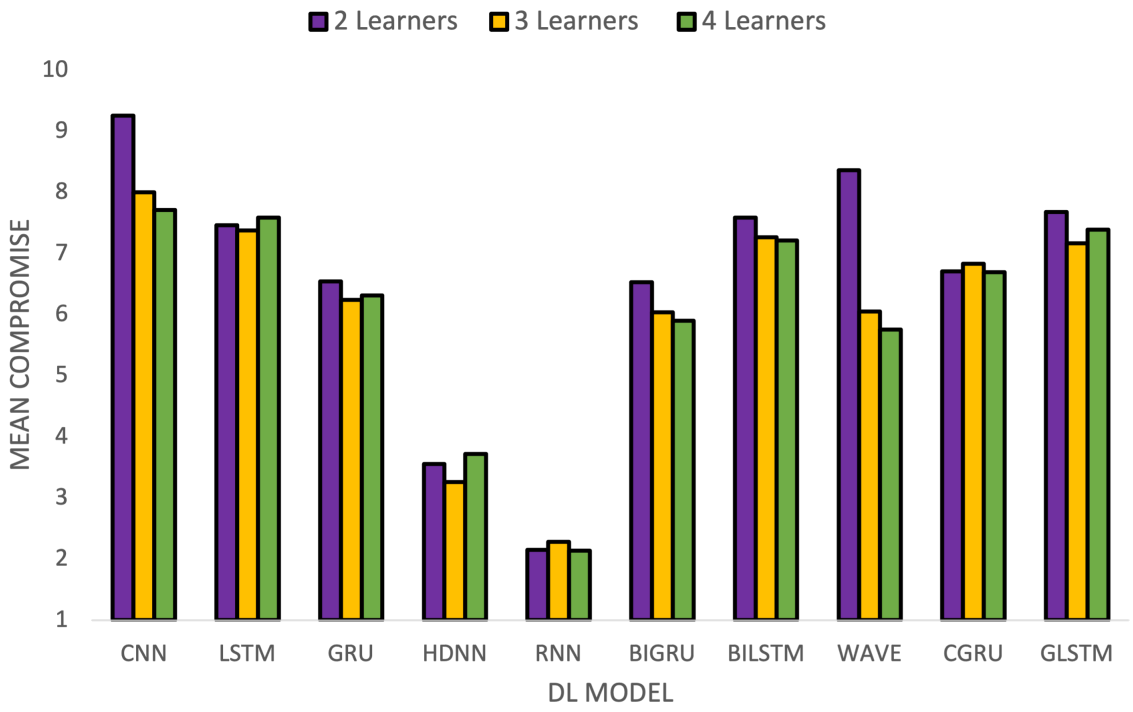


Figure 5.3. Impact of Number of Base Learners in Resiliency

resiliency of the proposed method. However, adding more models after 3 learner ensemble does not bring a significant resiliency benefit (for some models, it even decreases the resiliency). Specifically, these ensemble configurations (2 learners, 3 learners, and 4 learners) have 6.58, 6.05, and 6.04 average compromise values across all models.

5.4.4 Mean Compromise Comparison

After we selected the optimal configuration for *DENSE-DEFENSE*, we compare our approach with two other training settings (white-box and black-box). Table 5.1 shows the mean compromise values for the 3 selected settings: white-box, black-box and our method *DENSE-DEFENSE*. We can observe that for all DL methods, our approach provides the highest resiliency, showing the superiority of our approach.

Based on the values in Table 5.1, we also calculate our method's improvement over the white-box and black-box settings. Table 5.2 presents the *DENSE-DEFENSE* resiliency improvement over the selected approaches. Compared to white-box and black-box settings, our

Table 5.1. Mean Compromise Comparison

DL Model / Approach	White-box	Black-box	*DENSE-DEFENSE*
CNN	72.31	72.31	**7.71**
LSTM	7.83	8.46	**7.59**
GRU	7.50	7.87	**6.31**
HDNN	122.50	87.25	**3.72**
RNN	4.36	3.77	**2.13**
BIGRU	7.22	6.67	**5.90**
BILSTM	8.30	8.76	**7.21**
WAVE	25.59	24.45	**5.75**
CGRU	13.19	11.79	**6.69**
GLSTM	8.40	9.17	**7.39**

Table 5.2. *DENSE-DEFENSE* Resiliency Improvement

DL Model / Improvement (%)	White-box	Black-box
CNN	89.34	89.34
LSTM	3.13	10.28
GRU	15.83	19.72
HDNN	96.96	95.74
RNN	51.09	43.42
BIGRU	18.38	11.56
BILSTM	13.11	17.66
WAVE	77.52	76.47
CGRU	49.25	43.24
GLSTM	11.98	19.42
Average	**42.66**	**42.69**
Maximum	**96.96**	**95.74**

method improves the resiliency by up-to 96.9% and 95.7% respectively. For both approaches, we obtain 43% average resiliency improvement. The results show that our method provides a more resilient learning solution. Hence, our ensemble training approach is an effective defense mechanism against different adversarial attacks.

5.5 Conclusion

ML methods are impacted significantly by small perturbations in input data. Hence, adversarial attacks against ML methods can lead to bad outcomes for predictive maintenance

applications. To provide one possible defense against those attacks, in this chapter we propose diversity promoting ensemble adversarial training where selected diverse base learners' perturbed instances are included in the training process. Our experiments show that our method can be a really effective defense mechanism against different adversarial attacks where we improve the resiliency by up to 97% (43% on average) compared to state-of-the-art training approaches.

In this chapter, we focused on creating an effective defense mechanism against adversarial attacks. In this thesis, we focused on deep learning (DL) models so far, yet IIoTs resource constrained devices might not be suitable for DL training and inference. To address this challenge, we utilize a novel learning paradigm hyper-dimensional computing (HDC) and investigate its resiliency against adversarial attacks in the rest of this thesis.

5.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1911095, #1952225, #2003277, and #2003279.

Chapter 5, in part, is a reprint of the material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "DENSE-DEFENSE: Diversity Promoting Ensemble Adversarial Training Towards Effective Defense". *IEEE SENSORS*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 6

Hyperdimensional Computing for Resilient IIoT Predictive Analytics

6.1 Introduction

Abundant system monitoring data in IIoT systems makes data-driven predictive maintenance (PdM) popular where machine learning (ML) is used for identifying best maintenance schedules [100]. Intelligent fault diagnosis (IFD) is a key data-driven PdM application that finds and classifies different fault types before they occur. There are variety of IFD methods proposed in the literature such as convolutional neural network (CNN) [86], long short-term memory (LSTM) [128], gated recurrent unit (GRU) [129], ensemble learning [121], and so on. The success of these ML-based methods heavily depends on input data. Adversarial attacks against ML methods manipulate legitimate inputs and force the trained model to produce incorrect outputs leading to incorrect predictions. Since ML is in the center of intelligent fault diagnosis, these attacks may have serious consequences such as undetected failures [6]. Hence, there is a need for novel intelligent learning solutions that can stay resilient against various adversarial attacks.

In this work, we propose hyperdimensional computing (HDC) as a resilient learning solution against different black-box adversarial attacks for intelligent fault diagnosis (IFD). Our black-box attack is based on a transferable attack strategy [130]. We first train a substitute deep learning model, a wide deep convolutional neural network (WDCNN), and create artificial test

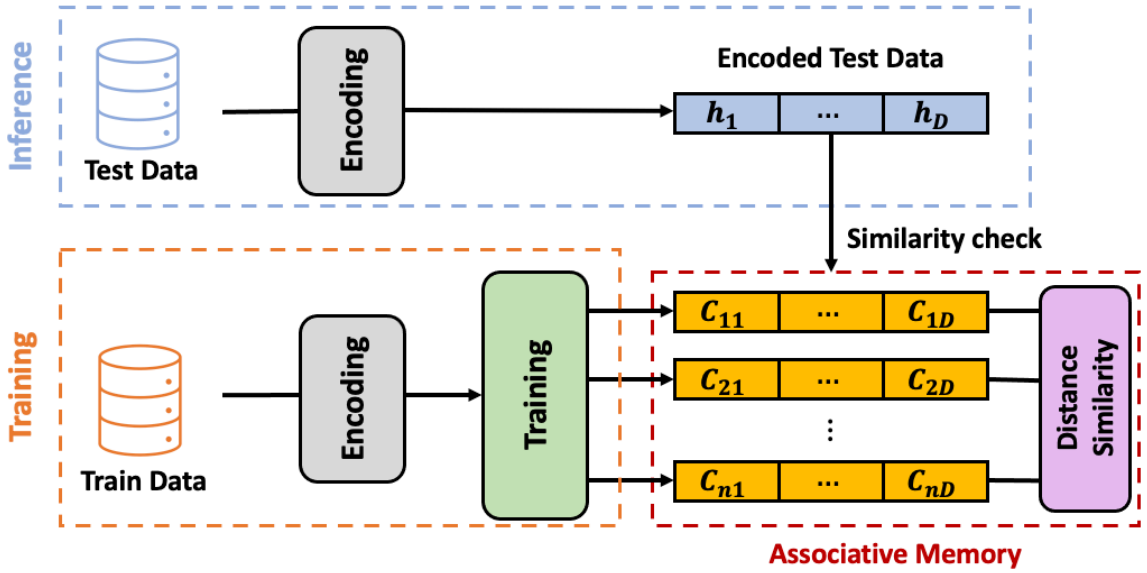


Figure 6.1. HDC Learning Framework

samples using this trained model. We then transfer these instances to the target methods (e.g., LSTM, GRU, HD). In testing, we measure the classification accuracy of the target models before and after the attacks. The accuracy change gives us the resiliency of a target method. We show that HD is the most resilient and lightweight method, outperforming all deep learning (DL) methods on commonly used CWRU Bearing dataset [88]. HD is up to 67.5% more resilient and 25.1× faster during training compared to the state-of-the-art DL methods.

6.2 Background and Related Work

6.2.1 HDC Background

HDC has 3 main parts, encoding, training, and inference as illustrated in Figure 6.1:

Encoding aims to map input data to hypervectors (HVs). Most of the proposed encoding methods [131] linearly combine HVs corresponding to each feature, resulting in sub-optimal classification quality [132]. In this chapter, we use non-linear encoding which considers the non-linear interactions between the feature values with different weights and exploits the kernel trick. This encoding approach is based on a study which shows that the Gaussian kernel

function can be approximated by the dot product of two vectors. Assume an input vector in original space $\vec{\Xi} = \{\xi_1, \xi_2, \ldots \xi_n\} \in \mathbb{R}^n$. The encoded high-dimensional vector is represented as $\vec{H} = \{h_1, h_2, \ldots, h_D\} \in \mathbb{R}^D$ where $D \gg n$. Encoding from $\vec{\Xi}$ to h_i is as follows:

$$h_i = cos(\vec{\Xi} \cdot \vec{B}_i + b_i) sin(\vec{\Xi} \cdot \vec{B}_i) \tag{6.1}$$

where $\vec{B_k}$s are randomly chosen of dimension $D \simeq 10k$ and $b_i \sim U(0, 2\pi)$. That is, $\vec{B}_{kj} \sim N(0,1)$ and $\delta(\vec{B}_{k1}, \vec{B}_{k2}) \simeq 0$, where δ is the cosine similarity.

Training has two steps to generate HVs representing each class. The first step, **initial training**, performs element-wise addition of all encoded hyper-vectors in each existing class. Let's assume that $\vec{H}_i$ is the encoded hyper-vector of input i. We know that each input i belongs to a class j. Hence, $\vec{H}_i^j$ denotes the hyper-vector for input i from class j. In the initial training, HD simply adds all hyper-vectors of the same class to generate the final model hyper-vector:

$$\vec{C}^j = \eta \vec{H}_0^j + \eta \vec{H}_1^j + \cdots = \sum_m \eta \vec{H}_m^j \tag{6.2}$$

where η is the learning rate. This process is also called as one-pass training since each input is visited only once. The second step of HD training, **retraining**, performs model adjustment by iteratively going through the training dataset. Retraining is beneficial for HD to improve the prediction accuracy. During this step, the encoded hyper-vector of each input is created again, and its similarity with the existing class hyper-vectors is checked. If HD misclassifies, say that $\vec{H}^j$ from class $\vec{C}^j$ is predicted as class $\vec{C}^k$, it updates its model as follows:

$$\begin{aligned} \vec{C}^j &= \vec{C}^j + \eta \vec{H}^j \\ \vec{C}^k &= \vec{C}^k - \eta \vec{H}^j \end{aligned} \tag{6.3}$$

which means that the information of $\vec{H}^j$ causing misclassification to $\vec{C}^k$ is discarded.

Inference checks the similarity of each encoded test data with the class hyper-vector.

Most commonly, cosine similarity is used for the similarity check although other metrics (e.g., Hamming distance) could be suitable based on the problem. To calculate cosine similarity between hyper-vector $\vec{H}$ and class hyper-vector $\vec{C}^j$:

$$cos(\vec{H},\vec{C}^j) = \frac{\vec{H}\cdot\vec{C}^j}{\|\vec{H}\|\cdot\|\vec{C}^j\|} \tag{6.4}$$

which is calculated by the dot product of the $\vec{H}$ and $\vec{C}^j$ divided by the product of these two vectors' lengths. As an output of this step, HD provides the most similar class.

Hyperdimensional computing (HDC) was introduced as a brain-inspired learning solution for robust and efficient learning [133]. Although HDC has been used in a broad range of applications, the resiliency aspect of HDC classifiers has not been completely understood under adversarial attacks. There are some studies in the literature aiming to test HDC resiliency against adversarial attacks. Yang and Ren [134] showed that HDC can be vulnerable to adversarial samples. Their proposed adversarial attack misled the HDC classifier to a wrong prediction label. To enhance HDC security, they proposed adversarial training. Chen and Li [135] analyzed the impact of adversarial attacks on an HDC speech recognition classifier and their proposed attack based on differential evolution algorithm reached up to 85.7% attack success rate. Moraliyage et al. [136] evaluated the adversarial robustness of HDC text classifiers. They observed that different adversarial attacks lead to false prediction labels for language recognition and text classification tasks. Ma et al. [137] introduced distance-guided fuzzing which iteratively mutates inputs. By using the distance between query hypervector and reference hypervector, they generate new inputs that can trigger incorrect behaviors of the HDC model. Thapa et al. [138] developed an automated black-box differential testing framework to fool an HDC model. They were able to improve the HDC model accuracy using retraining. Wang and Jiao [139] designed HDC-specific poisoning attack framework based on confidence-based label-flipping method. They also proposed data sanitization as a defense to filter suspect samples before training. In our work, we show that by using non-linear encoding, HDC can stay resilient against different black-box

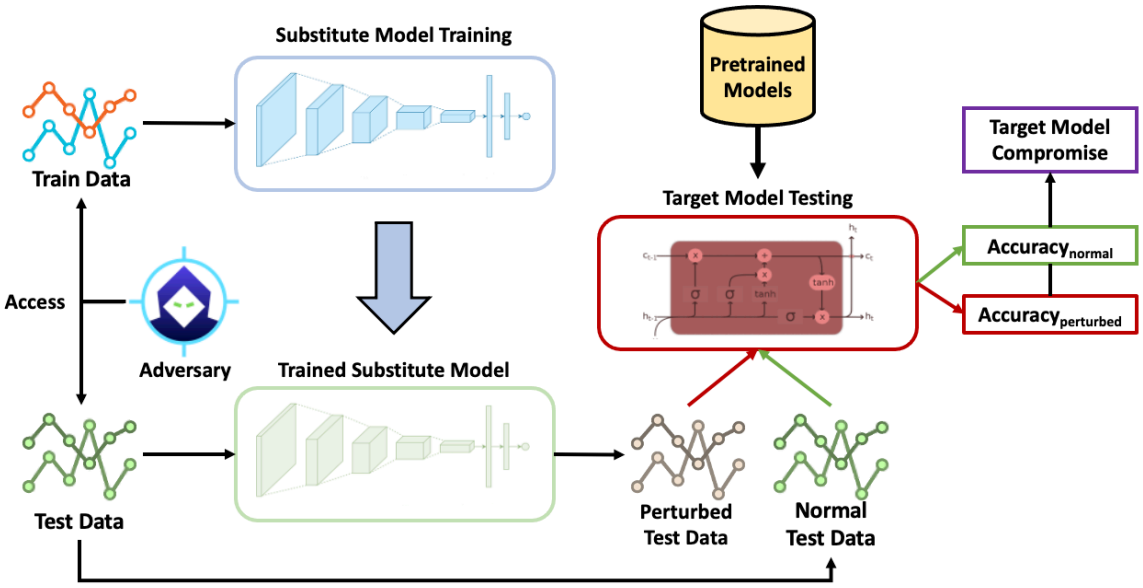


Figure 6.2. Black-box Attack Framework

adversarial attacks. To the best of our knowledge, HDC has not been used in the PdM domain where it can provide both lightweight and robust learning solution.

6.3 Proposed Framework

We use a black-box transferable attack strategy which first trains a substitute model and crafts new test instances using the trained substitute model. As our substitute model, we select wide deep convolutional neural network (WDCNN) since it is one of the most commonly used DL methods in intelligent fault diagnosis [87, 121]. For our black-box attack setting, we assume that an adversary has access to the training and test data, yet does not know anything about the attacked (target) models. We illustrate our black-box attack framework in Figure 6.2. Attacker first trains the substitute model (WDCNN) and use the trained model to create perturbed test data. Attacker can employ different attack strategies to obtain perturbed test data. Afterwards, adversary sends these crafted examples to the target models in testing time. In Figure 6.2, we give long short-term memory (LSTM) as the target model for illustration purposes. However, note that there is a pool of pretrained target models adversary is not aware

of (thus black-box attack). Attacker simply sends the created examples to the target models to see if the attack will be successful or not. We measure the attack success based on change in test data classification accuracy before and after the attack where accuracy is defined as: $Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of test samples}}$. Change in accuracy gives us the resiliency of a learning method which we measure by the metric called *Compromise* which is formulated as:

$$Compromise = \frac{Accuracy_{normal}}{Accuracy_{perturbed}} \tag{6.5}$$

where $Compromise > 1$ (under the assumption that attacks lead to worse prediction performance). The smaller the compromise value is, the more resilient the model becomes against the adversarial attack. For instance, given two methods RNN and LSTM, and their compromise values 5 and 2 respectively, we can conclude that LSTM is more resilient against the adversarial attack. If we have M number of adversarial attacks ($M > 1$), then we need to calculate the mean compromise value for each learning method as follows:

$$Compromise_{mean} = \left(\sum_{i=1}^{M} \frac{Accuracy_{normal}}{Accuracy^{i}_{perturbed}} \right) / M \tag{6.6}$$

Because we have multiple attack strategies, mean compromise gives a more accurate idea about single model resiliency. Overall, we obtain mean compromise value for each learning method and use this metric for our experimental analysis. Furthermore, to show the HDC resiliency improvement, we define the following improvement metric:

$$Improvement = \left(\frac{Compromise_{DL} - Compromise_{HDC}}{Compromise_{DL}} \right) \tag{6.7}$$

where $Compromise_{DL}$ denotes the single DL model mean compromise value, and HDC mean compromise is $Compromise_{HDC}$. We report the improvement in percentage (%). Improvement value shows the resiliency of the HDC learner compared to a single DL model.

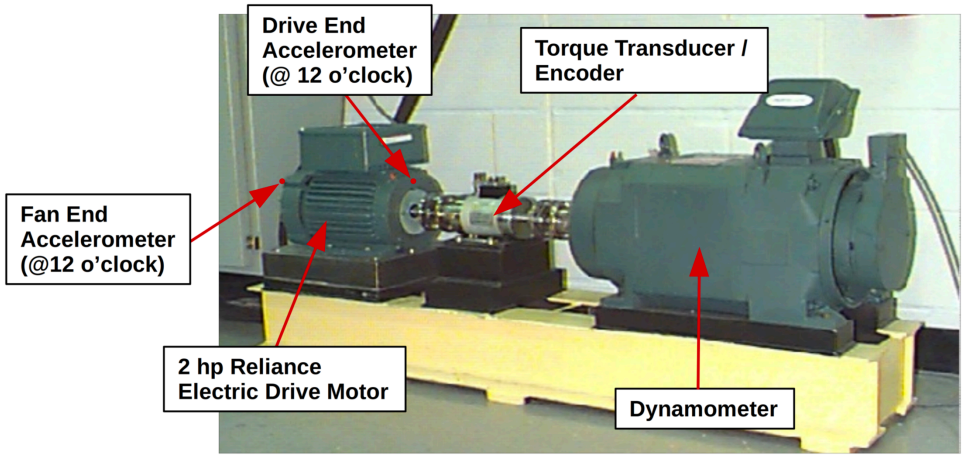


Figure 6.3. CWRU Experimental Test Apparatus

6.4 Experimental Analysis

6.4.1 Dataset Description

We use the Case Western Reserve University (CWRU) Bearing dataset [88]. Rolling element bearing (REB) failure is one of the most frequent reasons for machine breakdown leading to severe loss of safety and property [140]. Figure 6.3 represents the experimental test apparatus to collect this dataset. The data were collected from both the drive end accelerometer and the fan end accelerometer at 12k samples/second over a range of motor loads (from 0 hp to 3 hp). Both datasets (drive end and fan end) contain 19,800 training and 750 test samples. Bearing used in this experiment has three components: rolling element, inner race, and outer race. 9 different fault types are provided in the dataset based on the fault diameter (0.007, 0.014, and 0.021 inches) and the component (plus the normal bearing condition).

6.4.2 Experimental Setup

Selected Deep Learning (DL) Methods: We select 9 different DL methods: long short-term memory (LSTM) [128], gated recurrent unit (GRU) [129], wide deep convolutional neural network (WDCNN) [104], convolutional recurrent neural network (CRNN, CLSTM, CGRU) [141], and simplified CRNN (SCRNN, SCGRU, SCLSTM) [141]). We cover a good

range of DL methods, increasing the generalizability of our study.

Adversarial Attack Methods: We select the following parameters for our adversarial methods [114, 115]: amount of perturbation (ε): 0.1, step size (α): 0.001, number of iterations (I): 100, decay factor (μ): 1.

Parameter Selection: For both DL methods and HDC, we use a sliding time window of size 100 with a number of epochs of 100. We replicate each experiment 10 times and report average values. We run all experiments on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. For **DL methods**, the following hyper-parameters are selected: *Adam* optimizer with learning rate 0.001, *relu* activation function, batch size of 16. For **HDC**, we select the following parameters: encoding: non-linear, hyper-vector dimension size: 10,000, learning rate: 0.005, number of epochs: 100, similarity metric: cosine.

Number of Training Samples: We measure the resiliency of the selected methods by using different number of training samples while considering the whole test data. This is due to the fact that it might be extremely costly to label specific fault types in a collected dataset [121]. Specifically, our smallest experiment configuration uses 1.2% (240 samples) of the whole training data where we double this ratio until we reach approximately 38.8% (7680 samples). We call this ratio sample training ratio (STR) for the rest of this chapter. Considering different STRs is crucial for IFD since it might not always be feasible to label fault data for the whole training dataset. IFD methods should perform well under limited supervision [121].

6.4.3 Resiliency Analysis

Mean Compromise Comparison: We analyze the resiliency of selected DL models and HD using mean compromise metric defined in Equation 6.6. Figure 6.4 and Figure 6.5 show the mean compromise values of the 6 most resilient learning methods under different sample training ratios for drive end and fan end datasets respectively. In these figures, x-axis represents the selected STRs and y-axis gives the mean compromise values where each color represents a different learning method. We can observe that the mean compromise of a DL method changes

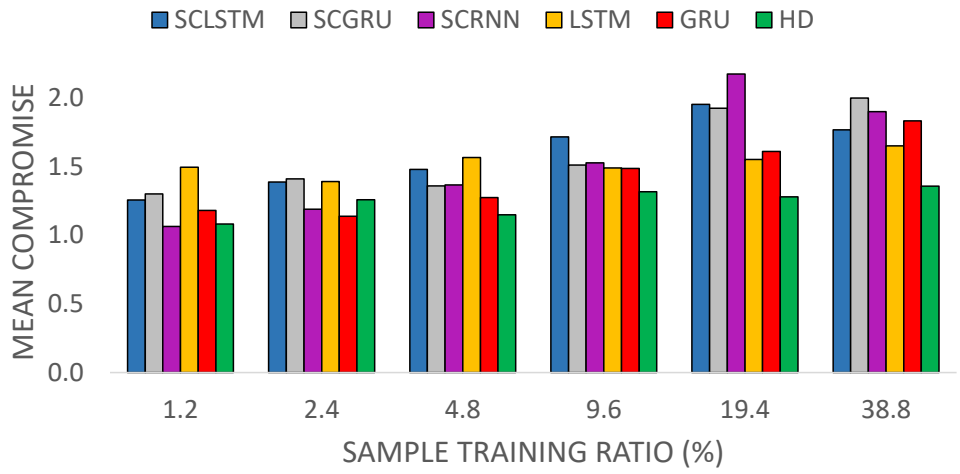


Figure 6.4. Mean Compromise Analysis (Drive End Dataset)

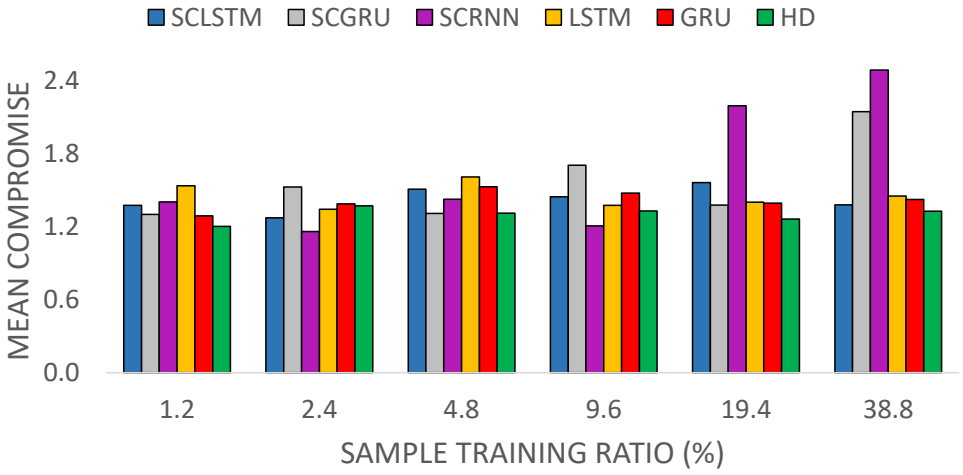


Figure 6.5. Mean Compromise Analysis (Fan End Dataset)

significantly. To illustrate, while SCRNN (represented with purple color) is one of the most resilient methods for really small STRs, it becomes the least resilient algorithm as we reach the maximum STR. For drive end dataset (Figure 6.4), we can observe that HD is the most resilient method for STRs greater than 2.4%. As the number of training samples increases, HD becomes a more resilient method compared to the DL algorithms. We can make a similar observation for fan end dataset as well (Figure 6.5). For STRs larger than 9.6%, HD is the most resilient method against adversarial attacks. To present a single mean compromise value (for better understanding), we calculate the average of mean compromise values over all STRs. Table 6.1

Table 6.1. Average Compromise Comparison

Method / Dataset	Drive end	Fan end
CGRU	2.77	2.52
CLSTM	2.52	2.69
CRNN	2.30	2.77
SCLSTM	1.59	1.42
SCGRU	1.58	1.56
SCRNN	1.54	1.69
LSTM	1.52	1.45
GRU	1.42	1.42
HD	**1.24**	**1.30**

Table 6.2. Average and Maximum Resiliency Improvement of HD over DL Methods

	Drive end		Fan end	
DL Method	**Average (%)**	**Maximum (%)**	**Average (%)**	**Maximum (%)**
CGRU	**51.34**	**61.9**	44.25	64.5
CLSTM	48.1	54.1	48.3	**67.5**
CRNN	42.1	52.4	**49.9**	64.4
SCLSTM	21.1	34.5	8.1	19.3
SCGRU	20.2	33.5	14.4	38.1
SCRNN	15.3	41.1	14.8	52.4
LSTM	18.5	27.6	10.0	21.7
GRU	10.9	25.9	8.0	14.2

presents these average compromise values for all the methods. When we compare DL methods (i.e., all methods excluding HD), we can observe that recurrent neural network structures are the most resilient. Specifically, GRU is the most resilient DL method with an average compromise value of 1.42 for both datasets. This observation can be due to the fact that our hybrid DL model structures contain convolutional layers. Since our crafted examples are based on wide deep convolutional neural network, more test examples are able to deceive hybrid methods. Most importantly, according to Table 6.1, **HD** is the **most resilient** method on average outperforminng other DL methods at both datasets. This shows that HD provides a resilient learning solution performing well even under different black-box adversarial attack configurations.

HD Resiliency Improvement: We calculate HD resiliency improvement over the selected DL models using Equation 6.7 for each STR configuration. Then, we find the maximum

Table 6.3. Target Models Training Time Comparison

	Sample Training Ratio (%)						Average
Method	**1.2**	**2.4**	**4.8**	**9.6**	**19.4**	**38.8**	**Normalized**
LSTM	151.0	366.1	514.8	980.4	1882.4	3294.8	25.1
GRU	161.5	307.7	451.5	861.5	1623.3	3165.5	23.0
CLSTM	16.0	31.8	81.0	174.6	413.7	820.2	5.4
CGRU	15.7	61.2	94.2	196.9	360.8	727.5	5.1
SCLSTM	15.6	28.9	52.2	119.5	246.4	488.6	3.3
SCGRU	15.2	27.6	50.6	106.8	228.1	448.5	3.1
CRNN	10.8	19.4	36.1	77.4	167.6	327.1	2.2
SCRNN	7.7	13.8	26.9	53.6	98.6	198.3	1.4
HD	**6.0**	**10.2**	**18.6**	**37.5**	**72.0**	**141.8**	**1.0**

and average improvement for each DL method. Table 6.2 demonstrates the HD average and maximum resiliency improvement over the selected DL methods. For drive end experiment, HD improves DL model resiliency by up to 61.9%, and up to 67.5% for the fan end dataset. Compared to the most resilient DL method (GRU), HD improves the resiliency by up to 25.9% and 14.2% for drive end and fan end data sets respectively. We are able to verify that HD provides a resilient learning solution against adversarial attacks.

Training Overhead Comparison: Table 6.3 presents target models' training time (in seconds). In this table, each row represents a different target model (where the models are ordered in decreasing training overhead) and each column corresponds to the selected STR. We can observe that **HD** is the **most lightweight** model across all STRs. In the last column of this table, we share the average (across sample training ratios) normalized training time with respect to HD. HD can achieve up to 25.1$\times$ training speed up compared to DL methods. Compared to the most resilient DL method (GRU), HD brings 23$\times$ faster training. From this analysis, we can conclude that HD also provides a computationally efficient learning solution while being resilient to adversarial attacks. Training overhead is especially critical for IIoT systems since data is collected continuously. When new data arrives, learning models require retraining to keep their prediction performances at a certain level [7]. HDC can alleviate this retraining overhead due to its lightweight feature.

6.5 Conclusion

Adversarial attacks deceive ML methods with fake inputs leading to worse prediction performance. Hyperdimensional computing (HDC) is a novel learning solution which is robust against noise. In this paper, we utilize HDC to stay resilient against created black-box attack scenarios. Our experiments show that HDC can improve the resiliency of the state-of-the-art DL methods by up to 67.5%. HDC can also achieve up to 25.1× training speed up compared to DL methods, providing a lightweight learning solution. This means that HDC can still perform well and efficiently under adversarial attacks, leading to more accurate replacement and maintenance decisions even under cyberattacks.

In this chapter, we showed that HDC can be used as a lightweight, and resilient learning solution in IIoT domain. However, there is still a need to analyze its resiliency against adversarial attacks. To reach this goal, we devise HDC specific adversarial attack in the next chapter.

6.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1911095, #2003277, and #2003279.

Chapter 6, in part, is a reprint of material as it appears in O. Gungor, T. Rosing, and B. Aksanli, "HD-I-IoT: Hyperdimensional Computing for Resilient Industrial Internet of Things Analytics". IEEE/ACM Design, Automation and Test in Europe Conference and Exhibition (DATE), 2023. The dissertation author was the primary investigator and author of this material.

Chapter 7

Hyperdimensional Computing Novel Adversarial Attack Design

7.1 Introduction

The security solutions designed for traditional IT systems cannot be directly applied for IIoT systems due to IIoT's constrained functionality, limited power, and lightweight network protocols [142]. Intrusion Detection System (IDS) is one of the security solutions that monitor the network data to detect attacks and anomalies [10]. ML methods have been heavily used for IDS due to their great performance in detecting attacks [143, 144, 145]. However, ML methods are quite vulnerable to adversarial attacks. These attacks could pose significant threats to ML-based IDS where data collected from different devices can be perturbed to cause malicious data to be classified as benign, consequently bypassing the IDS. Hence, there is a need to evaluate ML-based IDS against adversarial attacks and create realistic effective attacks that can deteriorate IDS classification performance. By understanding the adversarial robustness rigorously, we can develop better defense mechanisms that can protect IIoT systems against these attacks.

Hyperdimensional computing (HDC) was introduced as a brain-inspired learning solution for robust and efficient learning. Compared to deep neural networks, HDC has shown advantages such as smaller model size, less computation cost, and robustness to noise, making it a promising alternative in low-cost computing devices in IIoT [146]. To the best of our knowledge, HDC has not been used in an ML-based IDS domain previously. HDC can be a suitable IDS mechanism

since it provides high energy efficiency, low power consumption, and fast training/inference while its prediction performance is comparable with well-known ML methods. Similar to ML methods, HDC can also be vulnerable to small perturbations on input data to produce wrong classification [137, 138]. Previous studies on HDC security [134, 137, 136] mainly focus on simple perturbations which are easy to detect and defend against, decreasing their effectiveness against stronger attacks. Therefore, our goal is to develop an HDC adversarial attack mechanism that would consistently work better than individual simple attacks.

In this chapter, we propose a diversity-induced adversarial attack framework for HDC. We present our high-level framework in Figure 7.1. Given the test data, we first apply 9 different perturbation methods ranging from transfer adversarial attacks to simple perturbations. For transfer attacks, we utilize a pre-trained convolutional neural network. Then, we use perturbed test data to calculate diversity among attacks. To introduce diversity, we measure pair-wise Manhattan distance among attacks. By diversity inclusion, we eliminate possible overlap in adversarial subspaces, minimize HDC encoding overhead, and increase attack performance. Based on the calculated distances, we first select the two most diverse attacks and provide these attacks to the sample based attack selection process. Here, among the attacks leading to misclassification, we select the most effective attack which gives the maximum distance between attack hyper-vector and pre-trained HDC class hyper-vector. We then check if the attack performance is improved, i.e., lower F_1 score. If this holds, we expand the attack set until no further improvement is obtained. The experimental results on the X-IIoTID dataset [147] show that our attack design is able to fool HDC model significantly more compared to selecting the same attack for all samples or random attack selection. We can improve the attack success rate by up to 36%, and F_1 score by up to 61% compared to the most effective single attack.

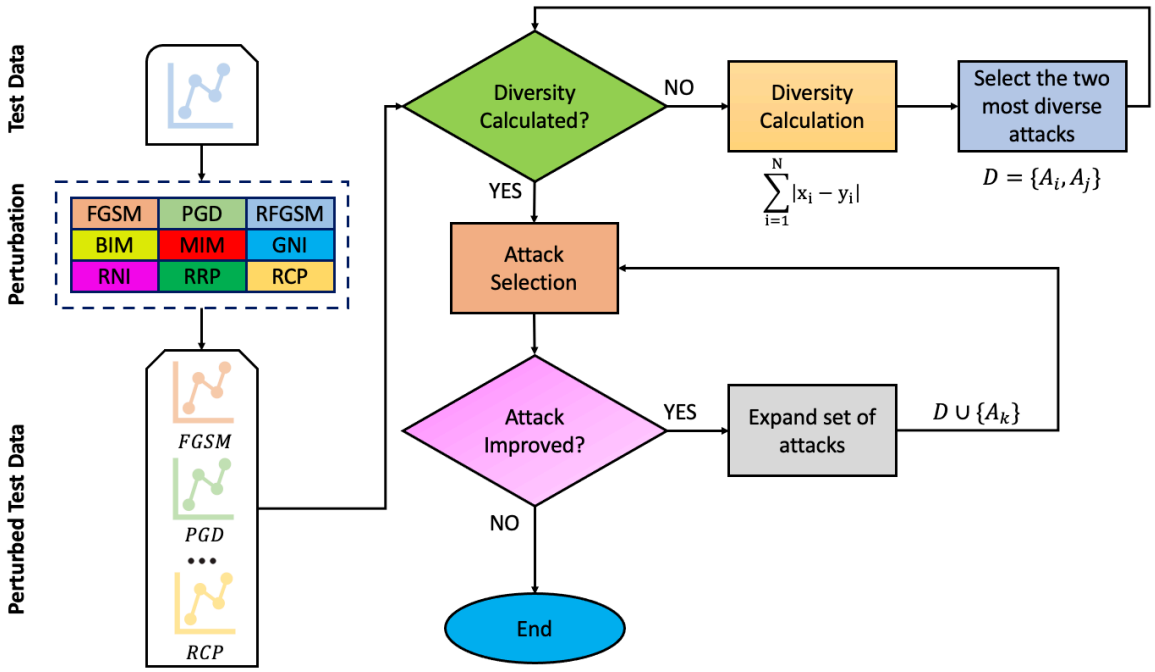


Figure 7.1. Our Proposed Attack Design Framework

7.2 Related Work

In this chapter, we have used random projection encoding for HDC [148]. Training and test steps of HDC learning are identical to presented material in Chapter 6.

Random projection encoding: Assume that a feature vector in original space $F = \{f_1, f_2, \ldots, f_n\} \in \mathbb{R}^n$. Encoding stage maps this feature vector to a D-dimensional hypervector $H = \{h_1, h_2, ..., h_D\} \in \mathbb{R}^D$ where $D \gg n$. Random projection encoding first creates D dense bipolar vectors with the same dimensionality as original domain, $P = \{p_1, p_2, \ldots, p_D\}$, where $p_i \in \{-1, 1\}^n$. The inner product of a feature vector with each randomly generated vector gives us a single dimension of a hypervector in high-dimensional space. For encoding, we perform a matrix vector multiplication between the projection matrix and the feature vector: $H = sign(PF)$ where *sign* is a function that maps the result to +1 or -1.

ML-based IDS is a security solution that utilizes historical IoT network data to train ML models and detects attacks and anomalies. Different ML methods are proposed in the literature

such as logistic regression, support vector machine, random forest, deep neural network, and recurrent neural network [149]. Although these methods provide great prediction performance, they are quite sensitive to small perturbations in the input data. An adversary can tamper with the data inputted into the ML model to fool the learner, exacerbating the classification performance.

HDC can be used in ML-based intrusion detection systems due to its lightweight and robust characteristics. Although HDC has been used in a range of applications, the security aspect of HDC classifiers has not been completely understood under strong attacks. Existing HDC security studies focus on distinct domains: image classification [134, 137, 138, 139], text classification [136], speech recognition [135]. Nevertheless, they mainly utilize simple perturbations, e.g., rotation, skew, noise, to create adversarial attacks. These attacks are easy to detect and defend, bringing the need of understanding HDC classifiers under stronger and more realistic adversarial attacks. One work in this direction is published by Gungor et al. [17] who tested HDC under transfer attacks. They showed that HDC is more resilient against different adversarial attacks than well-known DL methods. Different than the state-of-the-art, we develop an attack mechanism that works significantly better than simple and single attack scenarios.

7.3 Adversarial Attack Design Framework

Figure 7.1 represents our diversity included adversarial attack design framework. Given pre-trained HDC and CNN models (trained previously using the training data), the first step is to create perturbed test data via nine different perturbation methods. After we obtain the perturbed test data, we introduce diversity to prevent possible overlaps in adversarial subspaces, minimize HDC encoding overhead, and increase attack effectiveness. We start with the two most diverse attacks and then increment the number of attacks until no further improvement (prediction performance under attacks) is observed. Given diverse set of attacks, we then perform sample-wise attack selection to find the attack that can fool the HDC model the most based on the distance between class hyper-vector and attack hyper-vector. Here, we assume that the

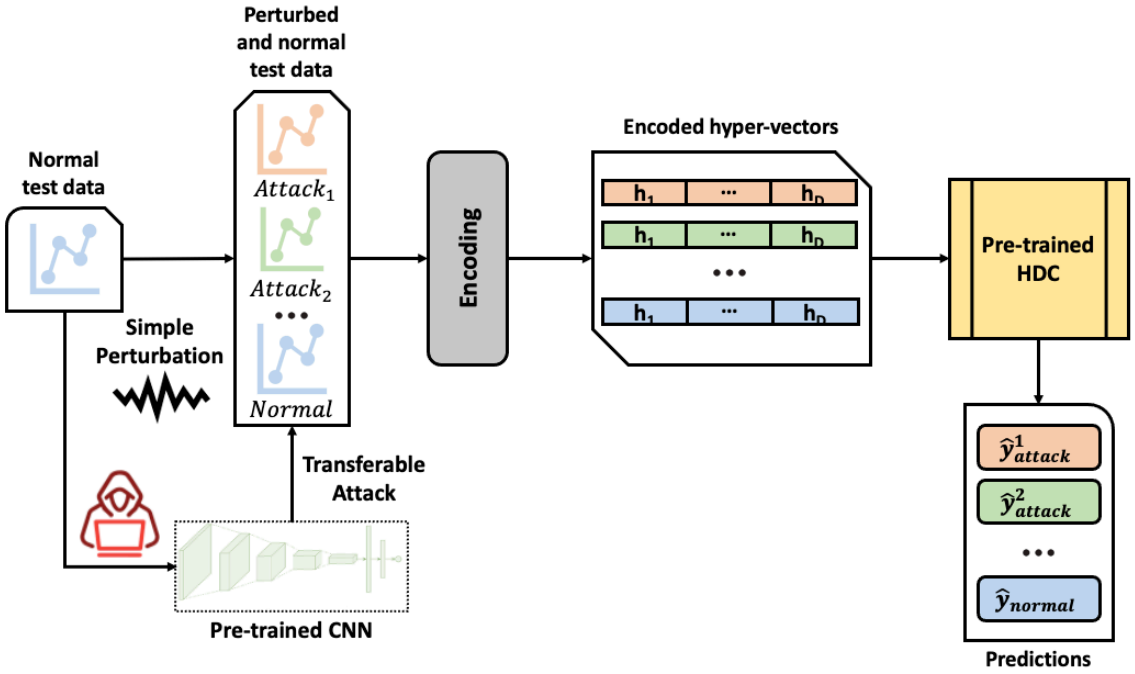


Figure 7.2. Perturbation Framework

attacker can access the pre-trained HDC model to send a query. Overall, our attack design framework consists of 3 main modules: perturbation creation, diversity inclusion, and real-time attack selection.

7.3.1 Perturbation Creation

Figure 7.2 illustrates our perturbation framework which consists of two groups of perturbation methods: (i) transfer adversarial attacks, and (ii) simple perturbations. Transfer attacks start with the attacker accessing pre-trained CNN model (substitute model) and the test data. The attacker exploits loss gradient information in the target CNN and adopts five different attack generation methods to create perturbed test data: fast gradient sign method (FGSM) [112], randomized fast gradient sign method (RFGSM) [127], projected gradient descent (PGD) [115], basic iterative method (BIM) [116], and momentum iterative method (MIM) [114]. The perturbed data is then transferred to pre-trained HDC (target model). Although these attacks are not HDC-specific, an attacker relies on the *transferability property* which is satisfied when an attack

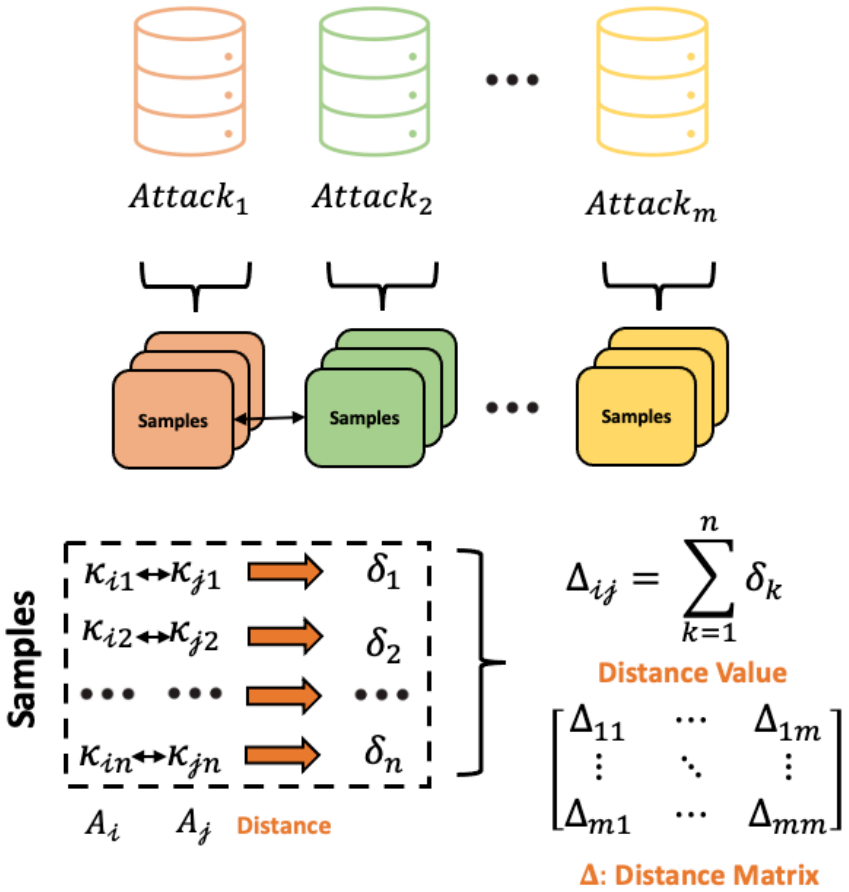


Figure 7.3. Diversity Calculation Framework

developed for a substitute model is also effective against the target model. Other than adversarial attacks, we also include 4 simple perturbation methods [137]: random row perturbation (RRP), random column perturbation (RCP), random noise injection (RNI), and Gaussian noise injection (GNI). Each attack uses a parameter, called perturbation amount (ϵ), that denotes the amount of noise added to the clean data. Based on the selected diverse set of attacks after diversity inclusion, we encode the selected perturbed attack data and obtain the encoded attack hyper-vectors as illustrated in Figure 7.2. These are then given to the pre-trained HDC model and we obtain class predictions for the attack $\hat{y}^i_{attack}$ and clean test data $\hat{y}_{normal}$. We will use encoded HVs and HDC predictions in attack selection.

7.3.2 Diversity Inclusion

We introduce diversity to select the most diversified set of attacks due to 3 main reasons: (i) different attacks can lead to same prediction labels due to overlap in adversarial subspaces, (ii) HDC encoding is computationally expensive [148], and (iii) attack performance can be increased by considering a subset of attacks. Overall, our goal is to minimize HDC encoding overhead while keeping attack performance at a maximum level. Figure 7.3 depicts our diversity calculation process. Given n samples and m attacks, we first calculate sample-wise distance (δ) among attacks. To find δ, we use Manhattan distance as a distance metric. Then, to find the pair-wise distance between attacks i and j (Δ_{ij}), we sum the distances over all samples, i.e., $\Delta_{ij} = \sum_{k=1}^{n} \delta_k$. To construct the distance matrix Δ (which is hollow symmetric), we place Δ_{ij} appropriately. For instance, the second row and the third column of Δ corresponds to the Manhattan distance between second and third attacks. After we generate this matrix, the next step is to select the largest value in this matrix, providing us the set of the most diverse two attacks $D = \{A_i, A_j\}$. After sample-wise attack selection, we revisit Δ to find the next most diverse attack (second largest value in Δ) and add the corresponding attack to the existing attack set. Let A_k be the next most diverse attack, then we expand D as follows: $D = D \cup \{A_k\}$. We expand the set D until we no longer improve the attack performance which we measure by F_1 score.

7.3.3 Real-time Attack Selection

Given the prediction labels, we first compare attack predictions with clean data prediction to test if an attack can fool the HD model for each sample. Let γ denote the number of attacks that can fool HD. We analyze 3 different scenarios based on the value of γ:

1. $\gamma = 0$: This scenario represents the worst-case scenario where both clean and attack data lead to same class prediction. In this scenario, we need to tune attacks (e.g., increase perturbation amount ε) or generate a completely new attack.

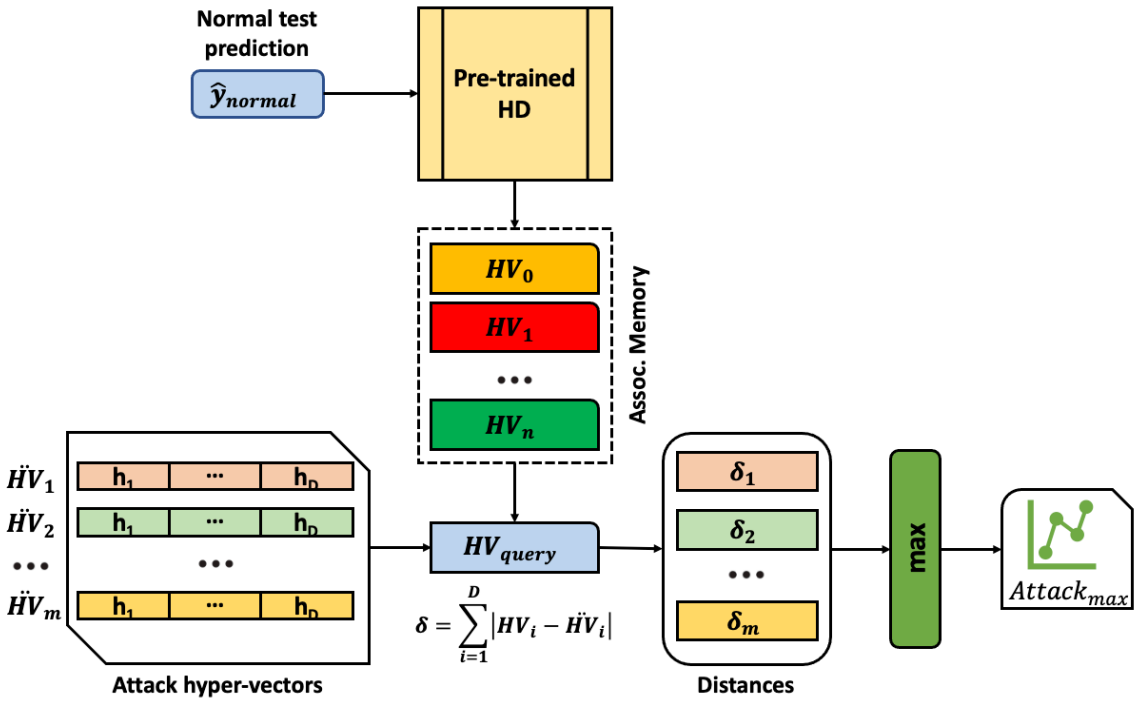


Figure 7.4. Attack Selection Framework

2. $\gamma = 1$: There is only one attack that can mislead HDC. We simply select that single attack.

3. $\gamma > 1$: This scenario occurs when there are multiple attacks that can mislead HDC. Here, there is a need to select the most effective attack. Figure 7.4 presents our attack selection framework. For each given sample, our goal is to select the attack that is able to fool the HDC model the most. We perform the evaluation using Manhattan distance among query hyper-vector (clean test sample class hyper-vector from HDC associative memory) and attack hyper-vectors. We select the Manhattan distance metric since it is the most preferable for high dimensional applications [150]. For each attack i, we calculate its Manhattan distance δ_i from the query hyper-vector. Then, we select the maximum distanced attack for a given sample. We repeat this attack selection process for all samples.

7.4 Experimental Analysis

7.4.1 Dataset Description

To validate the proposed attack design against HDC, we use a realistic IIoT intrusion dataset, X-IIoTID [147]. This connectivity agnostic and device agnostic dataset reflects the changes and heterogeneity of network traffic and systems' activities generated from various IIoT devices, connectivity protocols, and communication patterns. To create the dataset, Brown-IIoTbed testbed is used which is a holistic and end-to-end IIoT security testbed developed based on an industrial Internet reference architecture (IIRA). This dataset contains 18 different attacks: generic scanning, scanning vulnerabilities, fuzzing, discovering resources, brute force attack, dictionary attack, malicious insider, reverse shell, MitM attack, MQTT cloud broker-subscription, Modbus-Register reading, TCP relay attack, command and control, exfiltration, false data injection, fake notification, crypto-ransomware, and ransom denial of service. Overall, with the normal data, we have a classification problem with 19 labels. The collected data is related to the end-to-end network traffic (i.e., from physical field devices to the edge gateway and from the edge gateway to the cloud and enterprise devices), host device logs, and the host device's resources, physical properties, and alert logs. The period for capturing normal data began on December 5, 2019, ran for many hours each day, and ended on March 23, 2020 (not continuous). The experiments on the collected attack data took place over different times and days from January 7, 2020 to March 27, 2020, with each attack experiment repeated multiple times to collect more data.

7.4.2 Experimental Setup

We run all experiments on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. For CNN, we selected *SGD* optimizer with learning rate 0.01, *relu* activation function, and batch size of 32. For HDC, we set hypervector dimensionality, D, to 1000 and used random projection encoding. To measure attack performance, we use three different metrics: (untargeted)

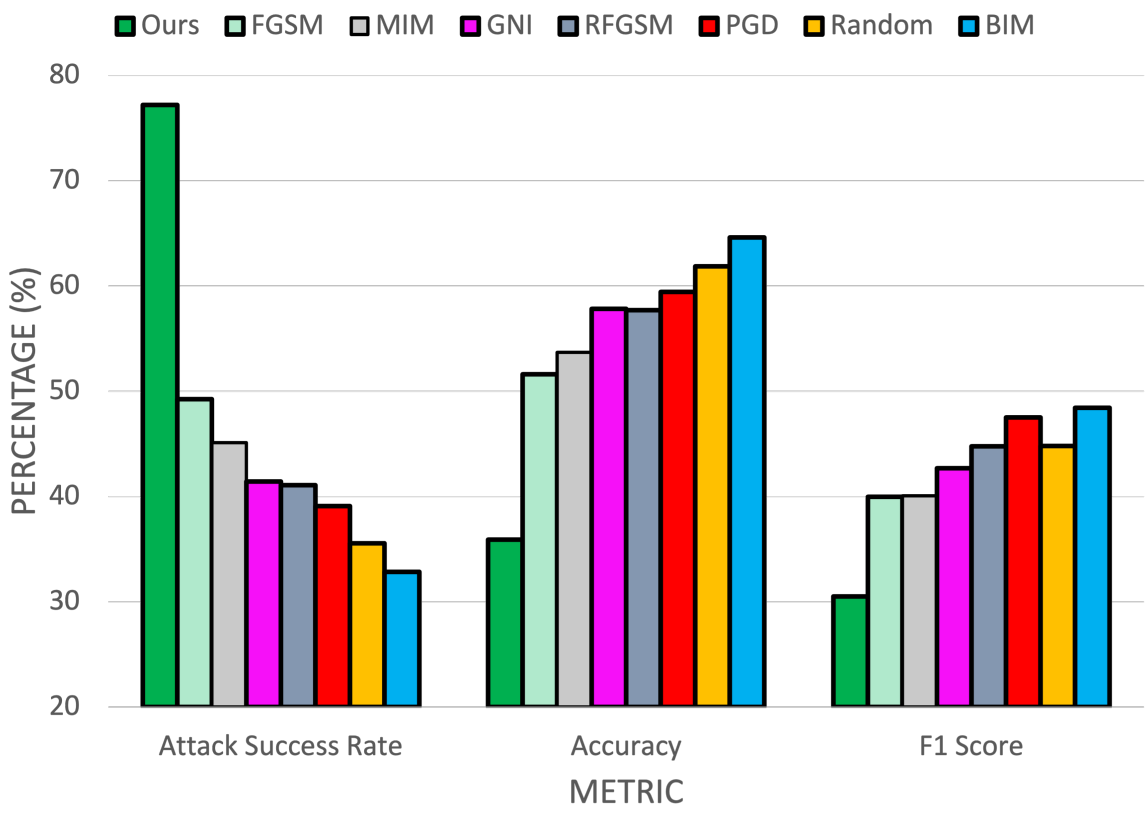


Figure 7.5. Attack Performance Comparison ($\varepsilon = 0.1$)

attack success rate, accuracy, and F_1 score. Attack success rate is the ratio of misclassified number of samples to the total number of samples under any attack. Accuracy is the ratio of number of correct predictions to the total number of samples. F_1 score is formulated as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7.1}$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

7.4.3 Experimental Results

We compare our diversity-induced attack design with two benchmarks: (i) selecting the same attack for all samples (denoted by the attack name, e.g., FGSM), (ii) random attack selection for a given sample (denoted by Random). We experimented with different perturbation amounts

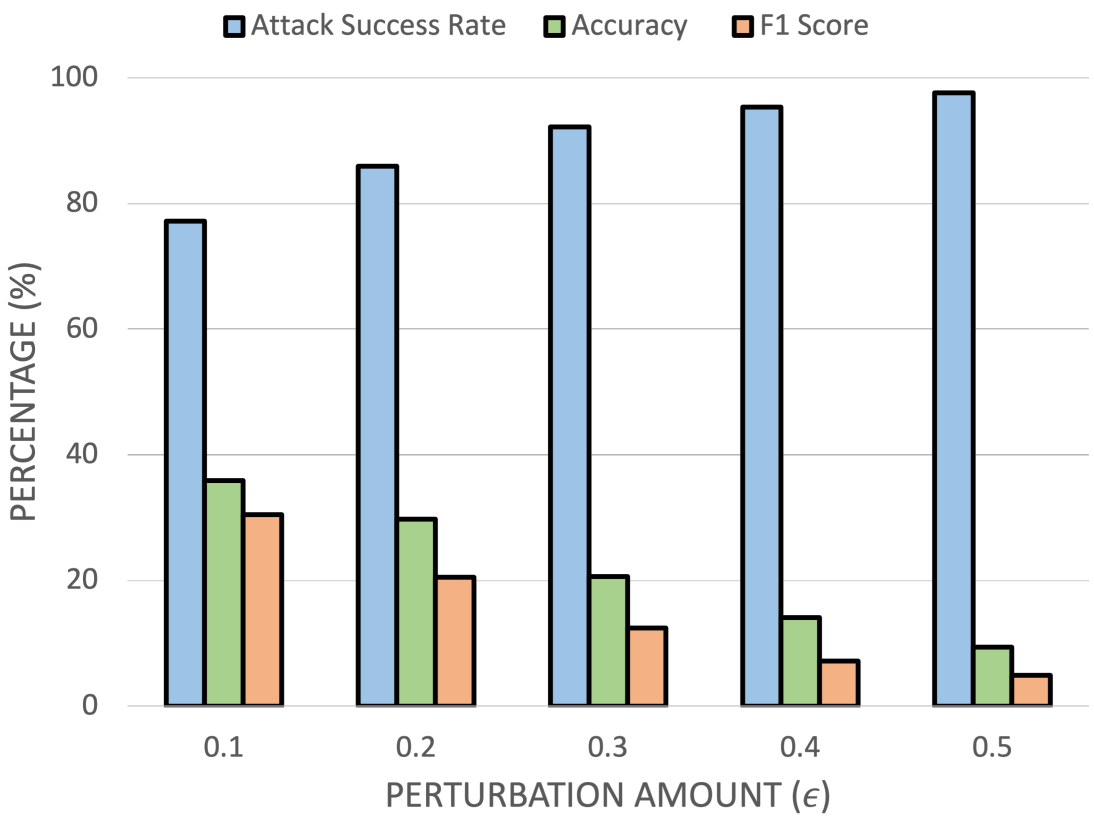


Figure 7.6. Our Attack Design Performance

(ε) from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Figure 7.5 demonstrates attack performance comparison where $\varepsilon = 0.1$. While y-axis provides percentage values, we have our metrics on the x-axis: attack success rate, accuracy, and F_1 score. In this figure, green denotes our attack design (leftmost bar), random selection is represented with orange, and selected single attacks (best performing 6 attacks) are represented with distinct colors. We can see that our approach is far superior to the single attacks. Our attack design can reach 77.2% attack success rate while the most effective single attack (FGSM) can only achieve 49.3%. In terms of accuracy and F_1 score, we observe the best attack performance under our approach. We can decrease the prediction accuracy to 35.9% and F_1 score to 30.5%. However, with FGSM, we obtain 51.6% and 39.9% accuracy and F_1 score respectively. Random selection approach is somewhere in between different single attacks.

Figure 7.6 illustrates our attack design performance under selected ε values. We present attack success rate, accuracy, and F_1 score with blue, green, and orange colors respectively.

Table 7.1. Improvement over the best single attack (FGSM)

Perturbation Amount (ε)	**Attack Success Rate (%)**	**Accuracy (%)**	**F1 Score (%)**
0.1	**36.2**	30.5	23.7
0.2	26.8	24.1	30.1
0.3	22.2	27.8	45.9
0.4	18.1	41.7	**61.1**
0.5	15.8	**52.5**	57.1
Average	23.8	35.3	43.6

Table 7.2. Attack Selection Overhead

Number of Attacks	2	3	4	5	6	7	8
Elapsed Time (ms)	0.91	1.47	1.49	1.52	1.75	2.08	2.12

We can observe that as ε increases, our method became much more effective while prediction performance (both accuracy and F_1 score) decreases significantly. We can reach up-to 97.6% attack success rate, 9.4% accuracy, and 4.9% F_1 score (when $\varepsilon = 0.5$). With chosen ε values, the selected number of attacks are 6, 5, 5, 5, and 7 respectively. This selection consistently gives us the lowest F_1 scores. We also make a comparison with the single best attack (FGSM) under different ε values. Table 7.1 presents the results for our method's improvement over FGSM. As ε increases, attack success rate improvement decreases while accuracy and F_1 score improvement increases. We can reach up-to 36.2% attack success rate improvement, 52.5% accuracy improvement, and 61.1% F_1 score improvement over.

Attack Selection Overhead: When we analyze our real-time attack selection, we observe that it has a small computational overhead while increasing attack effectiveness significantly. Table 7.2 shows attack selection overhead with respect to increasing number of attacks. As the number of attacks increases, the attack selection overhead also increases. In the worst case, the overhead of our framework is limited by 2.12 ms, and on average 1.75 ms.

7.5 Conclusion

IIoT security is one of the major obstacles that prevent the widespread adoption of IIoT technology [14]. Intrusion Detection Systems (IDSs) dynamically monitor the behavior of a system to detect and respond to malicious activity. Machine learning methods are quite popular in IDSs due to its accurate prediction performance. Hyperdimensional computing (HDC) is a brain-inspired learning solution for robust and efficient learning which can be a beneficial ML solution for IDSs. However, HDC is sensitive to adversarial attacks, hence increasing the need for investigating its security aspect. In this chapter, we proposed a novel adversarial attack design targeting HDC. After we find out the most diverse set of attacks, we select the most effective attack sample by sample. Our experimental results show that we can improve attack success rate by up to 36%, and F_1 score by up to 61% compared to the most effective single attack.

7.6 Acknowledgements

This work has been funded in part by NSF, with award numbers #1911095, #2003277, and #2003279.

Chapter 7, in part, is a reprint of material as it appears in O. Gungor, T. Rosing, and B. Aksanli, “Adversarial-HD: Hyperdimensional Computing Adversarial Attack Design for Secure Industrial Internet of Things”. Cyber-Physical Systems and Internet of Things Week, 2023. The dissertation author was the primary investigator and author of this material.

Chapter 8

Summary and Future Work

The Industrial Internet of Things (IIoT) is the connection of industrial assets with the information systems and the business processes. It continuously monitors and analyzes collected data towards better system efficiency and reliability. It is crucial to utilize collected system data towards high-level business decisions. Machine learning (ML) solutions are deployed for predictive analytics which aims to predict the likelihood of future outcomes. There are numerous challenges that should be addressed to widely deploy ML solutions in IIoT environments: (i) difficulty of selecting a single ML method that can perform well across different IIoT settings, (ii) impact of adversarial attacks on ML prediction performance, and (iii) creating efficient ML solutions to be used at resource-constrained IIoT devices.

8.1 Thesis Summary

This thesis aims to devise intelligent, secure, and efficient ML solutions for IIoT predictive analytics. It presents novel methods to increase prediction performance, secure ML algorithms against adversarial attacks, and deploying lightweight learning approaches for IIoT environments. We first design novel ML solutions which are capable of great prediction performance under varying IIoT conditions. Nevertheless, to implement these intelligent learning solutions, IIoT environments employ various sensors and edge computing devices. This dramatically increases IIoT security vulnerabilities. Notably, attackers can conduct adversarial attacks to significantly

affect ML prediction performance. In secure learning, we aim to solve this problem by proposing resilient learning solutions against adversarial attacks. Although these intelligent and secure learning solutions bring many benefits, they also increase the computational burden of IIoT systems, which rely on small computing devices. Hence, there is a need to deploy efficient learning solutions in resource-constrained IIoT devices. For that purpose, we devise efficient learning solutions for IIoT predictive analytics. The following discussion summarizes thesis contributions briefly:

8.1.1 Diversity-induced Optimally Weighted Ensemble Learner

To address training overhead in ensemble learning, we propose a diversity-induced ensemble learning solution. To keep ensemble accuracy at a certain level, we solve a quadratic programming model for optimal weights discovery. For diversity, we select the most diverse learners. Our approach provides 39.2% faster retraining than only accuracy included ensemble with only 3.4% loss in accuracy.

8.1.2 Ensemble Few-Shot Learning under Limited Labeled Data

We present a novel few shot ensemble learning to solve limited labeled data problem. Our approach combines five different Siamese neural network architectures using an iterative majority voting classifier. Our approach improves the best algorithm significantly while using very limited labeled data. We obtain up to 16.4% improvement over the best algorithm by only using 0.3% of the training data.

8.1.3 Resilient Stacking Ensemble Learner Against Adversarial Attacks

To address adversarial attacks in IIoT domain and create a resilient learning solution, we devise a novel stacking ensemble learning-based framework. Our ensemble learner combines the most resilient DL method predictions based on our iterative selection procedure. We show that adversarial attacks can impact the performance of DL-method considerably, leading up to

120× prediction performance loss. The proposed stacking ensemble improves resiliency against adversarial attacks by up to 60% compared to the most resilient single method.

8.1.4 Diversity Promoting Ensemble Adversarial Training

We present a novel defense framework against adversarial attacks. Our defense first measures the loss gradient similarity among pre-trained ML models and selects the most dissimilar ones to promote diversity. Then, we create perturbed training instances using the selected diverse base learners and augment those examples into our training data. Our defense improves the resiliency by up to 97% compared to state-of-the-art training settings.

8.1.5 Hyperdimensional Computing for Resilient IIoT Predictive Analytics

To present an efficient and robust learning solution for IIoT systems, we apply hyperdimensional computing (HDC) for intelligent fault diagnosis. We also test HDC against various adversarial attacks. HDC leads to a more resilient and lightweight learning solution than the state-of-the-art deep learning methods. HDC has up to 67.5% higher resiliency compared to the state-of-the-art methods while being up to 25.1× faster to train.

8.1.6 Hyperdimensional Computing Novel Adversarial Attack Design

To further explore HDC resiliency against adversarial attacks, we devise a novel adversarial attack design. We select the most diverse set of attacks and perform a real-time attack selection. Compared to the most effective single attack, our design strategy can improve attack success rate by up to 36%, and F_1 score by up to 61%.

8.2 Future Work

8.2.1 Intelligent Learning

Although this thesis addressed certain IIoT predictive analytics challenges, there are other key challenges that require attention. These are (i) dynamic nature of real industrial environments, and (ii) black box aspect of state-of-the-art ML models. IIoT is a dynamic and ever-changing system where deployed ML models might lose their accuracy across time. Keeping their performance requires frequent model updates leading to high computational cost. ML solutions should be adaptive to constantly changing IIoT environments by learning continuously. Lifelong Machine Learning (LML) can be a suitable solution to address this issue. LML is capable of adapting past knowledge to handle unseen situations [151]. Another challenge with deploying ML models is their lack of transparency, trustworthiness, and explainability. Most ML models are black box where it is not possible to explain the output based on inputs. Explainable Artificial Intelligence (XAI) helps AI decisions to be understood, interpreted, and explained by humans [152]. Implementing XAI for IIoT is critical since it facilitates the understanding behind certain predictions for industrial stakeholders.

8.2.2 Secure Learning

There are two main adversarial attack challenges to be addressed for IIoT: (i) offline static adversarial attacks/defenses, and (ii) labeled data assumption in adversarial attack/defense creation. IIoT is a dynamic system where collected data and operating conditions can change continuously. Thus, adversarial attacks can happen in real-time with unknown perturbations and frequencies [153]. To address this challenge, there is a need to create real-time adversarial attacks and defenses. Other than this challenge, most state-of-the-art adversarial attacks and defenses rely on the existence of labeled data which might not hold for IIoT systems. To break this assumption, it is crucial to create adversarial attacks and defenses in an unsupervised manner.

8.2.3 Efficient Learning

There are two possible paths for HDC to be widely used in IIoT: (i) HDC algorithmic novelties, and (ii) HDC-based adversarial defense mechanisms. HDC is an emerging learning paradigm which requires further investigation to apply it safely for the IIoT domain. To illustrate, hyper-vector size, encoding algorithm, and data type (e.g., image, time-series) significantly affects HDC prediction performance. Thus, there is a need to develop efficient algorithmic HDC solutions, e.g., novel encoding methods, integrating ML with HDC, and implement those solutions for the IIoT domain. The second path is related to designing HDC specific defense mechanisms. In Chapter 7, we showed that HDC can be vulnerable to adversarial attacks. Hence, it is imperative to protect HDC against adversarial attacks. For that purpose, novel defense designs should be created for HDC-based predictive analytics.

Bibliography

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *Ieee access*, vol. 6, pp. 78238–78259, 2018.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[3] P. Daugherty and B. Berthon, "Winning with the industrial internet of things: How to accelerate the journey to productivity and growth," *Dublín: Accenture*, 2015.

[4] T. Kotsiopoulos, P. Sarigiannidis, D. Ioannidis, and D. Tzovaras, "Machine learning and deep learning in smart manufacturing: The smart grid paradigm," *Computer Science Review*, vol. 40, p. 100341, 2021.

[5] H. J. Wilson *et al.*, "How companies are using machine learning to get faster and more efficient," *Harvard Business Review*, 2016.

[6] G. R. Mode and K. A. Hoque, "Crafting adversarial examples for deep learning based prognostics (extended version)," *arXiv preprint arXiv:2009.10149*, 2020.

[7] O. Gungor, T. S. Rosing, and B. Aksanli, "Dowell: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 3125–3134, 2021.

[8] "Predictive maintenance market." https://www.marketsandmarkets.com/\PressReleases/operational-predictive-maintenance.asp, 2020.

[9] M. S. K. Kopuru, S. Rahimi, and K. Baghaei, "Recent approaches in prognostics: State of the art," in *ICAI*, pp. 358–365, 2019.

[10] E. Anthi, L. Williams, M. Rhode, P. Burnap, and A. Wedgbury, "Adversarial attacks on machine learning cybersecurity defences in industrial control systems," *Journal of Information Security and Applications*, vol. 58, p. 102717, 2021.

[11] R. Polikar, "Ensemble learning in ensemble machine learning: Methods and applications; zhang, c., ma, y., eds," 2012.

[12] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," *Journal of manufacturing systems*, vol. 47, pp. 93–106, 2018.

[13] D. Wu *et al.*, "Cybersecurity for digital manufacturing," *Journal of manufacturing systems*, vol. 48, pp. 3–12, 2018.

[14] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of industrial internet of things security: Requirements and fog computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[15] M. Al-Hawawreh *et al.*, "An efficient intrusion detection model for edge system in brownfield industrial internet of things," in *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, pp. 83–87, 2019.

[16] "Understanding the cybersecurity threat landscape in asia pacific." https://news.microsoft.com/apac/features/cybersecurity-in-asia/, 2019.

[17] O. Gungor, T. Rosing, and B. Aksanli, "Res-hd: Resilient intelligent fault diagnosis against adversarial attacks using hyper-dimensional computing," *arXiv preprint arXiv:2203.08148*, 2022.

[18] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[19] K. Warr, *Strengthening deep neural networks: Making AI less susceptible to adversarial trickery*. O'Reilly Media, 2019.

[20] J. Li, Y. Liu, T. Chen, Z. Xiao, Z. Li, and J. Wang, "Adversarial attacks and defenses on cyber–physical systems: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5103–5115, 2020.

[21] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Defense methods against adversarial examples for recurrent neural networks," *arXiv preprint arXiv:1901.09963*, 2019.

[22] K. Rajaratnam *et al.*, "Speech coding and audio preprocessing for mitigating and detecting audio adversarial examples on automatic speech recognition," 2018.

[23] J. Hao, R. J. Piechocki, D. Kaleshi, W. H. Chin, and Z. Fan, "Sparse malicious false data injection attacks and defense mechanisms in smart grids," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1–12, 2015.

[24] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "{CommanderSong}: A systematic approach for practical adversarial voice recognition," in *27th USENIX security symposium (USENIX security 18)*, pp. 49–64, 2018.

[25] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 3–14, 2017.

[26] O. Kosut *et al.*, "Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures," in *2010 first IEEE international conference on smart grid communications*, pp. 220–225, IEEE, 2010.

[27] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[28] Z. Yang, B. Li, P.-Y. Chen, and D. Song, "Towards mitigating audio adversarial perturbations (2018)," in *URL https://openreview. net/forum*, 2018.

[29] E. J. Santana *et al.*, "Detecting and mitigating adversarial examples in regression tasks: A photovoltaic power generation forecasting case study," *Information*, vol. 12, no. 10, p. 394, 2021.

[30] W. Yan *et al.*, "Attack detection for securing cyber physical systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8471–8481, 2019.

[31] Y. Zhou *et al.*, "A secure control learning framework for cyber-physical systems under sensor attacks," in *2019 ACC*, pp. 4280–4285, IEEE, 2019.

[32] N. Papernot *et al.*, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

[33] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE, 2016.

[34] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14410–14430, 2018.

[35] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[36] T. Bai *et al.*, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.

[37] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, "Challenges and opportunities in securing the industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 2985–2996, 2020.

[38] "Am64x processor module for industrial iot, robotics and comms." https://www.eenewseurope.com/en/am64x-processor-module-for-industrial-iot-robotics-and-comms/, 2021.

[39] J. Cheng, Y. Yang, N. Hu, Z. Cheng, and J. Cheng, "A noise reduction method based on adaptive weighted symplectic geometry decomposition and its application in early gear fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 149, p. 107351, 2021.

[40] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, pp. 139–159, 2009.

[41] J. Morris *et al.*, "Multi-label hd classification in 3d flash," in *VLSI-SOC*, pp. 10–15, IEEE, 2020.

[42] O. J. Räsänen and J. P. Saarinen, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1878–1889, 2015.

[43] M. Imani *et al.*, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *ICRC*, pp. 1–8, IEEE, 2017.

[44] A. Moin *et al.*, "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition," *Nature Electronics*, vol. 4, no. 1, pp. 54–63, 2021.

[45] R. Zhao *et al.*, "Deep learning and its applications to machine health monitoring: A survey," *arXiv preprint arXiv:1612.07640*, 2016.

[46] W. Z. Khan *et al.*, "Industrial internet of things: Recent advances, enabling technologies and open challenges," *Computers & Electrical Engineering*, vol. 81, p. 106522, 2020.

[47] S. M. Lee, D. Lee, and Y. S. Kim, "The quality management ecosystem for predictive maintenance in the industry 4.0 era," *International Journal of Quality Innovation*, vol. 5, no. 1, pp. 1–11, 2019.

[48] O. Gungor, J. Garnier, T. S. Rosing, and B. Aksanli, "Lenard: Lightweight ensemble learner for medium-term electricity consumption prediction," in *IEEE SmartGridComm*, pp. 1–6, IEEE, 2020.

[49] A. Saxena *et al.*, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *IEEE ICPHM*, pp. 1–9, IEEE, 2008.

[50] T. Tinga and R. Loendersloot, "Physical model-based prognostics and health monitoring to enable predictive maintenance," in *Predictive Maintenance in Dynamic Systems*, pp. 313–353, Springer, 2019.

[51] M. Paolanti *et al.*, "Machine learning approach for predictive maintenance in industry 4.0," in *2018 14th IEEE/ASME MESA*, pp. 1–6.

[52] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using iot sensor data," in *2017 ICSESS*, pp. 87–90, IEEE, 2017.

[53] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: a survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[54] L. Guo *et al.*, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[55] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[56] S. Zheng *et al.*, "Long short-term memory network for remaining useful life estimation," in *IEEE ICPHM*, pp. 88–95, IEEE, 2017.

[57] D. Bruneo and F. De Vita, "On the use of lstm networks for predictive maintenance in smart industries," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 241–248, IEEE, 2019.

[58] X. Bampoula, G. Siaterlis, N. Nikolakis, and K. Alexopoulos, "A deep learning model for predictive maintenance in cyber-physical production systems using lstm autoencoders," *Sensors*, vol. 21, no. 3, p. 972, 2021.

[59] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6069–6078.

[60] A. Al-Dulaimi *et al.*, "Hybrid deep neural network model for remaining useful life estimation," in *IEEE ICASSP*, pp. 3872–3876, IEEE, 2019.

[61] O. Güngör, B. Akşanlı, and R. Aydoğan, "Algorithm selection and combining multiple learners for residential energy prediction," *Future Generation Computer Systems*, vol. 99, pp. 391–400, 2019.

[62] S. Gu, R. Cheng, and Y. Jin, "Multi-objective ensemble generation," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 234–245, 2015.

[63] Z. Li, K. Goebel, and D. Wu, "Degradation modeling and remaining useful life prediction of aircraft engines using ensemble learning," *Journal of Eng. for Gas Turbines and Power*, vol. 141, no. 4, 2019.

[64] J. Shi *et al.*, "Remaining useful life prediction of bearings using ensemble learning," *Journal of Computing and Information Science in Engineering*, pp. 1–35, 2020.

[65] Z. Li *et al.*, "An ensemble learning-based prognostic approach with degradation-dependent weights for remaining useful life prediction," *Reliability Engineering & System Safety*, vol. 184, pp. 110–122, 2019.

[66] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[67] K. Greff *et al.*, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232.

[68] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint*, 2014.

[69] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE CVPR*, pp. 1251–1258, 2017.

[70] A. v. d. Oord *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[71] L. Jayasinghe, T. Samarasinghe, C. Yuen, J. C. N. Low, and S. S. Ge, "Temporal convolutional memory networks for remaining useful life estimation of industrial machinery," *arXiv preprint:1810.05644*, 2018.

[72] X. Zhang *et al.*, "Remaining useful life estimation based on a new convolutional and recurrent neural network," in *IEEE CASE*, pp. 317–322, IEEE, 2019.

[73] K. Lee *et al.*, "Cnn and gru combination scheme for bearing anomaly detection in rotating machinery health monitoring," in *IEEE ICKII*, pp. 102–105, IEEE, 2018.

[74] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Multipath parallel hybrid deep neural networks framework for remaining useful life estimation," in *IEEE ICPHM*, pp. 1–7, IEEE, 2020.

[75] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[76] E. Ramasso, "Investigating computational geometry for failure prognostics.," 2014.

[77] J. Wang *et al.*, "Remaining useful life estimation in prognostics using deep bidirectional lstm neural network," in *IEEE PHM*, pp. 1037–1042, IEEE, 2018.

[78] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 ICRA*, pp. 284–289, IEEE, 2004.

[79] M. ApS, *The MOSEK Optimization Toolbox for MATLAB manual. Version 9.2*, 2020.

[80] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[81] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[82] S. M. LaValle *et al.*, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[83] M. Ardakani, M. Askarian, A. Shokry, G. Escudero, M. Graells, and A. Espuña, "Optimal features selection for designing a fault diagnosis system," in *Computer Aided Chemical Engineering*, vol. 38, pp. 1111–1116, Elsevier, 2016.

[84] Z. Chen, A. Mauricio, W. Li, and K. Gryllias, "A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks," *Mechanical Systems and Signal Processing*, vol. 140, p. 106683, 2020.

[85] B. Zhang, Y. Miao, J. Lin, and Y. Yi, "Adaptive maximum second-order cyclostationarity blind deconvolution and its application for locomotive bearing fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 158, p. 107736, 2021.

[86] X. Wang, D. Mao, and X. Li, "Bearing fault diagnosis based on vibro-acoustic data fusion and 1d-cnn network," *Measurement*, vol. 173, p. 108518, 2021.

[87] A. Zhang, S. Li, Y. Cui, W. Yang, R. Dong, and J. Hu, "Limited data rolling bearing fault diagnosis with few-shot learning," *IEEE Access*, vol. 7, pp. 110895–110904, 2019.

[88] "Case western reserve university bearing data center website." https://csegroups.case.edu/bearingdatacenter/pages/project-history.

[89] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2005.

[90] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[91] D. Wu, F. Zhu, and L. Shao, "One shot learning gesture recognition from rgbd images," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 7–12, IEEE, 2012.

[92] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015.

[93] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," *arXiv preprint arXiv:1606.04080*, 2016.

[94] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," *arXiv preprint arXiv:1710.11622*, 2017.

[95] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint arXiv:1605.06065*, 2016.

[96] Y. Yang, H. Wang, Z. Liu, and Z. Yang, "Few-shot learning for rolling bearing fault diagnosis via siamese two-dimensional convolutional neural network," in *PHM-2020 Jinan*, pp. 373–378, IEEE, 2020.

[97] Q. Fang and D. Wu, "Ans-net: anti-noise siamese network for bearing fault diagnosis with a few data," *Nonlinear Dynamics*, pp. 1–18, 2021.

[98] S. Dixit and N. K. Verma, "Intelligent condition-based monitoring of rotary machines with few samples," *IEEE Sensors Journal*, vol. 20, no. 23, pp. 14337–14346, 2020.

[99] J. Wu, Z. Zhao, C. Sun, R. Yan, and X. Chen, "Few-shot transfer learning for intelligent fault diagnosis of machine," *Measurement*, vol. 166, p. 108202, 2020.

[100] O. Gungor, T. S. Rosing, and B. Aksanli, "Opelrul: Optimally weighted ensemble learner for remaining useful life prediction," in *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–8, IEEE, 2021.

[101] C. Zhang, Y. He, B. Du, L. Yuan, B. Li, and S. Jiang, "Transformer fault diagnosis method using iot based monitoring system and ensemble machine learning," *Future Generation Computer Systems*, vol. 108, pp. 533–545, 2020.

[102] X. Li, H. Jiang, R. Wang, and M. Niu, "Rolling bearing fault diagnosis using optimal ensemble deep transfer network," *Knowledge-Based Systems*, vol. 213, p. 106695, 2021.

[103] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[104] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," *Sensors*, vol. 17, no. 2, p. 425, 2017.

[105] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, "A siamese long short-term memory architecture for human re-identification," in *European conference on computer vision*, pp. 135–153, Springer, 2016.

[106] G. James, "Majority vote classifiers: theory and applications," 1998.

[107] G. R. Mode, P. Calyam, and K. A. Hoque, "False data injection attacks in internet of things and deep learning enabled predictive analytics," *arXiv preprint arXiv:1910.01716*, 2019.

[108] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *International Conference on Machine Learning*, pp. 4970–4979, PMLR, 2019.

[109] A. Mirzaeian, M. Sabokrou, M. Khalooei, J. Kosecka, H. Homayoun, T. Mohsening, and A. Sasan, "Learning diverse latent representations for improving the resilience to adversarial attacks," *arXiv e-prints*, pp. arXiv–2006, 2020.

[110] M. Löwe, J. Villareale, E. Freed, A. Sladek, J. Zhu, and S. Risi, "Dealing with adversarial player strategies in the neural network game innk through ensemble learning," in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, pp. 1–10, 2021.

[111] K. L. Wong, M. Bosello, R. Tse, C. Falcomer, C. Rossi, and G. Pau, "Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles," in *Proceedings of the Conference on Information Technology for Social Good*, pp. 85–90, 2021.

[112] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[113] A. Kurakin, I. Goodfellow, S. Bengio, *et al.*, "Adversarial examples in the physical world," 2016.

[114] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

[115] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[116] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, *et al.*, "Adversarial attacks and defences competition," in *The NIPS'17 Competition: Building Intelligent Systems*, pp. 195–231, Springer, 2018.

[117] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2018.

[118] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[119] H. I. Fawaz *et al.*, "Adversarial attacks on deep neural networks for time series classification," in *IJCNN*, pp. 1–8, IEEE, 2019.

[120] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[121] O. Gungor, T. Rosing, and B. Aksanli, "Enfes: Ensemble few-shot learning for intelligent fault diagnosis with limited data," in *2021 IEEE Sensors*, pp. 1–4, IEEE, 2021.

[122] O. Gungor, T. Rosing, and B. Aksanli, "Stewart: Stacking ensemble for white-box adversarial attacks towards more resilient data-driven predictive maintenance," *Computers in Industry*, vol. 140, p. 103660, 2022.

[123] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[124] H. Yang, J. Zhang, H. Dong, N. Inkawhich, A. Gardner, A. Touchet, W. Wilkes, H. Berry, and H. Li, "Dverge: diversifying vulnerabilities for enhanced robust generation of ensembles," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5505–5515, 2020.

[125] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," *arXiv preprint arXiv:1901.09981*, 2019.

[126] Z. Yang, L. Li, X. Xu, S. Zuo, Q. Chen, P. Zhou, B. Rubinstein, C. Zhang, and B. Li, "Trs: Transferability reduced ensemble via promoting gradient diversity and model smoothness," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17642–17655, 2021.

[127] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[128] J. Lei, C. Liu, and D. Jiang, "Fault diagnosis of wind turbine based on long short-term memory networks," *Renewable energy*, vol. 133, pp. 422–432, 2019.

[129] Y. Tao, X. Wang, R.-V. Sánchez, S. Yang, and Y. Bai, "Spur gear fault diagnosis using a multilayer gated recurrent unit approach with vibration signal," *IEEE Access*, vol. 7, pp. 56880–56889, 2019.

[130] S. Bhambri *et al.*, "A survey of black-box adversarial attacks on computer vision models," *arXiv preprint arXiv:1912.01667*, 2019.

[131] A. Rahimi, A. Tchouprina, P. Kanerva, J. d. R. Millán, and J. M. Rabaey, "Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1958–1969, 2020.

[132] Z. Zou, H. Alimohamadi, F. Imani, Y. Kim, and M. Imani, "Spiking hyperdimensional network: Neuromorphic models integrated with memory-inspired framework," *arXiv preprint arXiv:2110.00214*, 2021.

[133] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*, pp. 445–456, IEEE, 2017.

[134] F. Yang and S. Ren, "Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers," *arXiv preprint arXiv:2006.05594*, 2020.

[135] W. Chen and H. Li, "Adversarial attacks on voice recognition based on hyper dimensional computing," *Journal of Signal Processing Systems*, vol. 93, pp. 709–718, 2021.

[136] H. Moraliyage, S. Kahawala, D. De Silva, and D. Alahakoon, "Evaluating the adversarial robustness of text classifiers in hyperdimensional computing," in *2022 15th International Conference on Human System Interaction (HSI)*, pp. 1–8, IEEE, 2022.

[137] D. Ma, J. Guo, Y. Jiang, and X. Jiao, "Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 391–396, IEEE, 2021.

[138] R. Thapa, D. Ma, and X. Jiao, "Hdxplore: Automated blackbox testing of brain-inspired hyperdimensional computing," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 90–95, IEEE, 2021.

[139] R. Wang and X. Jiao, "Poisonhd: poison attack on brain-inspired hyperdimensional computing," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 298–303, IEEE, 2022.

[140] W. A. Smith and R. B. Randall, "Rolling element bearing diagnostics using the case western reserve university data: A benchmark study," *Mechanical Systems and Signal Processing*, vol. 64, pp. 100–131, 2015.

[141] A. Shenfield and M. Howarth, "A novel deep learning model for the detection and identification of rolling element-bearing faults," *Sensors*, vol. 20, no. 18, p. 5112, 2020.

[142] M. A. Ferrag *et al.*, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.

[143] J. Goh *et al.*, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering*, pp. 140–145, IEEE, 2017.

[144] M. A. Teixeira *et al.*, "Scada system testbed for cybersecurity research using machine learning approach," *Future Internet*, vol. 10, no. 8, p. 76, 2018.

[145] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, pp. 72–83, 2018.

[146] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, 2020.

[147] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2021.

[148] M. Imani *et al.*, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[149] P. Arora *et al.*, "Evaluation of machine learning algorithms used on attacks detection in industrial control systems," *Journal of The Institution of Engineers (India): Series B*, vol. 102, no. 3, pp. 605–616, 2021.

[150] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*, pp. 420–434, Springer, 2001.

[151] Z. Chen and B. Liu, *Lifelong machine learning*, vol. 1. Springer, 2018.

[152] V. Belle and I. Papantonis, "Principles and practice of explainable machine learning," *Frontiers in big Data*, p. 39, 2021.

[153] Y. Gong, B. Li, C. Poellabauer, and Y. Shi, "Real-time adversarial attacks," *arXiv preprint arXiv:1905.13399*, 2019.