Real-Time Machine Learning for Quickest Detection

Yongxin Liu
Real-Time Machine Learning for Quickest Detection

by

Yongxin Liu

A dissertation submitted to the Faculty of Embry-Riddle Aeronautical University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science

Embry-Riddle Aeronautical University
Daytona Beach, Florida
June 2021
Real-time Machine Learning for Quickest Detection

by Yongxin Liu

This dissertation was prepared under the direction of the candidate’s Dissertation Committee Chair, Dr. Houbing Song, and has been approved by the members of the dissertation committee. It was submitted to the College of Engineering and accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Electrical Engineering and Computer Science.
“Stay Hungry. Stay Foolish.”

Steven Jobs
Abstract

Safety-critical Cyber-Physical Systems (CPS) require real-time machine learning for control and decision making. One promising solution is to use deep learning to discover useful patterns for event detection from heterogeneous data. However, deep learning algorithms encounter challenges in CPS with assurability requirements: 1) Decision explainability, 2) Real-time and quickest event detection, and 3) Time-efficient incremental learning.

To address these obstacles, I developed a real-time Machine Learning Framework for Quickest Detection (MLQD). To be specific, I first propose the zero-bias neural network, which removes decision bias and preferabilities from regular neural networks and provides an interpretable decision process. Second, I discover the latent space characteristic of the zero-bias neural network and the method to mathematically convert a Deep Neural Network (DNN) classifier into a performance-assured binary abnormality detector. In this way, I can seamlessly integrate the deep neural networks’ data processing capability with Quickest Detection (QD) and provide real-time sequential event detection paradigm. Thirdly, after discovering that a critical factor that impedes the incremental learning of neural networks is the concept interference (confusion) in latent space, and I prove that to minimize interference, the concept representation vectors (class fingerprints) within the latent space need to be organized orthogonally and I invent a new incremental learning strategy using the findings, I facilitate deep neural networks in the CPS to evolve efficiently without retraining. All my algorithms are evaluated on real-world applications, ADS-B (Automatic Dependent Surveillance Broadcasting) signal identification, and spoofing detection in the aviation communication system. Finally, I discuss the current trends in MLQD and conclude this dissertation by presenting the future research directions and applications.

As a summary, the innovations of this dissertation are as follows: i) I propose the zero-bias neural network, which provides transparent latent space characteristics, I apply it to solve the wireless device identification problem. ii) I discover and prove the orthogonal memory organization mechanism in artificial neural networks and apply this mechanism in time-efficient incremental learning. iii) I discover and mathematically prove the converging point theorem, with which we can predict the latent space topological characteristics and estimate the topological maturity of neural networks. iv) I bridge the gap between machine learning and quickest detection with assurable performance.
Acknowledgements

First of all, I would like to thank my Ph.D. advisor, Dr. Houbing Song, for supporting me during the past four years. Dr. Song is a professional full of knowledge and wisdom, and he is one of the smartest people I know. He is a great advisor who always leads me with patience and a good and warm friend. I am very fortunate to have worked with Dr. Song, and I hope that I could be as lively, enthusiastic, and energetic as him and someday be able to flourish in academia works as he does.

I want to thank all my committee members for years of guidance and collaboration in my Ph.D. life. I am grateful that Dr. Radu Babiceanu provided me kind help with my career and his fantastic mentoring of system engineering related topics. Great thanks to Dr. Tianyu Yang for his excellent teaching in several core classes and the Air-Force proposal writing collaboration. Moreover, Dr. Yang gave generous support for my future career. I also want to thank Dr. Hong Liu and Dr. Jiawei Yuan for their valuable inputs to my research in machine learning and big data. Moreover, Dr. Yier Jin and Dr. Qi Cheng have provided me a lot of useful advices in deepening my dissertation.

I also need to express my gratitude to all SONG Lab members who gave me helps in academia. Especially, I appreciate Jian Wang, Yushan Jiang, Kai Zhang and Bowen Chen for their collaboration on many machine learning projects and algorithm development. Moreover, I want to thank Dr. Dahai Liu for the inspiring discussions we had. Many Thanks to Dr. Timothy Wilson for the advice on my faculty position interviews. Furthermore, I also want to thank my Ph.D. financial support agents: the Center for Advanced Transportation Mobility (CATM), USDOT Grant #69A3551747125 and through the National Science Foundation under Grant No. 1956193.
Finally and most importantly, I want to thank my father, Zhilong Liu, and my mother, Xia Zhang, for their selfless support so that I can always pursue the path that I love. I also want to thank my parents for always warming my heart with unconditional love and standing by my side when troubles come. Moreover, I also want to thank all other families who always believe in me and support me. I want to dedicate this milestone of my life to them for their love and support.
Contents

Abstract iii

Acknowledgements iv

Contents vi

List of Figures ix

List of Tables xi

1 Introduction 1

1.1 Background and Motivation 1

1.2 Proposed Methodology 3

1.3 Significance 4

1.4 Organization of the Dissertation 5

2 Literature Review 6

2.1 Abnormality detection for neural networks 6

2.1.1 Statistical Modeling 7

2.1.2 Reconstruction Approaches 8

2.1.3 Predictive Approaches 9

2.2 Incremental learning 11

2.3 Quickest Event Detection 13

3 Machine Learning for Quickest Event Detection: A Novel Framework 18

3.1 Zero-bias Neural Network 20

3.2 Deep Learning Enabled Quickest Event Detection 21

3.3 Time Efficient Incremental Learning 22

4 Zero-Bias Deep Learning with Application in Wireless Device Identification 23

4.1 Zero-bias Dense Layer for Deep Learning 24

4.1.1 Formalization of Zero-bias Neural Network 24

4.1.2 Zero-bias Neural Network for Abnormality Detection: A Threshold Based Approach 29

4.1.3 Error Tendency Analysis for Zero-bias Neural Network 30

vi
4.1.4 Incremental Learning for Zero-Bias Neural Network: A Direct Approach ......................................................... 32

4.2 Zero-bias Deep Neural Network for Wireless Device Identification ................................................................. 34
4.2.1 Introduction .................................................................................................................................................. 34
4.2.2 Related Work ............................................................................................................................................. 36
4.2.2.1 Specific feature-based approaches ...................................................................................................... 37
4.2.2.2 Deep Learning for Wireless Transmitter Identification .................................................................. 38
4.2.3 Problem definition ....................................................................................................................................... 39
4.2.4 Methodology ............................................................................................................................................... 40
4.2.4.1 Baseband demodulation .................................................................................................................... 40
4.2.4.2 Feature extraction .............................................................................................................................. 41
4.2.5 Validations ................................................................................................................................................. 43
4.2.5.1 Validation I: Trainability of Zero-bias Neural Network .................................................................. 43
4.2.5.2 Validation II: Unknown device identification .................................................................................. 45
4.2.5.3 Validation III: Incremental learning ................................................................................................. 48
4.2.5.4 Validation IV: Error Tendency Analysis ........................................................................................... 53

4.3 Further Discussion ............................................................................................................................................. 54
4.3.1 Multi-kernel Classification Using Zero-bias Neural Network .................................................................. 54

4.4 Concluding Remark .......................................................................................................................................... 56

5 Orthogonal Memory Organization for Time Efficient Incremental Learning in Wireless Device Identification 57
5.1 Topological Characteristics of Class Representation Vectors ............................................................................ 58
5.1.1 Optimal Separation of Class Representation Vectors ................................................................................ 59
5.1.2 Conflict of Class Representation Vectors and Orthogonal Memory Representation .................................. 63
5.1.2.1 Catastrophic Forgetting and Degree of Conflicts ........................................................................... 63
5.1.2.2 Orthogonal Memory Organization and Proof of Optimality .......................................................... 64
5.2 Channel Separation Incremental Learning (CSIL) ............................................................................................ 69

5.3 Application of CSIL in Wireless Device Identification ..................................................................................... 73
5.3.1 Introduction ............................................................................................................................................... 73
5.3.2 Evaluation Dataset .................................................................................................................................... 74
5.3.3 Performance Evaluation .......................................................................................................................... 75
5.3.4 Ablation Analysis ...................................................................................................................................... 78

5.4 Further Discussion: Local Degree of Conflict During Incremental Learning ......................................................... 79
5.5 Concluding Remark .......................................................................................................................................... 82

6 Deep Learning Enabled Quickest Event Detection with Application in Identity Spoofing Detection 84
6.1 Regional Association Characteristics of Zero-bias Neural Network ................................................................. 85
6.1.1 Abnormality detection in DNN with zero-bias dense layer ........................................................................ 87
6.1.1.1 Deriving Abnormality Detectors from Existing DNN Classifiers .................................................... 87
6.1.1.2 Theoretic Performance Analysis of the Single-Shot Binary Abnormality Detector .......................... 91

6.2 Zero-bias DNN for Quickest Abnormal Event Detection ..................................................................................... 95
6.2.1 Sequential Formalization And Detectability .......................... 95
6.2.2 Quickest Detection Algorithm ........................................... 96
6.3 Zero-bias Deep Learning for Quickest Identity Spoofing Detection .... 98
6.3.1 Introduction ............................................................ 98
6.3.2 Problem definition ..................................................... 100
6.3.3 Performance Evaluation ............................................... 102
   6.3.3.1 Decision Boundaries in Real-World Zero-Bias DNN ........ 103
   6.3.3.2 Quickest Abnormal Event Detection with Zero-Bias
            DNNs ............................................................. 105
6.4 Concluding Remark ...................................................... 107

7 Concluding Remark and Future Work ................................. 109
7.1 Concluding Remark ...................................................... 109
7.2 Future Works ........................................................... 111
   7.2.1 Unsupervised Deep Learning Driven Quickest Detection Algo-
         rithms .......................................................... 111
   7.2.2 Incremental Learning With Adaptive Network Expansion ...... 111

Bibliography ................................................................. 113
List of Figures

2.1 Transfer learning versus incremental learning ........................................ 11
2.2 Categorization of event detection approaches ........................................ 14
2.3 Categories of quickest detection on known post-change distribution ....... 15
2.4 Quickest detection with the unknown post-change distribution .............. 17

3.1 Theoretic framework of machine learning for quickest detection .......... 18
3.2 Related research in this dissertation .................................................... 20

4.1 Typical architecture of deep neural network ........................................... 24
4.2 Data flow of zero-bias layer .................................................................. 25
4.3 Transformation from a regular dense layer to a zero-bias dense layer ....... 29
4.4 Relation of fingerprint vectors and feature vectors ............................... 30
4.5 Distribution of maximum similarity scores ............................................ 30
4.6 Fingerprint distance matrix of Minst example ...................................... 31
4.7 Noise extraction on typical signals ....................................................... 42
4.8 Correlation coefficients of pseudo noise/ .............................................. 43
4.9 Collection of ADS-B signals for model validation .................................... 44
4.10 Geographic distribution of aircraft transponders .................................... 44
4.11 Deep neural network architecture for wireless transmitter identification . 45
4.12 Comparison of training performance .................................................... 45
4.13 Validation accuracy in terms of training data size for each transmitter ... 46
4.14 Performance of Threshold based anomaly detection in neural network with zero-bias layer ............................................................ 46
4.15 Performance of Threshold based anomaly detection in conventional neural network .................................................................................. 47
4.16 Performance of anomaly detection using one-class support vector machine .......................................................... 47
4.17 Transponders’ appearance frequency ................................................... 48
4.18 Performance comparison of zero-bias layer and regular dense layer for incremental learning .............................................................. 49
4.19 Effects of L2 regularization ................................................................... 51
4.19 Effects of L2 regularization ................................................................... 52
4.20 Compare of numerical stability during incremental learning ................. 53
4.21 Cosine similarity matrix of class fingerprint before and after incremental learning .................................................................................... 53
4.22 The architecture of the dual-kernel classification neural network .......... 54
4.23 Training of the dual-kernel neural classifier .......................................... 55
4.24 Unnecessary decomposition of fingerprints ........................................... 56
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>A comparison of regular and zero-bias DNN considering degree of conflicts and training accuracy.</td>
</tr>
<tr>
<td>5.2</td>
<td>Distance matrix of fingerprints after inserting new fingerprints and fine-tuning.</td>
</tr>
<tr>
<td>5.3</td>
<td>Lower triangular proportion of fingerprint cosine similarity matrix after incremental learning.</td>
</tr>
<tr>
<td>5.4</td>
<td>Channel separation for incremental learning.</td>
</tr>
<tr>
<td>5.5</td>
<td>Distance matrix of fingerprints after regular training and CSIL.</td>
</tr>
<tr>
<td>5.6</td>
<td>Degree of Conflict among IL algorithms.</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of incremental learning strategies for wireless device identification.</td>
</tr>
<tr>
<td>5.8</td>
<td>Comparison influential factors in CSIL during incremental learning.</td>
</tr>
<tr>
<td>5.9</td>
<td>Local Degree of Conflict of finetune with weight consolidation.</td>
</tr>
<tr>
<td>5.10</td>
<td>Local Degree of Conflict of regular EWC with weight consolidation.</td>
</tr>
<tr>
<td>5.11</td>
<td>Local Degree of Conflict of regular EWC with weight consolidation.</td>
</tr>
<tr>
<td>5.12</td>
<td>Local Degree of Conflict of CSIL.</td>
</tr>
<tr>
<td>6.1</td>
<td>3D unit hyperspherical surface in zero-bias DNNs.</td>
</tr>
<tr>
<td>6.2</td>
<td>Stacked diagrams of: a) Voronoi graph of remapped fingerprints. b) Class decision boundaries. c) remapped validation and abnormal data feature vectors. The two subfigures are taken at different training stages with all data points projected to a 2D space using t-SNE algorithm [1].</td>
</tr>
<tr>
<td>6.3</td>
<td>Stacked diagrams of: a) Voronoi graph of remapped fingerprints. b) Class decision boundaries. c) remapped validation and abnormal data feature vectors. Data are projected to a 2D space using t-SNE algorithm [1].</td>
</tr>
<tr>
<td>6.4</td>
<td>Relation of normal and abnormal data.</td>
</tr>
<tr>
<td>6.5</td>
<td>The coverage ration per class and maximum number of distinguishable class in zero-bias DNN.</td>
</tr>
<tr>
<td>6.6</td>
<td>Classification errors in zero-bias DNN, feature vector C, D, E and F are erroneously projected into governing regions of wrong fingerprints.</td>
</tr>
<tr>
<td>6.7</td>
<td>Range of true-positive and false-positive rates.</td>
</tr>
<tr>
<td>6.8</td>
<td>System model of zero-bias deep learning enabled quick and reliable abnormality detection in IoT.</td>
</tr>
<tr>
<td>6.9</td>
<td>Identity spoofing in aviation communication systems.</td>
</tr>
<tr>
<td>6.10</td>
<td>Stacked Voronoi diagram of dimension-reduced fingerprints and validation set two different scenarios. Fingerprints and feature vectors are projected to a 2D space using t-SNE algorithm [1].</td>
</tr>
<tr>
<td>6.11</td>
<td>Performance of the converted abnormality detector.</td>
</tr>
<tr>
<td>6.12</td>
<td>Abnormal event detection latencies.</td>
</tr>
<tr>
<td>6.13</td>
<td>Distribution of abnormal event detection latencies.</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Methods for open set recognition ........................................ 7
2.2 Comparison of incremental learning strategies .......................... 13
4.1 Comparisons of approaches for incremental learning ................. 49
5.1 A comparison of the DoC of DNN models before and after incremental learning ............................................................... 64
5.2 A comparison of the degree of conflicts of DNN models with different training strategies ......................................................... 72
5.3 Ablation analysis of CSIL. All metrics are in percentage .............. 78
6.1 Description of dataset .......................................................... 103
For Security and Optimization for Networked Globe Laboratory (SONG Lab) at Embry-Riddle Aeronautical University
Chapter 1

Introduction

In this chapter, I will first introduce the motivation of my research: the obstacles that deep learning is facing when they are applied in safety-critical and latency-constrained CPS. I will then introduce the solution briefly. Finally, I will highlight the significance and contribution of the research.

1.1 Background and Motivation

A Learning-enabled CPS (LE-CPS) is defined as a cyber-physical system composed of one or more Learning-enabled Components (LECs). A LEC is a component whose behavior is driven by “background knowledge” acquired and updated through a “learning process”. A promising solution to implement LECs is Deep Learning (DL), particularly, Deep Neural Networks (DNNs), in which unified frameworks are provided to simplify pattern recognition algorithms. For example, in my previous work of detecting rogue UAS (Unmanned Aerial Systems), deep learning is employed to recognize the appearance of drones by identifying their signals [2, 3].
Chapter 1. Introduction

Deployment and broader proliferation of LE-CPS in safety-assured and latency-constrained applications are still challenging and controversial. Several factors impede the deployment and adoption of DL:

- **Explainability**: Deep neural networks (DNNs) are essential building blocks for LE-CPS. However, users have little knowledge of how a neural network can associate a certain input with a specific label and how a neural network behaves when encountering an unseen novel input. In extreme cases, neural networks can associate irrelevant inputs to known labels confidently.

- **Performance assurability**: The learning-enabled CPS should be able to respond to known or unknown events with the lowest assurable latency. The system should respond properly to unknown events and make decisions on known events with the lowest latency.

- **Evolvability**: Conventional deep neural networks encounter catastrophic forgetting when they are trained for new tasks. It is inefficient to retain old training samples during the life cycle of CPS. Learning-enabled CPS are supposed to be evolving in a controllable manner to adapt to operational variations.

Therefore, to deploy deep learning algorithms to safety-critical CPS, there is a need to develop an enhanced framework integrating real-time machine learning with quickest detection. Simultaneously, there is another need to enhance the dependability and adaptability of deep learning algorithms, particularly deep neural networks.
1.2 Proposed Methodology

In this dissertation, I will develop a holistic framework for LE-CPS in safety-critical and latency-constrained applications. Specifically, my works are divided into three parts: a) Enhancement of deep neural networks for explainability and abnormality detection. b) Quickest detection for deep neural networks. c) A time-efficient incremental learning algorithm for evolving CPS.

First, I thoroughly analyze the mathematical process of dense layers in deep neural networks. I discover that the last dense layer is only a biased single kernel nearest neighbor matching process using cosine similarity. Accordingly, I design the zero-bias dense layer to replace the last dense layers (classification dense layer) in deep neural networks to increase their explainability. I have applied this improvement of deep neural network for physical layer emitter identification in ADS-B systems. My solution maintains equivalent accuracy and surpasses existing solutions in terms of automated anomaly detection and incremental learning.

Second, I discover that deep neural networks can use the maximum confidence value to assign known and abnormal inputs to two different probability distributions. This phenomenon not only provides the potential to design an anomaly detector but also provides the chance to incorporate deep learning algorithms with the quickest event detection theory. I analyze the latent space characteristics of neural networks and find that the response characteristics of neural networks can be modeled by two Bernoulli distributions. I propose to leverage parametric event detection with the CUSUM (cumulative sum control chart) enabled generalized likelihood ratio test algorithm to detect the emerging point with the shortest delay under given false alarm constraints.
Third, deep neural networks in safety-critical CPS may be subject to the situation of life-long learning. They should be capable of incremental learning new labeled data without forgetting. Existing methodologies use knowledge replay, network expansion, or critical connection protection to prevent catastrophic forgetting. However, knowledge replay may not be feasible since I can not retain generative models for all old data. Furthermore, the critical connection protection strategy encounters numerical instability for regularized training. To solve the problem, I discovered and mathematically prove that organizing concept representation vectors in the latent space in a mutually orthogonal manner, is an optimal way that minimizes mutual interference. And the essence of catastrophic forgetting is concept confusion or interference. I then invented a novel incremental learning algorithm, Channel Separation Incremental Learning (CSIL), which allows neural networks to organize concepts in different stages orthogonally.

1.3 Significance

Deep learning has been employed as an important building block of learning-enabled CPS. Other researches focus merely on accuracy. My research focuses on how to enable deep learning in safety-critical and performance-assured CPS. My highlights are in three folds: First, my work explicitly explores the mathematic essence and the techniques on how to prevent deep neural networks from making risky decisions. Second, I enable deep learning to be used in performance-assured CPS for real-time event detection, I provide a novel method to seamlessly bridge quickest event detection with Deep Learning. Third, I discover and mathematically prove that organizing concept representation vectors in the latent space mutually orthogonal is an optimal way that minimizes mutual interference. Finally, based on the discovery of orthogonal memory representation, I design a novel
incremental learning algorithm, Channel Separation Incremental Learning (CSIL), which beats the existing incremental learning algorithms with or without historical rehearsal data.

1.4 Organization of the Dissertation

Finally, the remainder of this dissertation is structured as follows: In chapter 2, I will give a comprehensive review of the enabling technologies of event detection in CPS. And then, Chapter 3 will introduce a novel machine learning for quickest detection framework with all theoretical findings in this dissertation. Next, the zero-bias neural network with its internal mechanisms will be discussed in Chapter 4. After that, I will present my findings on the mathematical mechanisms of catastrophic forgetting and discuss the time-efficient channel separation incremental learning algorithm in Chapter 5. Moreover, the zero-bias neural network enabled quickest event detection algorithm will be discussed in Chapter 6. Finally, a conclusion of the dissertation and a discussion of the future directions will be presented in Chapter 7.
Chapter 2

Literature Review

Deep learning has been utilized in LE-CPS. However, as in other domains, it encounters several problems, such as dependability, interpretability, incremental learning, and quickest event detection. Researches in CPS rarely cover these problems and they are becoming emerging issues in the proliferation of learning-based systems.

2.1 Abnormality detection for neural networks

One critical problem for learning is that neural classifiers only recognize pre-trained objects but can erroneously associate irrelevant objects with existing labels. From the perspective of Artificial Intelligence (AI), this issue is categorized as the Open Set Recognition or Abnormality Detection problem [4, 5]. The Abnormality Detection problem and the taxonomy of existing approaches are given in Table 2.1.
Table 2.1: Methods for open set recognition

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
<th>Complexity</th>
<th>Memory</th>
<th>Pros &amp; Cons</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAN</td>
<td>Use the discriminator from GAN model as an outlier detector.</td>
<td>High¹</td>
<td>Depends on final network</td>
<td>• Can catch deep latent features.</td>
<td>[6, 7]</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>Train a deep Autoencoder on known signals and use its reconstruction error to judge outliers.</td>
<td>High¹</td>
<td>Depends on final network</td>
<td>• Can catch deep latent features.</td>
<td>[8, 9]</td>
</tr>
<tr>
<td>Statistic</td>
<td>Measure the possibility of whether a signal or its fingerprint is generated by a given distribution (form by known transmitters)</td>
<td>Low</td>
<td>Low</td>
<td>• Provide explainable results.</td>
<td>[10–12]</td>
</tr>
<tr>
<td>Clustering</td>
<td>Perform clustering analysis on known signals' fingerprints to judge whether it is in identical cluster as known ones.</td>
<td>Medium²</td>
<td>Depends on the number of existing fingerprints</td>
<td>• Provide explainable results.</td>
<td>[10, 13]</td>
</tr>
</tbody>
</table>

¹ Needs to specify both network architecture and hyperparameters. ² Needs to specify clustering algorithms to use.

2.1.1 Statistical Modeling

Statistical modeling approaches aim to judge whether a specific device is operating under an abnormal situation. In [14], a Markov chain based transition model of the devices’ state machine was utilized to judge whether an IEEE 802.11 device is compromised by calculating the probability of its sequential transition of the protocol state machine. In [15], the authors modeled the Electronic Magnetic (EM) harmonics peaks of medical CPS devices as probabilistic distributions to assess whether a specific device is under attack. They assumed that when devices are operated under an abnormal scenario (with the rogue shellcode executing), its EM radiometric signals can deviate from known scenarios. However, statistical modeling requires manual selection of potentially informative features and ranking their importance.

When it is in the Deep Neural Network, the response of a network can be utilized for anomaly detection. To exploit the feature mapping function of neural networks, in [13], the authors formulated it as a semisupervised learning problem. They first trained a CNN model with the last layer as a Softmax function on a collection of known data, and then they removed the Softmax and turned the neural network into a nonlinear feature mapper. Finally, they used cluster analysis on the remapped features. Their results show that such a semisupervised learning method has the potential of detecting...
untrained emitters, but their methods are still far from mature. In [10], the authors provide two methods to deal with unknown inputs: i) Reuse trained convolutional layers to transform inputs to feature vectors and then use Mahalanobis distance to judge the outliers. ii) Reuse the trained convolutional layers and then perform k-means ($k = 2$) clustering to group outliers. In [16] and [17], the data traffic attributes were obtained from flow-level network telemetry to recognize different IoT devices. The authors utilized Principle Component Analysis along with an adaptive one-class clustering algorithm to find the optimal representative components and cluster centers for each device. They show that their method can even detect devices that are hijacked and with abnormal communication behaviors.

Using neural network for abnormality detection is an emerging topic, some state-of-art approaches require more complicated reconstruction approaches such as GAN (Generative Adversarial Network) and Autoencoders.

### 2.1.2 Reconstruction Approaches

Reconstruction approaches aim to discover domain-specific patterns from devices’ normal operation records. In my scope, this means that there is a need to develop a learning agent to ”overfit” the normal schemes of IoT devices by producing low reconstruction errors, at the same time, I hope that the learning agent produces high reconstruction error for unknown scenarios.

This goal is generally achieved using deep autoencoders. Technically, a deep autoencoder is composed of two sequentially connected neural networks, the first network (a.k.a. encoder) maps high dimension input data to a low dimension space. In contrast, the second network (a.k.a. decoder) aims to reconstruct the original inputs from the low
dimension representation. Since a great amount of useful information is removed by the encoder, the decoder needs to reconstruct the lost information according to its domain-specific ‘knowledge.’ As a consequence, once abnormal inputs are given to a well-trained autoencoder, its decoder would not be able to reconstruct such unknown inputs and output a high abnormal score (reconstruction error). In [18], the authors used 155 features of each packet to feed in stacked autoencoders trained on normal operational data and use them for anomaly detection. Similar works are presented in [19, 20], the authors utilized autoencoders to detect abnormal activities by analyzing the data packets of wireless devices, once abnormal activities are detected (high reconstruction error), another classifier was used to classify intrusions. In [21], the authors have shown that compared with other anomaly detection methods (one-class SVM [22], Isolation Forest [23] and Local Outlier Factor [24]), deep autoencoder yields the best result in terms of reliability and accuracy.

### 2.1.3 Predictive Approaches

To use temporal information in devices’ operation records, prediction approaches model the operational data of wireless devices as a multidimension time series, in which device-oriented prediction models are trained using sequential records in normal schemes. It is assumed that, when devices are hijacked for rogue activities, it will not behave as predicted, and prediction errors will occur as alarms.

In [25], the authors employed a CNN based predictor to analyze the abnormal behaviors of users’ cellular network usage records. They show that predictors trained without abnormal data are sensitive to anomalies. Similar work is presented in [26], and the authors used an autoregression model to capture the normal varying trend of devices’
traffic volumes. However, modeling a single variable can not be sufficient in dealing with complicated scenarios. Recent studies combine deep Autoencoders with Long-Short-Term Memory (LSTM) to first derive an abstract representation of complex scenarios and make predictions. In [27] and [28], Deep Predictive Coding Neural Network [29] was used to predict consecutive frames of time-frequency video streams of wireless devices. They methods can even specify the class of attacks using the spatial distribution of error pixels in each frame.

A similar approach using Deep Learning is presented in [30]. The authors used TCP data traffics for each device to train an LSTM-enabled autoencoder to map inputs into a representative feature space. They then used a clustering algorithm to divide the training samples into their natural clusters. Finally, they used probabilistic modeling to associate new data with known clusters for device identification. Unfortunately, their experiments showed that unsupervised behavior identification may not work once there are devices of an identical model.

The aforementioned methods may not be generalized to nonlinear features. In [6], the authors first used a Generative Adversarial Network (GAN) to generate highly realistic fake signals. Then they exploited the discriminator network to distinguish whether an input is from an abnormal source. Similar methods are developed using autoencoders. Autoencoders first encode input vectors into a sparse latent representative space and then use trained domain-specific decoders to reconstruct the original inputs. When an unseen novel input is fed into an autoencoder, the reconstruction is deemed to fail and yields a high error score. However, both GAN and autoencoders introduce new black boxes into processing chains and need special adaptation for real-time safety-critical CPS.
Abnormality detection would partially solve the problem of knowing the unknown, however, many approaches are computationally expensive and can not be applied in safety-critical CPS.

2.2 Incremental learning

In practical scenarios, DNNs would be continuously evolving to adapt to operational variations. For example, a deep learning-enabled wireless device identifier has to learn the characteristics of new devices during its life cycle. These functionalities are defined as lifelong learning in neurophysiological science. Generally, there are two ways to achieve this goal: Transfer Learning (TL) and Incremental Learning (IL). In Transfer Learning, neural networks are pretrained in the lab and then finetune for deployment using practical data [31]. In incremental learning, neural networks are trained incrementally as new data come in progressively [32]. Compared with transfer learning, incremental learning does not allow neural networks to forget what they have learned in the early stages. The phenomenon in which a neural network forgets what it has previously learned after training on new data is named catastrophic forgetting. Therefore, transfer learning is useful when deploying new systems, and incremental learning is useful in regular software updates, as depicted in Figure 2.1.
There are several strategies to implement incremental learning for deep neural networks:

- **Knowledge replay:** The most intuitive solution for incremental learning is to replay data from old tasks while training neural networks for new tasks. However, such a solution requires longer training time and larger memory consumption. In addition, one can not judge how many old samples are enough to catch sufficient variation. Therefore, some studies employ a data generator network to replay data from old tasks. For instance, in [33], Generative Adversarial Network (GAN) based scholar networks were proposed to generate fake samples and mix them with the current task. In this way, the deep neural network could be trained on various data without using substantial memories.

- **Regularization:** Initially, regularization is to prevent models from overfitting by limiting the magnitude of parameters [34]. In incremental learning, the effect of regularization is to prevent the weights (parameters) of connections within neurons from changing dramatically. In this way, the knowledge (parameters) learned from the old tasks will be less likely to vanish when an existing network is trained on new tasks. There are two types of regularization strategies: global regularization and local regularization. Global regularization penalizes a network’s parameters from rapid change, but it can eliminate the network from adapting to new tasks. In local regularization strategies, such as Elastic Weight Consolidation (EWC) [35], the algorithms identify important connections and protect them from changing dramatically. Therefore, the neural network can use noncritical connections to adapt to new tasks.

- **Dynamic network expansion:** Network expansion strategies lock the weights of existing connections and supplement additional structures to learn new tasks.
Chapter 2. Literature Review

For instance, Dynamic Expanding Network (DEN) [36] algorithm first trains an existing network on a new dataset with regularization. The algorithm compares the weights of each neuron to identify task-relevant units. Finally, critical neurons are duplicated to allow network capacity expansion.

Incremental learning algorithms, as well as anomaly detection are critical building blocks for safety-critical CPS. Anomaly detection enables neural networks to know what they do not know, while incremental learning significantly reduces the amount of time for readapting to varying situations. A brief comparison of incremental learning strategies is given in Table 2.2.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Highlights</th>
<th>Major drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge replay w/o GAN-based data generator</td>
<td>Replaying old training data when training for new tasks</td>
<td>Longer training time and higher memory consumption.</td>
</tr>
<tr>
<td>Regularization</td>
<td>Preventing weights from changing dramatically.</td>
<td>Regularization also prevents learning new knowledge.</td>
</tr>
<tr>
<td>Dynamic expandable network (DEN)</td>
<td>Expanding network structure to allow learning new capacity.</td>
<td>Longer training time with expanding memory consumption.</td>
</tr>
<tr>
<td>Elastic Weight Consolidation (EWC)</td>
<td>Preventing critical weights from changing dramatically.</td>
<td>Existing method suffers from numerical stability.</td>
</tr>
</tbody>
</table>

2.3 Quickest Event Detection

Real-time event detection is a critical function in safety-critical CPS. I briefly categorize the approaches for event detection in Figure 2.2. From the perspective of input data, I categorize them into single-shot and sequential detection paradigms. In single-shot detection, event detections are performed per observation (data batch), and the passed data will not be retained for future use. In contrast, the sequential detection paradigm allows accumulating information from past observations. Detecting unknown events is also categorized as anomaly detection. In this dissertation, I focus on real-time sequential
detection of events, and especially on how to integrate quickest detection theory with deep learning to provide a performance-assured solution to latency-constrained CPS.

From the perspective of a stochastic process, a system in different states can be described by distributions with measurable statistical properties. And therefore, transitions within states cause the change of those properties. My target is to detect the changes of statistical properties of a stochastic system and time series with the lowest expected latency. Quickest detection aims to detect the change as quickly as possible, subject to false alarm constraints. The process is in essence an optimization problem formalized by Pollak [37]:

\[
\begin{align*}
\min_{\tau} & \quad \sup_{v \geq 1} E_v[\tau - v | \tau \geq v] \\
\text{subj. to} & \quad E_\infty[\tau] \geq \beta
\end{align*}
\]  

where \( \tau \) is the moment that a change happens at \( v \) is detected. \( E_v[\tau - v | \tau \geq v] \) denotes the expectation of detection latency. \( E_\infty[\tau] \) denotes the mean time between false alarms.
Some research employs another form by Lorden [38]:

\[
\min_{\tau} \sup_{v \geq 1} \sup_{\text{ess sup}} E_v[\tau - v | \tau \geq v] \tag{2.3}
\]

subject to \( E_{\infty}[\tau] \geq \beta \) \tag{2.4}

Compared with Pollak’s formation in (2.1), Lorden’s formation minimizes the worst-case detection delay. The approaches for quickest event detection can be categorized into two branches: a) detecting events with known postchange distributions. b) detecting events with unknown postchange distributions. Generally, detecting known events is faster than detecting unknown events. The scenarios for known event detection methods can be summarized in Figure 2.3. A general workflow with known event distribution is as follows:

**Step 1:** Choose two statistical models for pre-change \((D_0)\) and the changed \((D_1)\) observations.

**Step 2:** Select a metric \(L(D_1|D_0)\) to compute the likelihood of current observation belonging to a changed distribution against the unchanged. Here, the hypothesis is that \(L(D_1|D_0)\) has a mean value of zero before happening of a detectable event.
Step 3: Use a sequential cumulative or multiplicative process to accumulate the deviation of $L(D_1|D_0)$. With CUSUM algorithm we have:

$$S_n = \max(S_{n-1}, 1)L(D_1|D_0), \quad S_0 = 0$$  \hspace{1cm} (2.5)

With Shiryae-Roberts procedure, we have:

$$R_n = (R_{n-1} + 1)L(D_1|D_0), \quad R_0 = 0$$  \hspace{1cm} (2.6)

Given the prior knowledge that the change moment follows a weighted geometry distribution with parameter $p$ and a weighting $\pi$ at time zero, in Bayesian change detection, we have:

$$\Pi_n = \frac{\varphi_n}{1 + \varphi_n}$$  \hspace{1cm} (2.7)

$$\varphi_n = \frac{1}{1 - p}L(D_1|D_0)(\varphi_{n-1} + p), \quad \varphi_0 = \frac{\pi}{1 - \pi}$$  \hspace{1cm} (2.8)

Step 4: Set a threshold $\varepsilon$ according to the the constraints of false alarm, such as mean time between false alarms of false alarm rate.

Please note that Bayesian change point detection algorithms provide a chance to integrate prior knowledge of abruption moments.

In some scenarios, a postchange distribution may not be known in advance and make it difficult to calculate the likelihood ratio $L(D_1|D_0)$. The quickest detection algorithms on the unknown postchange events distribution are summarized in Figure 2.4. Please note that nonparametric strategies usually have higher latency and can not be computed recursively. There are generally two major nonparametric strategies to deal with
uncertain postchange distributions:

- **Earliest warning:** A range of possible parameters for an upcoming change point are defined and multiple detectors are executed simultaneously. The change is assumed to be detected by the detector that gives a warning first.

- **Numerical approximation:** The essential idea is to measure the statistical deviations within samples. Quantile-Quantile difference and sequential ranking based approaches are used to measure the deviation of samples in batch and then insert the measured deviations into the classical sequential cumulative or multiplicative processes.

Figure 2.4: Quickest detection with the unknown post-change distribution.

Quickest detection provides a performance-assured solution to detect change points (related to events) in sequential data. However, the selection of statistic metrics still depends on trial-and-error. And therefore, quickest event detection may not be a mature method to deal with complicated scenarios. It is of great significance to integrate quickest detection theory with deep neural networks to form a quickest event detection framework with automated statistical feature selection.
Chapter 3

Machine Learning for Quickest Event Detection: A Novel Framework

In this chapter, the Machine Learning for Quickest Detection (MLQD) framework will be introduced. I aim to give a big picture of the related technologies and findings in this dissertation. This framework can be generalized to other conventional ML applications.

![Theoretic framework of machine learning for quickest detection](image)

**Figure 3.1:** Theoretic framework of machine learning for quickest detection
A theoretic framework of machine learning for quickest detection is given in Figure 3.1 and the corresponding research and findings in this dissertation are given in Figure 3.2. In this framework is highly dependent on the characteristics of the zero-bias neural network. In essence, the zero-bias neural network provides a ladder for us to explore the topological characteristics of the latent space of neural networks. The zero-bias neural network also provides better performance on abnormality detection. Therefore, I can directly build the single-shot known event detector and the single-shot abnormal event detector. However, single-shot detectors can make mistakes, thereby causing miss detection or false alarms. One way to eliminate false alarms is to use a sliding window to accumulate warning signals and assure the real event. However, configuring the length of the sliding window will be another difficult task. In this dissertation, I use the Bernoulli Generalized Likelihood Ratio test algorithm to process the warning signals. The target here is to detect unknown events under certain false positive constraints with the minimum latency. Different from others, I do not simply feed the results from single-shot detectors into sequential event detectors (usually constructed by CUSUM algorithm) as in other nonparametric approaches. I explicitly explore the latent space of neural networks and propose a new way and make it possible to use parametric approaches for sequential event detection.

As the learning components in the CPS are to be dynamically evolve to adapt to operational variations, I also invented a channel separation enabled incremental learning strategy for time-efficient learning of neural networks. Different from existing approaches, I explicitly explore the root cause of catastrophic forgetting from the perspective of latent space and have provided some new insights on how to design more reliable incremental learning algorithms.

I validated the proposed framework using real ADS-B signals. I use two application
cases to demonstrate the effect of my proposed framework: a) Incremental learning enabled reliable non-cryptographic device identification. b) Quickest Identity Spoofing Detection for ADS-B system.

3.1 Zero-bias Neural Network

Zero-bias neural network is the most important building block of MLQD. Specifically, I invented the zero-bias dense layer to replace the final decision layer (usually the last dense layer before the softmax function) of regular neural networks. In regular scenarios, the decisions are made through biased and weighted cosine similarity matching. In the
zero-bias dense layer, I remove bias neurons and equalize the weights. And therefore, I
make it a fair decision process using cosine similarity as in Equation (4.8). Noted that
cosine similarity can be visualized using the unit hypersphere as in Figure 6.1 or using
Voronoi diagram in Figure 6.2. In this section, I only highlight several key features of
zero-bias neural networks that support the remaining findings.

1. **Regional Association Characteristic:** I visualize the decision boundaries of
zero-bias neural network and derive Remark 6.1, 6.3, and 6.5. These remarks
provide a novel solution to turn a well trained zero-bias neural network into a
performance assured binary abnormality detector.

2. **Optimal separation of class representation vectors in the latent space:**
I discover and prove that the class representation vectors (class fingerprints) in
the latent space will distribute following a minimal interference manner as in
Theorem 5.1, 5.6, and 5.5. Using these theorems I analyze the root cause of
catastrophic forgetting and invented my Channel Separation Incremental Learn-
ing (CSIL) Strategy. CSIL enables zero-bias neural network to evolve and adapt
to operational variations without needing retrain from scratch.

### 3.2 Deep Learning Enabled Quickest Event Detection

In this dissertation, I first use initial training to derive a zero-bias neural network for
event detection. In which, the event detector is capable of detecting some unknown
events and incrementally learning new events. The event detection scene of MLQD can
be categorized into two parts:
1. **Single-shot event detection:** in this case, the specific type of event has already been learned by the zero bias neural network event detector reliably. In this case, I don’t need to apply sequential event detection strategy.

2. **Sequential event detection:** When there’s interference as in the signal intelligence and spectrum sensing applications, sequential event detection strategy is needed. The merit of sequential event detection is that it gradually accumulate evidence and eliminate false alarms. One typical way is to use CUSUM algorithm to process the output of the event detector, CUSUM algorithm has been proved to have the minimum detection delay (quickest detection). However, before this dissertation, there’s still no way to connect CUSUM algorithm with neural network within a performance assured way.

### 3.3 Time Efficient Incremental Learning

Neural networks in CPS or IoT (Internet of Things) are supposed to evolve to adapt to operational variations during their life-cycle. However, learning new concepts without forgetting the existing ones is never easy. I use zero-bias neural network to explore the mathematical essence of catastrophic forgetting and derived my own incremental learning strategy. This part of work provides insights into the latent space characteristics of deep neural networks.
Chapter 4

Zero-Bias Deep Learning with Application in Wireless Device Identification

In this chapter, I propose an enhanced deep learning framework for accurate and interpretable identification of IoT devices with mathematically assured performance. I propose a zero-bias dense layer for Deep Neural Networks to jointly verify known devices and identify unknown ones. The effectiveness of the proposed framework in handling massive signal recognition and improving the performance of traditional neural networks has been demonstrated.

My research offers not only a new paradigm in DNNs, thus useful in promoting trustworthy IoT, but also a deep learning framework for intrusion detection. In addition, the introduction of zero-bias layer in deep neural networks represents an advance in deep learning, thus leveraging deep learning to enable the move from IoT to real-time control.
4.1 Zero-bias Dense Layer for Deep Learning

This section will provide a thorough analysis of the mechanism of dense layers in regular neural networks and introduce the zero-bias layer to replace the decision layer (the last dense layer) of DNNs. The framework is capable of reporting unseen patterns to increase dependability and explainability.

4.1.1 Formalization of Zero-bias Neural Network

A typical architecture of the deep learning model is given in Figure 4.1. Generally, the input data are processed with several convolution layers to extract latent information, then several dense layers are used to remove the redundancy and derive the final result. The last dense layer (the decision layer) is considered to have the potential for various applications, especially for transfer learning [39–41].

![Figure 4.1: Typical architecture of deep neural network](image)

Start from the last dense layer, suppose that I have $m$-dimension input vectors with batch size $k$, the layer needs to convert the input into $k$ $n$-dimension output vectors. A linear calculation on the input is conducted as:

$$Y_1 = W_1 X + b_1$$  \hspace{1cm} (4.1)

where $X$ denotes the input data, which is a $m$ by $k$ matrix. $W_1$ denotes the weights, which is an $n$ by $m$ matrix. Finally, $b_1$ denotes an $n$-dimension bias vector. I break the
regular dense layer into two consecutive parts, depicted in Figure 4.2, a regular dense layer denoted by $L_1$ and a dense layer without bias $L_2$, respectively. Then, the function applied to the input data becomes:

$$Y_2 = W_2Y_1 = W_2W_1X + W_2b_1 \quad (4.2)$$

where $W_1$ and $b_1$ belong to $L_1$ while $W_2$ belongs to $L_2$. Note that Equations (4.2) and (4.1) are performing equivalent transforms to $X$ and should not degrade the network performance. Therefore, I get my first theorem:

**Theorem 4.1** (Equivalent replacement of the last dense layer). *I replace the last dense layer of a neural network with a consecutive structure consisting of a regular dense layer ($L_1$) and a dense layer without bias ($L_2$). And this modification is equivalent to the original neural network.*

However, from a more systematic view, $L_1$ performs a dimension transform (reduction in most cases) of features from prior convolution layers. Moreover, in $L_2$, I can rewrite the matrix calculation into vectors:

$$Y_2 = [w_{21}, w_{22}, \ldots, w_{2n}]^T [y_{11}, y_{12}, \ldots, y_{1k}] \quad (4.3)$$
where $w_{21}, \ldots, w_{2n}$ are row vectors corresponding to $n$ output classes; $y_{11}, \ldots, y_{1k}$ are $k$ column vectors corresponding to the batch size. Each column vector denotes latent features, a.k.a. feature vector, of an input sample. For specific feature vector $y_{1k}$, the output of $L_2$ is:

$$Y_2[y_{1k}] = [w_{21} \cdot y_{1k}, w_{22} \cdot y_{1k}, \ldots, w_{2n} \cdot y_{1k}]$$  \hspace{1cm} (4.4)$$

where $y_{1k}$ and $Y_2[y_{1k}]$ are feature vector and output vector, respectively. And the Softmax layer picks up the position of the largest element in $Y_2[y_{1k}]$ as classification output. The process in equation (4.4) can be rewritten using the Cosine Similarity:

$$w_{2n} \cdot y_{1k} = ||w_{2n}|| \cdot ||y_{1k}|| \cdot \cos(w_{2n}, y_{1k})$$  \hspace{1cm} (4.5)$$

If $w_{21}, \ldots, w_{2n}$ is considered as fingerprints of classes 1 to $n$, the zero-bias dense layer $L_2$ actually calculates a linearly scaled version of cosine similarity among the inputs against the fingerprints of target classes.

Moreover, I can safely generalize this discovery to explain the behavior of dense layers in DNN:

**Remark 4.2 (Property of dense layer).** If the outputs of a decision layer (always the last dense layer) represent the degrees of confidence of associating specific labels to an input, then each confidence degree is a linear projection of cosine similarity to the corresponding class-related fingerprint. Its value is jointly controlled by the magnitude of the fingerprint, the input vector’s cosine similarity to the fingerprint, and the bias of the corresponding class.
Although the magnitude of an input feature vector $||y_{1k}||$ seems to take effect as in Equation (4.5), but in the consecutive Softmax layer, the magnitude $||y_{1k}||$ only contributes to a common base number as in Equation (4.6):

$$\text{Softmax}(L_2) = \frac{\exp(||y_{1k}|| \cdot ||w_{2n}|| \cdot \cos(w_{2n} \cdot y_{1k}))}{\sum_n \exp(||y_{1k}|| \cdot ||w_{2n}|| \cdot \cos(w_{2n} \cdot y_{1k}))}$$  (4.6)

where $\exp(||y_{1k}||)$ is the common base number and controls the steepness of the monotonic mapping curve.

**Remark 4.3.** Neural networks’ global preference (partiality): A neural network’s global preference or partiality to specific classes is encoded in its last dense layer before Softmax, and the partiality is jointly controlled by the magnitude of class-related fingerprint vectors and the bias value of the corresponding class.

In my proposed paradigm of modified dense layer, I can derive more specific corollaries:

**Corollary 4.4.** Fingerprints’ magnitude: If the variance of the magnitude of fingerprint vectors is small, the layer $L_2$ has less preference or partiality towards specific classes.

Currently, there are two approaches to remove the unwanted effects of fingerprint vectors’ magnitudes:

1. I can use regularization to eliminate the variance of fingerprints. For example, when it’s under L2 regularization, the weights of fingerprint vectors in a decision layer will be with small magnitude, and thus have smaller variation on decision weights;
2. I can replace Equation (4.3) by Equation (4.7):

\[ Y_2 = \left[ \frac{w_{21}}{\sqrt{w_{21}^2}}, ..., \frac{w_{2n}}{\sqrt{w_{2n}^2}} \right]^T [y_{11}, ..., y_{1k}] \]  

(4.7)

Moreover, I can eliminate the side effects of feature vectors’ magnitude at the same time:

\[ Y_2 = \lambda \left[ \frac{w_{21}}{\sqrt{w_{21}^2}}, ..., \frac{w_{2n}}{\sqrt{w_{2n}^2}} \right]^T \left[ \frac{y_{11}}{\sqrt{y_{11}^2}}, ..., \frac{y_{1k}}{\sqrt{y_{1k}^2}} \right] \]  

(4.8)

where \( \lambda \) is a trainable scalar to provide the freedom of controlling the steepness of the mapping curve. In this way, the magnitudes of fingerprints or even feature vectors will not take effect anymore. Please note that \( Y_2 \) is differentiable in these two scenarios. Equation (4.8) in this context is the cosine similarity comparison.

The transformation from a regular dense to a zero-bias dense layer is summarized as in Figure 4.3. Noted in the first three transformation stages, the zero-bias deep neural network is mathematically equivalent to regular neural networks, while the forth stage totally turns \( L_1 \) into a dimensionality reduction layer. I actually replace the last dense layer of a neural network with a consecutive structure consisting of a regular dense layer \( (L_1) \) and a single-kernel cosine similarity comparing layer \( (L_2) \).

I notice that some researches directly employ Equation (4.8) as cosine similarity [42, 43] in deep learning, I are different as: a) I provide a mathematically equivalent transform by using another regular fully connected layer \( L_1 \). b) my experiments show that directly applying cosine similarity without \( L_1 \) dramatically increases the difficulty of training.
4.1.2 Zero-bias Neural Network for Abnormality Detection: A Threshold Based Approach

If the side effects of fingerprints’ magnitude can be neglected (using the cosine similarity comparison), I have Corollaries 4.5 and 4.6:

**Corollary 4.5** (Fingerprints’ mutual distances). *Fingerprints in the zero bias dense layer (L₂) should have sufficiently small mutual cosine similarities.*

**Corollary 4.6** (Fingerprints’ relation to features). *Each fingerprint acts as an angular representative of the feature vectors of the corresponding classes but does not necessarily contain the complete information of the specific input category.*

A simplified example of Corollary 4.6 is given in Figure 4.4. Suppose that there are three classes (A, B, and X) for a deep neural network to distinguish. The fingerprint vector of each class only captures a rough angular representation (direction).

Using the properties of the zero-bias layer enabled neural network, I can utilize the similarity values for anomaly detection. I assume that the maximum similarity scores of known and abnormal (unknown) inputs are from two different distributions as in Figure 4.5. According to (4.8), I have:
Remark 4.7. The maximum similarities of known inputs would distribute close to positive $\lambda$ while the maximum confidences of abnormal inputs are smaller and would be distributed relatively closer to negative $\lambda$.

Remark 4.8. If the maximum confidence value of an input vector is lower than a certain threshold, this input is considered novel or is severely contaminated.

An example is in Figure 4.14 of Section 4.2.5.2, in which the zero-bias layer enables deep neural networks to have perfectly projected known and abnormal inputs to two different distributions. If the two distributions are distinctively separated and have small overlapping areas, I can even use thresholds.

### 4.1.3 Error Tendency Analysis for Zero-bias Neural Network

A very charming benefit of zero-bias neural networks is that it provides an elegant interface to evaluate how well different classes are mutually distinguishable.
Different from regular neural networks, the final decisions of zero-bias neural networks purely depend on similarity matching. Without the interference of weights and biases, I can easily interpret and visualize the mutual relationship of fingerprints (classes) in the latent space. I can construct a Fingerprint Distance (FD) matrix as:

\[
FD = \begin{bmatrix}
cos(w_1, w_1) & \ldots & cos(w_1, w_n) \\
\vdots & \ddots & \vdots \\
cos(w_n, w_1) & \ldots & cos(w_n, w_n)
\end{bmatrix}
\] (4.9)

This matrix can directly reflect how well different classes are separated in the latent space. I replace the last dense layer with zero-bias dense layer (containing both $L_1$ and $L_2$) in the MNIST example [44] and plot the FD matrices when the training accuracy reaches 60.2% and 95.8%, respectively. As in Figure 4.6, I use the color to denote the cosine similarity. Apparently, the fingerprints of classes in the latent space are more distantly separated when the neural network is trained after more iterations (with higher accuracy). Observing the matrix, digit 4 and digit 6 are more likely to get confused since their fingerprints are with higher cosine similar.

![Figure 4.6: Fingerprint distance matrix of Minst example](image)

Such a fingerprint distance matrix is visually analogue to a regular confusion matrix, but they are mathematically different in the following ways: a) a confusion matrix depends highly on the distribution and characteristics of the data while my method is
data independent. b) the method can be used to analyze the latent space characteristics of neural networks.

4.1.4 Incremental Learning for Zero-Bias Neural Network: A Direct Approach

I propose an incremental learning scheme for both zero-bias DNN and conventional DNN classifiers:

**Remark 4.9** (Incremental learning using the zero-bias dense layer). To enable a neural network to recognize a novel class, one only needs to insert its representative fingerprint in the last dense layer and fine-tune the old fingerprints’ directions when necessary.

Remark 4.9 indicates that:

- For a specific novel class, as long as the previous layers have extracted sufficient distinctive features, I do not need to retrain the previous layers.
- For new classes, I need to insert new fingerprints and then adjust the old fingerprints when necessary.

To adjust an old fingerprint, I need to identify which parameter (or dimension) is critical to the classification accuracy. According to the Elastic Weight Consolidation (EWC) [35], the Fisher Information Matrix is used to model the importance of parameters as:

$$F_\Omega = \left[ \frac{\partial \log(P(X_{CV}|\Omega))}{\partial \Omega} \right] \left[ \frac{\partial \log(P(X_{CV}|\Omega))}{\partial \Omega} \right]^T$$

$$P(X_{CV}|\Omega) \approx Y_{\text{Softmax}}(X_{CV}|\Omega)$$  \hspace{1cm} (4.10)
where \( \overline{\text{Y}_{\text{Softmax}}}(X_{\text{CV}}|\Omega) \) denotes the averaged outputs of Softmax layer on validation set \( X_{\text{CV}} \) given parameter set \( \Omega \), it approximates the posterior probability \( P(X_{\text{CV}}|\Omega) \). \( F_\Omega \) denotes the Fisher information matrix of the current task. In my experiment, I further apply an exponential function to the Fisher Information to increase the numerical stability as:

\[
F_\Omega := \exp(F_\Omega) \tag{4.11}
\]

Intuitively, the importance of a parameter is equivalent to the square of its gradient with respect to the logarithm of Softmax output function, also known as the log-likelihood function.

Knowing the importance of existing parameters, I can define an integral loss function for incremental learning as:

\[
F_1(\Omega) = \frac{\lambda_1}{2} \sum_i [F_{\Omega^*} \cdot (\Omega - \Omega^*^2)]
\]

\[
L(\Omega) = (L_2(\Omega) + F_1(\Omega)) \cdot G_m \tag{4.12}
\]

where \( F_1(\Omega) \) denotes the Fisher Loss with respect to old tasks (a.k.a., task-1). \( \Omega^* \) denotes the loss function and model parameters on task-1. \( L_2(\Omega) \) and \( \Omega \) denote the raw loss function on Task-2 and the new model parameters. \( \lambda_1 \) denotes the importance of task-1. Intuitively, this integral loss function additionally penalizes the change of critical parameters. \( G_m \) is a mask matrix, in which the value of each element can only be zero or one. These elements are one-to-one associated with the parameters of a neural network to control which parameter is locked or unlocked.
Given a neural network trained on Task-1 ($DNN_1$), incremental learning on Task-2 is performed as follows:

**Step 1:** Store all learnable parameters of $DNN_1$ as $\Omega^*$ and calculate their importance matrix $F_{\Omega^*}$.

**Step 2:** Generate the initial fingerprint of each new class by averaging their feature vectors.

**Step 3:** Concatenate initial fingerprints into the last dense layer or zero-bias dense layer.

**Step 4:** Lock the weights of previous layers and calculate the importance of parameters of old fingerprints. The importance of newly concatenated fingerprints is set to zeros; thus, I could allow them to learn freely.

**Step 5:** Use loss function as in Equation (4.12) and a training set of Task-2 to perform network training.

Notably, I do not need to retain old training data to learn a new task, and such a benefit is critical for DNN models in practical scenarios.

### 4.2 Zero-bias Deep Neural Network for Wireless Device Identification

#### 4.2.1 Introduction

The Internet of Things (IoT) is characterized by the interconnection and interaction of smart objects (objects or devices with embedded sensors, onboard data processing capabilities, and means of communication) to provide applications and services that
would otherwise not be possible [45]. The convergence of sensors, actuators, information, and communication technologies in IoT produces massive amounts of data that need to be sifted through to facilitate reasonably accurate decision-making and control [46]. Big data analytics has the potential to enable the move from IoT to real-time control [47]. However, due to the open nature of IoT, IoT is subject to cybersecurity threats [48, 49]. One typical cybersecurity threat is identity spoofing attacks, where an adversary passively collects information and then mimics the identity of legitimate devices to send fake information or conduct other malicious activities. Such attacks can be extremely dangerous when appearing in critical infrastructures [50].

Conventional approaches to prevent identity spoofing attacks employ cryptographic algorithms to verify that a trusted source generates a message. However, the cryptographic approaches depend on the secrecy of encryption keys and encounter challenges from the open and heterogeneous ecosystems of IoT. For example, a number of commercially successful IoT systems, which do not operate with cryptographic keys, require a large investment to become cryptographically secure [51]. Therefore, there is a need for non-cryptographic solutions to verify the identity of IoT devices, thus ensuring trustworthy IoT.

Non-cryptographic IoT device identification is inspired by signal identification technology in speech and acoustic signal processing [3]. The assumption is that each signal source modulates its unique features into propagated signals. Comparably, in non-cryptographic IoT device identification, I assume that each wireless transmitter randomly picks up certain types of imperfection (a.k.a radiometric fingerprint) during their manufacture [52] and could be reflected in the demodulated signals. Existing works on non-cryptographic device identification can be classified into two categories: specific feature recognition and deep learning. Specific feature-based approaches focus
on deriving distinctive features (a.k.a., transmitter fingerprints) from received signals [53, 54] to recognize known devices. Deep learning based approaches do not require knowing the devices’ radiometric characteristics and show even higher accuracy [55, 56]. However, the challenge of applying deep learning approaches for IoT device identification lies in two aspects: unseen device recognition and model interpretability. The first challenge requires deep neural networks to report unseen devices rather than erroneously associating them with known ones. The second challenge requires that the behaviors of neural networks to be interpretable.

In this application, I use zero-bias deep neural network for accurate and interpretable identification of IoT devices with mathematically assured performance. This application offers not only a solution to accurate identification of IoT devices, thus useful in promoting trustworthy IoT, but also a deep learning framework for intrusion detection. In addition, the introduction of zero-bias layer in deep neural networks represents an advance in deep learning, thus leveraging deep learning to enable the move from IoT to real-time control.

4.2.2 Related Work

Non-cryptographic device identification is emerging as a solution to the physical layer security of IoT. Corresponding methods can be classified into two categories: specific feature based and deep learning based.
4.2.2.1 Specific feature-based approaches

The specific feature-based approaches require human efforts to discover distinctive features for device identification. The methods rely on the fact that there are various manufacturing imperfectnesses in wireless devices’ RF frontends. These imperfectnesses do not degrade the communication quality but can be exploited to identify each transmitter uniquely. Those features are named Physical Unclonable Features (PUF) [57, 58]).

There are two categories of PUFs: error patterns and transient patterns.

In the error pattern approach, it is assumed that the statistical properties of the received symbols’ noise could uniquely profile wireless devices. In [59], the authors show that the phase error of Phase Lock Loop in transmitters can provide promising results even with low Signal-to-Noise Ratio (SNR). In [60], the authors use the difference between received signals and theoretical templates to construct error vectors. Error vectors’ statistics and time-frequency features are combined as fingerprints for transmitter identification. In [61], the authors employ the differential constellation trace figure (DCTF) to capture the time-varying modulation error of Zigbee devices. They then develop their low-overhead classifier to identify 54 Zigbee devices.

In the transient pattern approach, it is assumed that a malicious entity can not forge the transient response characteristic of wireless transmitters [62]. Transient patterns are commonly seen at the beginning and end of wireless packet transmission. In [63], nonlinear in-band distortion and spectral regrowth of the signals are utilized to distinguish the masquerade emitter. In [64], the authors employ the transient energy spectrum of transmitters’ turn-on amplitude envelopes to identify, and they show that frequency-domain features outperform time-domain features.
Feature-based approaches require efforts to manually extract features or high-order statistics for different scenarios. Therefore, more effortless and versatile methods are required.

### 4.2.2.2 Deep Learning for Wireless Transmitter Identification

Deep Neural Networks (DNNs) are frequently used as a general-purpose BlackBox for pattern recognition. Naturally, they are applied to perform device-specific identification.

A typical DNN enabled wireless device identification system employs convolutional layers to extract latent features. Convolutional layers apply filters (a.k.a., kernels) to obtain helpful information automatically. Such benefit reduces the hardship of manual feature discovery. In [65], the authors provide a novel method that performs signal denoising and emitter identification simultaneously using an autoencoder and a Convolution Neural Network (CNN). Their solution shows promising results even with low SNR. Similar work in [66] employs stacked denoising auto-encoder and showed similar results. DNNs perform well even on raw signals. In [67], the authors provide an optimized Deep Convolutional Neural Network to classify SDR-based emitters in 802.11AC channels, they show that, even by using raw signals without feature engineering, CNN surpasses the best performance of conventional statistical learning methods. In [68], neural networks were trained on raw IQ samples using the open dataset\(^1\) from CorteXlab. Their works also show similar results. Compared with the specific feature-based approach, deep neural networks dramatically reduce the requirement of domain knowledge and the quality of fingerprints.

In general, DNNs are becoming a promising building block in non-cryptographic wireless device identification. DNNs encounter a challenge in terms of anomaly detection,

\(^1\)[https://wiki.cortexlab.fr/doku.php?id=tx-id]
which requires that deep learning identification systems not only perform well on trained objects but also can report unknown objects that it would make a wrong decision. Furthermore, for dependable machine learning in practical scenarios, I need to understand how a neural network associates an input with a corresponding label. These two aspects are rarely covered in signal identification, thus motivating my research.

### 4.2.3 Problem definition

In this research, I focus on deriving a protocol-agnostic solution to identify IoT devices from physical layer signals. The reason is that the signal features directly correspond to hardware components and reveal the identities of IoT devices.

I define that an IoT device $i$ transmits specific message with corresponding baseband signal $m_i(t)$. $m_i(t)$ is modulated into:

$$M_i(t) = C_i[m_i(t)] \quad (4.13)$$

where $C_i(x)$ denotes the frequency band processing chain. At receiver $j$, the received signal becomes:

$$R_{ij}(t) = S_{ij}[M_i(t)] \quad (4.14)$$

where $S_{ij}$ denotes the effect of wireless channel between $i$ and $j$. This function can incorporate the effect of attenuation or additive noise. The demodulated signal is:

$$\hat{m}_i(t) = S_j^{-1}[C_j^{-1}[R_{ij}(t)]] \quad (4.15)$$

$$= S_j^{-1}[C_j^{-1}[S_{ij}[C_{i}[m_i(t)]]]]$$
where \( C_j^{-1}(x) \) and \( S_j^{-1}(x) \) are \( j \)'s estimated reverse function of \( C_i(x) \) and \( S_{ij} \), respectively. The estimation can hardly be idealistic. Therefore, at the receiver side, \( j \), the effect of such discrepancies are reflected in \( \hat{m}_j(t) \) as:

\[
\hat{m}_j(t) = r_i(t) + \delta_j(t)
\]  

(4.16)

where \( r_i(t) \) is directly correlated with \( m_i(t) \) while the residual, \( \delta_j(t) \), is utilized to recognize a wireless device. As long as \( \delta_j(t) \) is uncorrelated with messages \( m_i(t) \), the recognition algorithm is protocol-agnostic. Apparently, this is a classification problem, to avoid the hardship of feature engineering, I use DNN and convert IoT device recognition problem into 3 subproblems:

1. Given message-related baseband signals from various wireless transmitters, how to extract message-independent components to develop a classifier using DNNs?

2. How to enable my classifier to properly respond to unseen signals?

3. How can I evaluate the distinguishability between different devices?

4.2.4 Methodology

In this section, I first present the feature extraction methods and then introduce the zero-bias deep learning framework for accurate and interpretable identification of IoT devices.

4.2.4.1 Baseband demodulation

In this research, I use an independent Software-Defined Radio (SDR) receiver, denoted as \( j' \), to collect baseband signals from wireless transmitters, denoted as \( \hat{m}_{ij'}(t) \). Given
an input signal \( x \), the quadrature demodulation function is defined as:

\[
C_{j'}^{-1}(x) = I(t) + i \cdot Q(t) \\
= LPF[x \cdot \cos(\omega_c t + \phi_0) + i \cdot x \cdot \sin(\omega_c t + \phi_0)]
\]

where \( I(t) \) and \( Q(t) \) are In-Phase and Quadrature components, respectively. \( \omega_c \) and \( \phi_0 \) are the center frequency and the phase offset of the receiver \( (j') \), respectively. \( i \) denotes the imaginary part of the complex function. With Phase Lock Loop (PLL), \( \omega_c \) and \( \phi_0 \) are supposed to be sufficiently close to RF characteristics of device \( i \). \( LPF \) denotes a low-pass filter. Therefore, at \( j' \), the demodulated baseband is:

\[
\hat{m}_{j'}(t) = C_{j'}^{-1}[R_{ij'}(t)]
\]

\( \hat{m}_{j'}(t) \) is complex-valued, and its instantaneous amplitude, phase and frequency are

\[||\hat{m}_{j'}(t)|| = \sqrt{I^2(t) + Q^2(t)}, \quad \angle \hat{m}_{j'}(t) = \tan^{-1}\left(\frac{Q(t)}{I(t)}\right)\]

and

\[\hat{\Omega}_{j'}(t) = \frac{d\angle \hat{m}_{j'}(t)}{dt}\], respectively.

Please note that discrepancies exist between \( \hat{m}_j(t) \) and \( \hat{m}_{j'}(t) \). Even if the wireless channel effects at the receiver \( j \) and \( j' \) are different, I assume that an SDR receiver could still capture the effect of each wireless device’s frequency band processing chain, \( C_i(x) \), to recognize them.

### 4.2.4.2 Feature extraction

For protocol-agnostic device recognition, I need to remove message-correlated part \( r_i(t) \) from \( \hat{m}_{j'}(t) \). In this way, I ensure that my device recognition mechanism is protocol-agnostic. In addition, I only use the first 1,024 samples of \( \hat{m}_{j'}(t) \).
**Pesudo Noise Extraction:** Suppose I have derived the numerical sequence of instantaneous metrics (amplitude, phase, or frequency), and the corresponding procedures are as follows:

**Step 1:** I separate the sequence (denoted as $s_{j'}(n)$) into several non-overlap segments, with each segment’s duration less than one symbol duration.

**Step 2:** For each segment, I perform *k-medoids* algorithm on signals instantaneous phase or amplitudes with $k = 2$. In essence, I use a clustering algorithm to associate numeric values to their closest medoids (representative values). Notably, I could only expect one or two possible choices of amplitudes or phases.

**Step 3:** In each segment, I generate the pesudo-noise as:

$$n_{j'}(n) = s_{j'}(n) - m_k[s_{j'}(n)] \quad (4.19)$$

where $m_k$ denotes the medoid of $s_{j'}(n)$, I subtract rationale signals from the demodulated baseband signals directly.

A brief comparison of related signals is in Figure 4.7. Medoids could be regarded as a less noisy version of the demodulated baseband signals $\hat{m}_{j'}(t)$.

![Figure 4.7: Noise extraction on typical signals.](image-url)
Chapter 4. Zero-bias Deep Learning

Figure 4.8: Correlation coefficients of pseudo noise/

The distribution of correlation coefficients (derived from 10,000 samples) of pseudo-noise against the corresponding baseband signals is depicted in Figure 4.8. The pseudo-noise signals are weakly correlated with the original messages.

**Frequency domain features:** I subtract the Fourier Transforms of both complex-valued baseband signals $\hat{m}_j(t)$ and the reconstructed rationale baseband signals to extract the message uncorrelated residual components in the frequency domain, formulated as:

$$\delta_j(\omega) = FFT[\hat{m}_j(t)] - FFT[r_j(t)]$$

(4.20)

where $r_j(t)$ is the reconstructed rational baseband signal. Please note that $\hat{m}_j(t)$ is complex-valued (QPSK) while $r_j(t)$ can be real-valued (2FSK, 2PSK, and etc.). I convert the residual components into a magnitude sequence ($||\delta_j(\omega)||$), namely, Mag.-Freq. residuals, and a phase sequence ($\angle \delta_j(\omega)$), namely Phase-Freq. residuals, respectively.

### 4.2.5 Validations

#### 4.2.5.1 Validation I: Trainability of Zero-bias Neural Network

Automatic Dependent Surveillance-Broadcast (ADS-B), which functions with satellite rather than radar technology to more accurately observe and track air traffic, is an
application of safety-critical CPS in aviation. In this subsection, I evaluate the zero-bias deep neural network on real ADS-B baseband signals.

In my data collection pipeline, depicted in Figure 4.9, I used a modified gr-adsb library to decode ADS-B messages and store raw baseband digital signals. I collected the ADS-B signal from 150 aircraft at Daytona Beach international airport (ICAO: DAB) for 24 hours (Jan 4th, 2020) with a Software-Defined Radio receiver (USRP B210). The receiver is configured with a sample rate of 8 MHz. During this period, more than 30,000 ADS-B messages are collected with geographical coordinates depicted in Figure 4.10.

![Collection of ADS-B signals for model validation.](image)

**Figure 4.9:** Collection of ADS-B signals for model validation.

![Geographic distribution of aircraft transponders.](image)

**Figure 4.10:** Geographic distribution of aircraft transponders.

I use a DNN model depicted in Figure 4.11 to verify the identity of transmitters with given raw baseband signals. The neural network employs a similar architecture as ResNet[69]. I use convolution layers with skip connections to extract useful features and use a modified dense layer followed by a softmax layer for final classification.
Figure 4.11: Deep neural network architecture for wireless transmitter identification.

A comparison of the training process of neural networks with the proposed zero-bias layer and a regular dense layer on the same dataset is given in Figure 4.12. They reach identical performance in terms of validation accuracy. However, the zero-bias layer requires more training iterations, and its rising rate of accuracy is lower at the beginning. To evaluate the deep learning model in terms of training data efficiency, I manually limit the number of samples of each transmitter in the training set and use this specially "reduced" training set to train the zero-bias DNN model. As depicted in Figure 4.13, the model converges after 800 iterations (40 epochs) and the experiment shows that I only need 200 samples to recognize each transmitter.

4.2.5.2 Validation II: Unknown device identification

A wireless device identification system needs to respond to unseen data. In conventional neural networks, the classification layer only associates labels with the largest activation value from the last hidden layer. However, such behavior would result in wrong answers given novel signals from unknown devices. In my research, I explore two ways
to identify whether a signal is from an unseen novel device. Suppose that my neural network converts input data $X$ to feature vectors $F_X$ through intermediate layers, I divide the discussion into two scenarios: Similarity thresholds and One-class Support Vector Machine.

I randomly pick ADS-B signals from 30 aircraft to train the neural network and use signals from the remaining 120 aircraft as unseen novel devices’ signals. This section will compare the performance of my zero-bias layer, regular dense layer, and one-class SVM.

Comparisons with threshold based methods: I employ a zero-bias layer (Equation (4.8)) for final output. The probability distributions and decision thresholds are depicted in Figure 4.14a and 4.14b, respectively. Figure 4.14a demonstrates that the similarities of unknown signals are higher than unknown signals in most cases. Figure 4.14b shows that I can select an optimum separation threshold to maximize the decision margin of
the anomaly detection algorithm. In this research, I used the median value of similarity score values of known signals minus its standard deviation as a decision threshold.

I train identical neural networks but with the zero-bias layer replaced by a regular dense layer under the same criteria. However, the anomaly detection performances are much worse, as depicted in Figure 4.15a and 4.15b, the similarity score distributions of the regular dense layer with known and unknown data are severely overlapped. The decision margin in this scenario is much smaller.

**Comparisons with One-class Support Vector Machine:** I use the feature vectors of the training set to train a one-class SVM model and use the feature vectors of the validation set and unseen signals to test the performance of one-class SVM. I collect the prediction scores on both known signals and unknown signals with the results presented in Figure 4.16a and 4.16b, respectively. The result indicates that the prediction score
distribution of one-class SVM significantly differs from the previous cases in the following aspects:

1. Prediction scores of signals from unseen devices only occupy a narrow area.

2. Scores of known devices occupy a much wider area (larger variance), which may cause difficulty for choosing the right threshold.

I can conclude that the performance of the zero-bias layer enabled neural network in anomaly detection is comparable with one-class SVM. However, in my experiment, the one-class SVM model ultimately stores more than 5,000 support vectors, while the zero-bias layer only stores low dimension fingerprints of known aircraft transponders. Therefore, I believe my solution is more adaptable for real-time machine learning.

4.2.5.3 Validation III: Incremental learning

To evaluate the zero-bias neural network for IL, I first filter out transponders with very few records. According to the appearance probability of RF transponders (Figure 4.17), I use 400 appearances as a threshold to separate my data set into two parts, namely task-1 and task-2 respectively. I first train the neural network on task-1 and use incremental learning mechanisms to let my network recognize wireless transmitters in task-2 without forgetting task-1.

![Figure 4.17: Transponders’ appearance frequency](image-url)
Comparison of incremental learning strategies: I compare other incremental learning strategies with my solution (locking prior layers while allowing EWC in the last layer). The descriptions of all these approaches are given in Table 4.1. I aim to compare the effect of EWC as well as other network knowledge protection methods. Please note that during the incremental learning, L2 regularization factors for the regular neural network and zero-bias neural networks are all set to 0 and 0.025, respectively. I will discuss the effect of L2 regularization in Section 4.2.5.3. The results are given in Table 4.1. I highlight several observations:

Figure 4.18. I highlight several observations:

1. In Global EWC, catastrophic forgetting is not prevented. Besides, the zero-bias layer retains far less knowledge from previous tasks.
2. Only training new fingerprints and locking all old weights in the network can help retain knowledge from previous tasks. This phenomenon indicates that the prior layers have already extracted useful features for the final classification. Moreover, the performance of the zero-using bias layer indicates that it would enable prior neural network layers to discover better features. Please note that this scenario also prevents the fine-tuning of existing fingerprints even if they are in sub-optimal directions.

3. Only protecting old fingerprints does not seem to be helpful. The new task will destroy all useful feature extractors in prior layers.

4. Applying EWC only in the last layer provides the most promising results. Notably, the neural networks with the zero-bias layer still outperform regular neural networks. This fact explains that EWC tries to protect old fingerprints from changing erroneously (forgetting) and enables fine-tuning.

Effects of regularization: Regularization can prevent neural networks' parameters from varying dramatically during training. I perform incremental learning using L2 regularization factors: 0, 0.025, and 0.05, respectively. The results in Figure 4.19a indicates that without L2 regularization, the neural networks’ performance on the old tasks can degrade slightly. With a larger L2 factor, as in Figure 4.19b and 4.19c, the performance on old tasks is better retained. However, larger L2 factors also worsen the performance of incremental learning. As shown in Figure 4.19c, although the performance on the old tasks is maintained, the accuracy of new tasks is bounded to 80%.

Similar experiments are conducted using regular dense layer with the results given in Figures 4.19d, 4.19e and 4.19f, respectively. In my test case, when the L2 regularization factor is zero, the neural network with regular dense layer gradually forgets what it
learns in task-1. Increasing the L2 regularization factor stops the forgetting trend, but the performance of my proposed zero-bias layer is much better. It allows the network to gain comparable performance on both new (task-2) and old (task-1) tasks.

I encounter the numerical instability issue when developing the incremental learning algorithm. By integrating my zero-bias layer with EWC along with locking prior layers, I temporally mitigate the problem of catastrophic forgetting.

**Numerical stability:** I compare the numerical stability of Fisher Loss during incremental learning. The results in Figure 4.20 demonstrate that without applying the exponential function as in Equation (4.11), the Fisher Loss is numerically unstable and
gradually vanishes to zero (depicted by dashed lines). When Fisher Loss becomes zero, the incremental learning algorithm can no longer penalize the neural network for forgetting the old tasks. In contrast, if the exponential function is applied, the Fisher Loss never vanishes to zero and thus prevents catastrophic forgetting. As the incremental learning moves on, the Fisher Loss gradually converges to a nonzero constant value, and the results indicate that the zero-bias layer in my solution has a smoother converging characteristic than the regular dense layer.

Figure 4.19: Effects of L2 regularization.
Chapter 4. Zero-bias Deep Learning

4.2.5.4 Validation IV: Error Tendency Analysis

I analyze the error tendency of the trained zero-bias neural network as in section 4.1.3. The cosine similarity matrix of devices’ fingerprints in the latent space is given in Figure 4.21. Figure 4.21 also presents the cosine similarity matrix of device fingerprints after incremental learning (Elastic Weight Consolidation). Interestingly, the newly learned devices’ fingerprints have much higher cosine similarity with the existing ones and can explain the performance degradation after incremental learning.
4.3 Further Discussion

4.3.1 Multi-kernel Classification Using Zero-bias Neural Network

As discussed in Section 4.1.1, I prove that in neural networks decisions will be made through a single kernel nearest neighbour matching process. Naturally, one may wonder is it possible to turn the single kernel matching to a multiple kernel matching. To explore the question, I first design a deep learning enabled dual-kernel nearest neighbour matching scheme as depicted in Figure 4.22. As depicted, I will have two decision layers to perform the cosine similarity matching.

![Figure 4.22: The architecture of the dual-kernel classification neural network.](image)

To prevent the two layers to learn identical fingerprints, I modify the loss function of the model as:

\[
L(\Theta, Fp_1^1 \cdots Fp_n^1, Fp_1^2 \cdots Fp_n^2) = L_C + L_K
\]

\[
L_K(Fp_1^1 \cdots Fp_n^1, Fp_1^2 \cdots Fp_n^2) = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{Fp_i^1}{||Fp_i^1||} \cdot \frac{Fp_i^2}{||Fp_i^2||} \right]
\]

(4.21)

where \(L_C\) denotes the classification loss in terms of cross entropy loss. \(L_K\) is the averaged kernel distance and denotes the averaged distances between corresponding fingerprints within the two zero-bias layers. \(Fp_1^1 \cdots Fp_n^1\) and \(Fp_1^2 \cdots Fp_n^2\) denote the class fingerprints in the first and second zero-bias dense layers, respectively. In essence, for each class, I want to encourage the neural network to find two sets of class representation vectors (fingerprints) that are distantly distributed in the latent space.
I test the network in Figure 4.22 as using the wireless device identification dataset. The training process of the neural classifier is given in Figure 4.23. As depicted, I can divide the training process of this network into two stages:

![Learning to classify and Tuning the kernels’ distances](image-url)

**Figure 4.23:** Training of the dual-kernel neural classifier.

1. **Learning to classify:** In this stage, the optimization target is to minimize the classification loss $L_C$ while average kernel distance $L_K$ grows.

2. **Optimizing kernel distance:** the target in this stage is to reduce the $L_K$ while maintaining classification accuracy $L_C$.

Although high classification accuracy is achieved, I discover that such a multi-kernel strategy brings severe side effects on the networks’ performance on abnormality detection. One reason is that the neural network actually performs unnecessary vector decomposition to reduce the term $L_K$ and the loss function, as depicted in Figure 4.24. The unnecessarily decomposed fingerprints increases the spatial occupation of each class thereby causing the degradation of abnormality detection performance. I believe that multi-kernel classification is not a good idea to be applied in neural classifiers.
4.4 Concluding Remark

This chapter discusses the formalization and definition of the zero-bias neural network, I then apply the zero-bias neural network to enhance the wireless transmitter identification problem. I make the decision process of deep neural network transparent and enable neural networks to properly respond to unknown inputs. In the meantime, I provide a novel method to analyze the potential error risk within my zero-bias deep neural networks. Furtherly, my zero-bias neural network performs better than regular neural networks in terms of state-of-art incremental learning strategies. However, in specific which human-like behaviors such as biases and preferabilities are expected, zero-bias neural network may not be a good solution since zero-bias neural network would abandon making decisions according to its general experience, which is encoded in bias and weights in the decision layer.
Chapter 5

Orthogonal Memory Organization
for Time Efficient Incremental Learning in Wireless Device Identification

In this chapter, I explore the topological properties of class representation vectors (class fingerprints) in the latent space of deep neural networks, I discover that the main cause of catastrophic forgetting is due to the nonoptimal distribution of feature vectors and their representatives (fingerprints) in the latent space. I provide a new metric, the Degree of Conflict (DoC), to quantitatively analyze the topological maturity of DNN models. Based on the discoveries, I designed an enhanced IL scheme, the Channel Separation Enabled Incremental Learning (CSIL). I automatically introduce separations in representative spaces between different tasks (learning stages). The effectiveness of the proposed framework in massive signal recognition and enhanced incremental learning
has been demonstrated. My research is useful for the future development of IL for DNNs. To my best knowledge, this is the first study that jointly explores DNN and IL in Signal Intelligence Applications. Right before the publication of this work, I realized that my algorithm actually has solid evidence from the most recent advancement of neural science [70]. I share some similar findings as in [70], but from a totally different perspective and a nonbiological road map. In addition, I provide the mathematical proof and are delighted to find an elegant connection between biological and artificial intelligence.

5.1 Topological Characteristics of Class Representation Vectors

I discover that the last dense layer of a DNN classifier performs the nearest neighbor matching with biases and preferabilities using cosine similarity, I also show that a DNN classifier’s accuracy will not be impaired if I replace its last dense layer with a zero-bias dense layer [71], in which the decision biases and preferabilities are eliminated. I can denote the mechanism of zero-bias dense layer as (also in Figure 4.2):

\[
Y_1(X) = W_0X + b
\]

\[
Y_2(X) = cosDistance(Y_1, W_1)
\]

(5.1)

where \(X\) is the output of the prior convolution layers, a.k.a., feature vectors. \(X\) is an \(N_0 \times q\) matrix, where \(N_0\) denotes the number of features while \(q\) denotes the batch size. \(W_0\) is an \(N_1 \times N_0\) matrix where \(N_1\) denotes the dimension of fingerprints in the latent space. \(W_1\) is a matrix to store fingerprints of different classes, namely the similarity matching layer and it is a \(C \times N_1\) matrix in which \(C\) denotes the number of
classes, I set $N_1 = 2C$ in this chapter. Please note that in $W_1$, each row represents a fingerprint of corresponding class whilst in $Y_1$ each column represents a feature vector in the latent space. Intuitively, the last dense layer is spitted into two layers, $L_1$ for feature embedding and $L_2$ for similarity matching. Considering the batch form, the cosine similarity matching is denoted as:

$$cosDistance(Y_1, W_1) = RU(W_1) \times CU(Y_1) \quad (5.2)$$

where $RU(\cdot)$ and $CU(\cdot)$ denote deriving column-wise and row-wise direction vectors (vectors’ magnitudes are normalized to one) of their inputs. My prior results [71, 72] prove that the zero-bias dense layer can work seamlessly with backpropagation mechanisms and trained using regular loss functions (e.g., binary crossentropy, etc.). Please note that even if $L_2$ can be replaced by a regular dense layer, it can also be viewed as a similarity matching layer, but the matching results are weighted and biased [71].

The cosine similarity in Equation 5.1 represents the similarity matching of fingerprints and feature vectors on an $N_1$-D unit hyperspherical surface.

### 5.1.1 Optimal Separation of Class Representation Vectors

Intuitively, if the devices’ fingerprints are distantly separated in the latent space, I will have less chance to confuse them. To quantify the separation, the sum of the mutual cosine distances of all devices’ fingerprints in a classification model can be defined as:

$$TD(f_1, \cdots, f_C) = \sum_{i=2, j=1, j<i}^{C} CosineDistance(f_i, f_j)$$

$$= \sum_{i=2, j=1, j<i}^{C} x_i^{(1)} x_j^{(1)} + x_i^{(2)} x_j^{(2)} + \cdots + x_i^{(N_1)} x_j^{(N_1)} z \quad (5.3)$$
where \( f_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(N_1)}) \) and \( f_j = (x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(N_1)}) \) are class fingerprint vectors. Actually, \( TD(f_1, \ldots, f_C) \) is the lower triangular summation of the class fingerprints’ cosine similarity matrix as in Figure 4.21.

Suppose I have \( C \) classes with \( N_1 \)-D fingerprint vectors. Noted that the fingerprints have been scaled into unit vectors. Therefore, if I need to find the optimal value of \( TD(\cdot) \), I need to incorporate the constraints:

\[
\forall i, \ g(f_i) = \sum_{d=1}^{N_1} (x_i^{(d)})^2 - 1 = 0 \quad (5.4)
\]

Equation 5.3 has now become a constrained optimization problem. I solve this constrained optimization problem with the Lagrange Multiplier as:

\[
L(f_1, \ldots, f_C, \lambda_1, \ldots, \lambda_C)
= TD(f_1, \ldots, f_C) - \sum_{i=1}^{C} \lambda_i g(f_i) \quad (5.5)
\]

And I need to solve:

\[
\nabla \left( x_1^{(1)} \ldots x_1^{(N_1)} \ldots x_C^{1} \ldots x_C^{N_1} \lambda_1 \ldots \lambda_i \right) L(f_1 \ldots f_C, \lambda_1 \ldots \lambda_C) = 0 \quad (5.6)
\]

which results in a linear system of equations. For each \( k \)th \((k = 1 \ldots N_1)\) dimension of fingerprint vectors \( x_1^{(k)}, \ldots, x_C^{(k)} \), I have:

\[
\frac{\partial L}{x_1^{(k)}} = -2\lambda_1 x_1^{(k)} + \sum_{i=1,i\neq 1}^{C} x_1^{(i)} = 0
\]

\[
\vdots \quad \cdots \quad \vdots
\]

\[
\frac{\partial L}{x_C^{(k)}} = -2\lambda_C x_C^{(k)} + \sum_{i=1,i\neq C}^{C} x_C^{(i)} = 0
\]

\[
(5.7)
\]
This is a homogeneous system of equations, and it is unlikely that it only has a trivial solution (zeros). Hence, $\lambda_1 = \lambda_2 = \cdots = \lambda_C = -0.5$ and Equation 5.7 can be converted into one equation:

$$\sum_{i=1}^{C} x_i^{(k)} = 0 \quad (5.8)$$

I square Equation 5.8 and expand it. According to Multinomial Theorem [73] I have:

$$\sum_{i=1}^{C} (x_i^{(k)})^2 + 2 \sum_{n=1, m<n}^{C} x_n^{(k)} x_m^{(k)} = 0 \quad (5.9)$$

Given that $k = 1 \cdots N_1$, I have $N_1$ Equations with an identical form of Equation 5.9.

By summing them up, I have:

$$\sum_{k=1}^{N_1} \sum_{i=1}^{C} (x_i^{(k)})^2 + 2 \sum_{k=1}^{N_1} \sum_{n=1, m<n}^{C} x_n^{(k)} x_m^{(k)} = 0 \quad (5.10)$$

On the left of Equation 5.10, the first part is the sum of the magnitude of fingerprint vectors. And its value is $C$. The second part is exactly two times $TD(f_1, \cdots, f_C)$ in Equation 5.3. Therefore, I have:

**Theorem 5.1. Converging Point Theorem:** The sum of the mutual cosine distances of classes’ fingerprints of the zero-bias DNN at a converging point is a predictable constant:

$$TD(f_1, \cdots, f_C) = -\frac{C}{2} \quad (5.11)$$

When such a value is reached, the separation of fingerprints are maximized (with minimized interference) in the latent space, indicating the lowest degree of conflict. I will use the term *Degree of Conflict (DoC)* to describe the characteristic of the zero-bias
DNN. Noted that the range of DoC is from $-\frac{C}{2}$ to $\frac{C(C-1)}{2}$. The maximum value is reached when all fingerprints collide into one single vector.

To demonstrate the Remark 5.1, I use a simple DNN [44] with two configurations. In the first configuration, a regular dense layer is applied for the final classification. And in the second configuration, the last dense layer is modified to perform the cosine similarity matching as in Equation 5.1. The two models are trained on the hand-written digit dataset (MNIST). And the change of DoC and accuracy during training are depicted in Figure 5.1. In Figure 5.1a, the degree of conflict of zero-bias DNN model converges to the predicted optimal constant $-\frac{10}{2} = -5$. However, in the regular DNN model, the metric stops at a nonoptimal point, $-3$. Notably, higher accuracy could sometimes reflect a lower DoC between fingerprints. Figure 5.1b also reveals that the zero-bias DNN model is less sensitive to the variation of DoC.
5.1.2 Conflict of Class Representation Vectors and Orthogonal Memory Representation

5.1.2.1 Catastrophic Forgetting and Degree of Conflicts

With the cosine similarity matching mechanism, one may assume that incremental learning can be performed by simply inserting new fingerprints. However, I discovered that such an intuitive method could cause significant performance degradation. An important factor to cause the performance degradation is the conflict of fingerprints.

To exemplify this phenomenon, I use two DNN models with an architecture specified in Figure 4.11, I modify their last dense layers as in Figure 4.2, I use cosine similarity matching in $L_2$ for the first DNN model and use regular dense layer for $L_2$ for the second one, and therefore, the second DNN is a regular DNN. The two models are trained and tested using a two-stage incremental learning scheme: a) in the first learning stage, the two models are first trained on a wireless signal identification dataset [74] to classify 18 most frequently seen wireless devices. b) Before the second learning stage, I insert the hypothetic fingerprints (generated by averaging feature vectors) of the remaining 16 new devices into their similarity matching layers and freeze all prior layers and fingerprints of learned devices. c) In the second stage, the IL stage, I finetune the newly inserted fingerprints. After the two-stage learning, the cosine similarity matrix of fingerprints in the two models before and after incremental learning is compared in Figure 5.2.

The results in Figure 5.2 indicate a typical conflict scheme. On the one hand, some fingerprints of the newly learned classes (devices) are less distantly separated as they have higher cosine similarities. On the other hand, some new devices’ fingerprints have high cosine similarities with old devices’ fingerprints. These two factors jointly cause
conflict and confusion. A more detail comparison is provided in Table 5.1. The two models degrees of maturity after IL are far from the expected optimal value. And DoC of the new fingerprints are also far from optimal.

**Table 5.1: A comparison of the DoC of DNN models before and after incremental learning.**

<table>
<thead>
<tr>
<th>DNN models</th>
<th>DoC (Acc.) initial training</th>
<th>DoC (Acc.) a.f.t. finetuning</th>
<th>DoC of new fingerprints</th>
<th>Acc. on new / old task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>-8.083 (90.54)</td>
<td>-1.16 (65.2)</td>
<td>9.05</td>
<td>75.5 / 54.2</td>
</tr>
<tr>
<td>Zero-bias</td>
<td>-8.96 (92.85)</td>
<td>-4.3 (84.2)</td>
<td>4.03</td>
<td>76.2 / 91.3</td>
</tr>
<tr>
<td>Optimal value</td>
<td>-9</td>
<td>-18 (92.2)</td>
<td>-8</td>
<td>92.2 / 93.1</td>
</tr>
</tbody>
</table>

**Remark 5.2.** The conflict between class representation vectors is an important cause of catastrophic forgetting.

Interestingly, the zero-bias DNN outperforms the regular DNN considering less catastrophic forgetting.

### 5.1.2.2 Orthogonal Memory Organization and Proof of Optimality

Theorem 5.1 will provide us a tool to explore catastrophic forgetting. Suppose that I have $N_1$ classes at the initial stage and $m$ new classes to learn afterwards. I define that
the averaged cosine distance between $N_1$ fingerprints is $D_0$, according to Remark 5.1, after initial training I have $\frac{N_1(N_1 - 1)}{2}$ pairs of cosine similarity values, according to Theorem 5.1 I have:

$$\frac{N_1(N_1 - 1)}{2}D_0 = -\frac{N_1}{2} \text{ and } D_0 = -\frac{1}{N_1 - 1}$$ (5.12)

when there are $N_1 + m$ classes, $D_0$ has to become:

$$D_1 = -\frac{1}{N_1 + m - 1}$$ (5.13)

It means that if the classes’ fingerprints are to be distantly and uniformly separated, the averaged angles of all old fingerprints need to be reduced while learning new classes. This requirement can not be satisfied if the old fingerprints are locked or prevented from changing.

When the prior layers are locked, the distribution of feature vectors in the latent space is fixed, simply reducing the separation of fingerprints in old classes will increase the degree of conflict and cause performance degradation, as depicted in Figure 5.1b.

**Remark 5.3.** In regular neural networks, without readjustment of old fingerprints or proper separation between old and new fingerprints, the conflict between fingerprints can not be resolved. Under such criteria, the resulting DNN’s performance will not be comparable to training with all data from scratch.

Dynamically adjusting the separation angles of all existing classes while inserting new classes is computationally difficult. However, there are several solutions I can apply to theoretically avoid introducing new interference or conflicts during increment learning:
Lemma 5.4. The dual orthogonality criteria: If the newly inserted classes’ fingerprints are mutually orthogonal in the task-specific subspace (Intra-Class Orthogonality), and they are orthogonal to the existing ones (Inter-Class Orthogonality), the incremental learning process will not introduce any unwanted conflict.

Proof. Suppose that I have \(C + m\) classes in which \(m\) classes are introduced by increment learning. Equation (5.3) can be expressed as (also in Figure 5.3):

\[
TD(f_1 \cdots f_C, f_{C+1} \cdots f_m) = \sum_{i=2, j=1, j<i}^{C+m} \text{CosineDistance}(f_i, f_j)
= TD(f_1, \cdots, f_C) + TD(f_{C+1}, \cdots, f_m) + SD(f_1 \cdots f_C, f_{C+1} \cdots f_m) \quad (5.14)
\]

Figure 5.3: Lower triangular proportion of fingerprint cosine similarity matrix after incremental learning.

where \(TD(f_1, \cdots, f_C)\) is the DoC of existing classes’ fingerprints. As represented by the green triangle in Figure 5.3.

\[
TD(f_1, \cdots, f_C) = \sum_{i=2, j=1, j<i}^{C} \text{CosineDistance}(f_i, f_j) \quad (5.15)
\]
$TD(f_{C+1}, \cdots, f_m)$ is the DoC of new classes’ fingerprints, as represented by the blue triangle in Figure 5.3.

$$TD(f_{C+1}, \cdots, f_m) = \sum_{k=C+1, l=C+2, k<l}^{m} CosineDistance(f_k, f_l) \quad (5.16)$$

The last term $SD(f_1 \cdots f_C, f_{C+1} \cdots f_m)$ is the sum of the pairwise cosine similarity between new and old classes’ fingerprints. As represented by the red rectangle in Figure 5.3.

$$SD(f_1 \cdots f_C, f_{C+1} \cdots f_m) = \sum_{r=C+1, s=1}^{r=m, s=C} CosineDistance(f_r, f_s) \quad (5.17)$$

If the new classes’ fingerprints are mutually orthogonal in their task-specific subspace, satisfying the Intra-Class Orthogonality condition. And in the meantime, if they are orthogonal to the existing ones, satisfying the Inter-Class Orthogonality condition, then I have:

$$TD(f_{C+1}, \cdots, f_m) = 0 \quad (5.18)$$

$$SD(f_1 \cdots f_C, f_{C+1} \cdots f_m) = 0 \quad (5.19)$$

That is:

$$TD(f_1 \cdots f_C, f_{C+1} \cdots f_m) = TD(f_1 \cdots f_C) \quad (5.20)$$

Therefore, the unwanted conflict will not be introduced.

Keeping the newly inserted fingerprints mutually orthogonal is not easy. However,
Equation (5.18) indicates that I only need to let the DoC of new classes to approximate zero. Since $TD(f_{C+1}, \cdots, f_m)$ is an aggregated value, therefore, I do not need to constrain the class fingerprints of new classes to be strictly orthogonal. Therefore, I can rewrite Lemma 5.4 into:

**Theorem 5.5.** Approximated Orthogonal Incremental Learning Theorem: If the newly inserted classes’ fingerprints are with a DoC value close to zero and they are orthogonal to the existing ones, the incremental learning process will not introduce any unwanted conflict.

I provide an intuitive way to keep the newly inserted classes’ fingerprints at any learning stage to approximate zero:

**Theorem 5.6.** Orthogonality Approximation Theorem: To minimize the mutual interference, the separation of class representation vectors (class fingerprints) approximates a mutually orthogonal relationship within the latent space as long as the number of classes is increased.

**Proof.** Let $\overline{D_1}$ be the averaged cosine similarity between fingerprints, according to Theorem 5.1 and Equation (5.12), if I have $N_1$ classes, at the converging point where conflict between fingerprints is minimized, I will have $\frac{N_1(N_1 - 1)}{2} \overline{D_1} = -\frac{N_1}{2}$ and $\overline{D_0} = -\frac{1}{N_1 - 1}$.

Therefore, $\overline{D_1}$ approximates zero when $N_1$ becomes larger. It means the averaged separation angle of fingerprints approximates 90 degrees (orthogonal).

Theorem 5.6 also explains the phenomenon that the larger the number of classes in each stage becomes, the less catastrophic forgetting there will be during incremental learning as in [75–78].
Theorem 5.1, 5.5 and 5.6 provide the following guidelines for designing incremental learning algorithms:

1. Learn as much classes as possible to satisfy the intra-class orthogonality condition.

2. Class fingerprints at each stage should be separated into their own subspace. And the subspaces of different learning stages should be mutually orthogonal so as to satisfy the inter-class orthogonality condition.

These guidelines motivate the new incremental learning strategy, The Channel Separation Enabled Incremental Learning (CSIL) Strategy.

5.2 Channel Separation Incremental Learning (CSIL)

To resolve the conflict of fingerprints, I proposed the Channel Separation Enabled Incremental Learning (CSIL), an integral approach incorporating dimension expansion and channel separation as depicted in Figure 5.4. Intuitively, the merits of this approach are: a) the class fingerprints at different stages automatically use their task-specific proportions (channels) of parameters in the feature embedding layer. b) the directions of fingerprints from different stages are constrained to be orthogonally separated.

At the initial stage, namely stage-0, I train a zero-bias DNN as normal. When at the \( k \)th learning stage, stage-\( k \). I first expand the feature embedding layer’s weight matrix as:

\[
W_0^{(k)} = \begin{bmatrix} W_0^{(k-1)} & \cdots & w_0^{(k)} \end{bmatrix}
\]  

(5.21)
where $W_0^{(k-1)}$ is the weight matrix of the feature embedding layer in $(k-1)$th stage and $w_0$ is the expanded proportion for the $k$th task. I then expand the similarity matching layer of the network as:

$$W_1^{(k)} = \begin{bmatrix} W_1^{(k-1)} & 0 \\ 0 & w_1 \end{bmatrix}$$  \hspace{1cm} (5.22)$$

where $W_1^{(k-1)}$ is the weight matrix of the similarity matching layer at in the $(k-1)$th stage and $w_1$ is the fingerprints for the $k$th task. The manually inserted zeros on the one hand keep the fingerprints in different stages orthogonal (depicted in Figure 5.5), on the other hand, they enable the feature embedding layer to learn task-specific parameters in different regions (a.k.a. channels). For instance, in Equation 5.22, the newly inserted fingerprints in $w_1^{(k)}$ only make use of embedded features from $w_0^{(k)}$ in Equation 5.21.

I only train the network with data from the $k$th stage, I use Knowledge Distillation (KD [79]) and Elastic Weight Consolidation (EWC [35]) to prevent the model from forgetting. Therefore, the loss function is defined as:

$$L(\Theta_{k-1}, \theta_k, G_m, X_k) = L_{CE} + L_D + L_{EWC}$$ \hspace{1cm} (5.23)
where $\Theta_{k-1}$ denotes the models’ weight at the $(k-1)$th stage. And $\theta_k = \{w_0^{(k)}, w_1^{(k)}\}$ denotes the extended weights for the $k$th stage. $G_m$ is a mask matrix, in which the value of each element can only be zero or one. These elements are one-to-one bound to the parameters of a neural network to control which parameter is locked or unlocked. $X_k$ is the training data of the $k$th stage. $L_{CE}$ is the cross entropy loss of the new task. $L_D$ is the Knowledge Distillation loss:

$$L_D = \| R_{k-1}(X_k) - R_k(X_k) \|$$  \hspace{1cm} (5.24)

where $R_{k-1}(X_k)$ is the response of $(k-1)$th model on $X_k$ and $R_k(X_k)$ is the response of the $k$th model. $F(\cdot)$ denotes the output of the similarity matching layer ($L_2$).

Knowledge Distillation aims to penalize DNNs’ behavior from changing drastically.

$L_{EWC}$ in Equation 5.23 represents the Elastic Weight Consolidation (EWC) loss. In EWC, Fisher Information Loss is used to measure the importance of existing parameters, I define EWC Loss for incremental learning as:

$$L_{EWC}(\Theta_k) = \frac{1}{2} \sum_i |F_{k-1} \cdot (\Theta_k - \Theta_{k-1})|^2$$  \hspace{1cm} (5.25)

where $F_{k-1}$ denotes the Fisher Information (FI) matrix with respect to the $(k-1)$th task. Intuitively, this loss function penalizes the change of critical parameters. The matrix can be estimated as:

$$F_{k-1} = \left[ \frac{\partial \log P(X_{k-1}|\Theta_{k-1})}{\partial \Theta_{k-1}} \right]^2$$

$$P(X_{k-1}|\Theta_{k-1}) \approx \overline{Y_{\text{Softmax}}}(X_{k-1}|\Theta_{k-1})$$  \hspace{1cm} (5.26)
where $\mathbf{Y}_{\text{Softmax}}(\mathbf{X}_{k-1}|\Theta_{k-1})$ denotes the averaged outputs of Softmax layer on validation set $\mathbf{X}_{k-1}$ given parameter set $\Theta$, it approximates the posterior probability $P(\mathbf{X}_{k-1}|\Theta_{k-1})$. $F_{k-1}$ denotes the Fisher information matrix.

To exemplify the concept, in Figure 5.5, I compare the fingerprints’ cosine similarity matrix after regular training and CSIL using the same dataset and scheme specified in Section 5.1.2. In this experiment, the convolution layers, the channels for the old task in the feature embedding layer, and the manually supplemented zeros in the similarity matching layer are locked. As a comparison, the DoC of fingerprints is much less apparent compared to Figure 5.2. A more systematic comparison is provided in Section 5.3.

More details about this experiment are presented in Table 5.2.

**Table 5.2:** A comparison of the degree of conflicts of DNN models with different training strategies.

<table>
<thead>
<tr>
<th>Training strategy</th>
<th>DoC (Acc.)</th>
<th>DoC of new / old fingerprints</th>
<th>Acc. on new / old task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>-16.25 (92.85)</td>
<td>n/a</td>
<td>92.85</td>
</tr>
<tr>
<td>CSIL</td>
<td>-16.15 (88.6)</td>
<td>-8.9 / -7.21</td>
<td>91.2 / 87</td>
</tr>
<tr>
<td>Optimal value</td>
<td>-17</td>
<td>-9 (-8)</td>
<td>100 / 100</td>
</tr>
</tbody>
</table>
5.3 Application of CSIL in Wireless Device Identification

5.3.1 Introduction

A typical way to implement smart decision functionality in IoT is by integrating learning-enabled components through Deep Learning (DL) and Deep Neural Networks (DNNs). One typical application of DNNs in IoT is the passive identification of IoT devices through their wireless signals for Non-cryptographic Device Identification (NDI) and Physical Layer authentication [48, 80, 81]. DL and DNNs are effective in wireless device identification under various scenarios, however, DNN models in these applications need to be continuous evolving to adapt to operational variations as new devices (as new classes) are emerging. Such a continuous evolving scheme is termed as Lifelong or Incremental Learning (IL).

Conventional approaches require periodic retraining to update DNNs. In this paradigm, DNNs are initialized from scratch and trained with all past and present devices’ signals. Even though the best accuracies are guaranteed in these Non-Incremental Learning (Non-IL) schemes, the memory consumption and training time can grow drastically as new devices are added in. Therefore, there is a need for IL with a reasonable balance between accuracy, memory consumption, and training efficiency. In IoT, less or zero memory for historical data are preferred during the continuous evolving [82].

Compared to conventional non-incremental learning (non-IL) schemes, DNN models can only use a very small proportion or even none of the data from the previous stages, a.k.a. old tasks, while they are trained to recognize new devices. The absence of data from old tasks results in Catastrophic Forgetting, a phenomenon of significant degradation of
accuracy after training on new tasks. IL has become an emerging topic in machine learning, however, many of the methods are not adaptable in IoT. For example, some works require storing specifically chosen old data and can consume a large amount of memory [75] gradually. Other works require incrementally training task-related generative models for knowledge replay, but these generative models require notorious efforts [83]. In addition, there are several attempts to either use regularization or knowledge distillation to implement memoryless methods to prevent DNNs from forgetting [82]. Balancing between learning and forgetting is difficult, especially when the internal mechanism of catastrophic forgetting is not yet clear. Besides, there is a lack of theoretic explanation to explore the difference between the key characteristics between IL and regularly trained models.

In this application, I apply the channel separation incremental learning strategy for time-efficient incremental learning in non-cryptographic device identification of IoT. I provide a comprehensive comparison of my CSIL strategy with other IL approaches without storing past data using massive signal recognition systems. To my best knowledge, this is the first study that jointly explores DNN and IL in Signal Intelligence Applications.

5.3.2 Evaluation Dataset

I use real-world ADS-B signals to verify IL methods for wireless device identification. ADS-B signals are transmitted by commercial aircraft to periodically broadcast their enroute information to Air Traffic Control (ATC) Centers in plain text. These signals are easy to receive and decode but are subject to identity spoofing attacks. I configure my SDR receiver (USRP B210) with a sample rate of 8MHz at 1090MHz, and for each piece of intercepted message, I use the first 1024 complex samples. This dataset is
publicly available at [84]. I first decode the ADS-B signals using a modified version of Gr-ADS-B in [74] to extract the payloads, then the aircraft’s identity codes are used as labels for the truncated messages’ signals. I filter out the wireless transponders with less than 500 samples and use the top 100 most frequently seen transponders to construct the dataset. As in Section 4.2.4.2, I extract the pseudonoise supplemented with the frequency domain information, I convert each truncated message signal into a 32 by 32 by 3 tensor. Finally, I got 100 wireless transponders. I use 60% of the dataset for training and the remaining 40% of the dataset for validation.

5.3.3 Performance Evaluation

In this subsection, I compare the CSIL algorithm with other incremental learning algorithms that do not require historical data. The configurations of the selected methods are as follows:

- **Channel Separation Enabled Incremental Learning (CSIL):** I lock the convolution layers and channels in the feature embedding layer which are used by old tasks. I train the new task-specific channels and fingerprints of devices.

- **Learning without Forgetting (LwF):** I lock the convolution layers and the feature embedding layer, I use LwF to train the similarity matching layer.

- **Elastic Weight Consolidation: (EWC):** I lock the convolution layer and feature embedding layer, I train the whole similarity matching layer. The EWC algorithm can adjust old and new fingerprints simultaneously.

- **Finetuning:** I lock the convolution layer, the feature embedding layer, and the old fingerprints, I train the similarity matching layer with new fingerprints.
In these configurations, I set the initial learning rate to be 0.01, momentum to 0.9, and $L^2$ regularization factor to be 0.01. Stochastic Gradient Descent is selected. I divide the data tensors from 100 wireless devices into 5 batches. I first train the selected DNN model with 20 randomly selected devices and then incrementally train the model with other data batches. During incremental training, the batch size is set to 64 and the models are trained for 10 epoches.

I compare their resulting models’ performance on old and incrementally learned new devices as in Figure 5.7. Since no historical data is available during incremental learning, forgetting of old tasks are unavoidable. From Figure 5.7a, the performances of all selected IL algorithms in recognizing new devices are close to the optimal non-IL scheme, in which the proposed CSIL yields the highest accuracy after IL while finetuning with locked old fingerprints shows the worst result. Comparably, in Figure 5.7b, in preventing forgetting, CSIL’s performance is not as good as finetuning with locked weights after learning more than 60 wireless devices (classes). Finetuning with locked weights prevents DNN models from forgetting but with a side effect that prevents the network from learning new devices. The overall performance is given in Figure 5.7c, my proposed algorithm CSIL yields the best performance on both old and new tasks.

A comparison of the metric, the Degree of Conflict (DoC), of all devices’ fingerprints during incremental learning, is given in Figure 5.6. The propose method, CSIL, yields the lowest DoC. Please note that the models’ DoC values are still lower than the optimal values (please refer to Equation 5.11) after incremental learning.
Chapter 5. Orthogonal Memory Organization for Incremental Learning

Figure 5.6: Degree of Conflict among IL algorithms

Figure 5.7: Comparison of incremental learning strategies for wireless device identification
5.3.4 Ablation Analysis

I compare the averaged stage loss of the CSIL considering three factors: a) the Fisher loss. b) The Knowledge Distillation loss. c) The effect of channel separation. The results of ablation analysis are given in Table 5.3. Apparently, the integral method combining channel separation, EWC, and Knowledge Distillation provides the best performance.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Initial Acc.</th>
<th>Acc. with all 100 devices</th>
<th>New acc. at the last stage</th>
<th>Old acc. at the last stage</th>
<th>Forget / stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS + EWC + KD</td>
<td>95.2</td>
<td>83.5</td>
<td>90</td>
<td>73</td>
<td>4.5</td>
</tr>
<tr>
<td>CS + EWC + KD</td>
<td>95.2</td>
<td>75.3</td>
<td>82.4</td>
<td>66.3</td>
<td>5.78</td>
</tr>
<tr>
<td>CS + EWC + KD</td>
<td>95.2</td>
<td>70.5</td>
<td>91</td>
<td>50</td>
<td>9</td>
</tr>
<tr>
<td>CS + EWC + KD</td>
<td>95.2</td>
<td>70.5</td>
<td>91</td>
<td>50.2</td>
<td>9</td>
</tr>
</tbody>
</table>

1 Identification accuracy on the first 20 devices, at this stage the network is trained from scratch.  
2 Overall accuracy (100 devices) after the last stage of incremental learning.  
3 Averaged decrease of accuracy on all trained devices after each incremental learning stage.

Notably, without channel separation, the combination of elastic weight consolidation and knowledge distillation can also prevent the network from forgetting. However, such a combination also prevents the network from learning new tasks. Therefore, elastic weight consolidation and knowledge distillation jointly prevent the network from forgetting old devices when training on new tasks, meanwhile, the channel separation mechanism prevents the conflict of class-specific fingerprints.

A more detailed comparison is presented in Figure 5.8. In Figure 5.8a, if the channel separation mechanism is not available, the DNN model will not perform well in learning new devices (classes), as analyzed in Remark 5.3, the incrementally inserted fingerprints of new devices can conflict with the existing ones, causing the performance degradation. In Figure 5.8b, the integral solution, CSIL, yields the highest accuracy in terms of memorizing old devices. Interestingly, the integral of knowledge distillation and elastic weight consolidation ranks the second place in memorizing old devices while showing
the worst performance for learning new ones. Therefore, the CSIL provides the best balanced performance between learning and forgetting.

### 5.4 Further Discussion: Local Degree of Conflict During Incremental Learning

To provide a more explicit analysis on the catastrophic forgetting, I did some more exploration on the topological characteristic classes’ fingerprints. I analyze the degree
of conflict of fingerprints during different learning stages using the same dataset and configuration as in Section 5.3.2 and Section 5.3.3. This part of discussion aims to verify Theorem 5.5 from a empirical perspective.

**Definition 5.7.** Local Degree of Conflict (DoC): I define that the degree of conflict of a subgroup of classes is the local degree of conflict. A subgroup can consist of classes learned at a specific stage.

Firstly, I compare the variation of the local DoC of fingerprints in the zero-bias neural network with the finetune for incremental learning. The architecture of the network is specified in Figure 4.11 with results in Figure 5.9. I consolidate all old weights and finetune new classes’ fingerprints in the decision layer. As depicted, the test accuracy of the oldest task (Task-1) reduces slightly as the incremental learning moves on. However, the local DoC values of the newly learned tasks (from Task 2 to 5) increase drastically. Indicating that it has become increasingly difficult for new classes’ fingerprints to be optimized as specified in Converging Point Theorem (Theorem 5.1).

![Figure 5.9: Local Degree of Conflict of finetune with weight consolidation](image)

Secondly, I compare the local DoC of Elastic Weight Consolidation (EWC) as depicted in Figure 5.10. As depicted, the DoC of the oldest task (Task-1) grows significantly during incremental learning while the test accuracy decreases as learning moves on. This phenomenon also applies to other tasks and learning stages, indicating that the
topological characteristic of classes in old tasks will be destroyed, gradually causing interference in the latent space.

![Image](image1.png)

(a) Test accuracy at different stages

(b) Local DoC at different stages

**Figure 5.10:** Local Degree of Conflict of regular EWC with weight consolidation

Thirdly, I design a Conflict-Aware Elastic Weight Consolidation (CEWC) algorithm, specifically, the algorithm dynamically checks the local DoC of previous tasks, once the local DoC of fingerprints in a specific task group becomes greater than zero, the weights in this group will be frozen to prevent getting worse. The results are depicted in Figure 5.11. As depicted, even if the topological characteristics of old tasks are protected, but since there is no proper separation strategy, the accuracies of old tasks still decrease as learning moves on.

![Image](image2.png)

(a) Test accuracy at different stages

(b) Local DoC at different stages

**Figure 5.11:** Local Degree of Conflict of regular EWC with weight consolidation

These experiments show that without proper separation in the latent space of old and new tasks, the local DoC of new tasks can be far from optimal, while the topological characteristics of old tasks would be destroyed gradually when not protected.
Finally, I did the same experiment using the CSIL strategy, with the results given in Figure 5.12. Topologically, the local DoC of task-2, 3, 4 are closer to the optimal value while the local DoC of Task-5 fails to reach the optimal value. Further experiment shows that the problem can be easily fixed as by using a tiny amount of historical data. Therefore, CSIL still has better performance than state-of-art memoryless incremental learning algorithms.

![Test accuracy at different stages](image1)

![Local DoC at different stages](image2)

**Figure 5.12:** Local Degree of Conflict of CSIL

### 5.5 Concluding Remark

In this chapter, I analyze the catastrophic forgetting problem of incremental learning from a new and more thorough perspective, the conflict of class-specific fingerprints. I also propose a novel incremental learning algorithm without using historical data. My contributions are as follows:

1. I provide a new metric, Degree of Conflict (DoC), to measure the degree of topological maturity of DNN models and discover that one important cause for performance degradation in IL is the conflict of classes’ representative fingerprints, in which the fingerprints of different classes are with high cosine similarity, thereby
causing confusion. I also prove that a minimal confusion solution is to keep those fingerprints mutually orthogonal.

2. I show that the conventional IL schemes without using historical data, can lead to DNN models with low topological maturity and high DoC. I discovered and proved the principle rules to design a reliable incremental learning strategy.

3. Based on the theoretic analysis, I propose a new IL scheme, the CSIL, based on channel separation and topological control of devices’ fingerprints at different stages of learning. I evaluation my proposed solution using the raw signal records from more than 100 aircraft’s wireless transponders, and the experiments demonstrate that my CSIL strategy provides the best balance between learning new devices incrementally while retaining the memory of old devices.

I believe the CSIL and the metric for quantifying the topological maturity of DNN models can be generalized to other domains, such as virus detection or medical image classification.
Chapter 6

Deep Learning Enabled Quickest Event Detection with Application in Identity Spoofing Detection

In this chapter, I utilize an enhanced deep learning framework based on my previous work [71], the zero-bias deep neural network, for quick and reliable detection of abnormalities with assured performance. I use zero-bias dense layers to facilitate DNNs with both assured and explainable performance in distinguishing known or abnormal inputs. I model the topological characteristics of the latent space and design a parametric quickest detection algorithm. In this way, the abnormality detection system can rapidly react to abnormalities with minimum latency under specified false alarm constraints. Furthermore, my solution efficiently derives single-shot abnormality detectors from existing DNN classifiers. The effectiveness of the proposed framework in handling massive signal recognition has been demonstrated.
6.1 Regional Association Characteristics of Zero-bias Neural Network

As discussed in Section 5.1 and Equation 5.1, the cosine similarity matching feature of the zero-bias neural network is equivalent to the angular similarity matching of fingerprints and feature vectors on a high dimension unit hyperspherical surface. A 3-D example is depicted in Figure 6.1. Fingerprints divide the unit hyperspherical surface into several subregions, and I can reduce the dimension of fingerprints and use Voronoi Diagram [85] to visualize their decision boundaries as in Figure 6.2. The decision boundaries of a specific fingerprint denote the boundaries of a class. Herein, I will use the terms class boundaries and fingerprint’s boundaries alternatively.

For example, in the DNN enabled MNIST handwritten digit recognition [44], the network’s last dense layer is replaced by a zero-bias dense layer with $N_1 = 10$. The Voronoi diagrams at two stages (85% and 97% accuracies) using fingerprints in the zero-bias dense layer and feature vectors from the validation set are depicted in Figure 6.2 with solid blue lines representing the decision boundaries of topologically adjacent classes. Please note that the Voronoi diagram can not be applied directly to DNN models without adapting zero-bias dense layers.
From the observation, I conclude that during training, a DNN with a zero-bias dense layer learns to project input data from identical classes closer to the corresponding fingerprints and separate data from different classes far away. The feature extractors in prior layers and fingerprints in the zero-bias dense layer are optimized simultaneously. In DNNs with zero-bias dense layer, classification errors result from two perspectives.
• If fingerprints are not distantly separated, data from the corresponding classes are highly possible to get confused. This fact is verified in my previous work in [71].

• The prior layers are poorly trained and the feature vectors are sparsely projected as depicted in Figure 4.6a.

Please note that, firstly, even if I eliminate the magnitude and bias constant for each fingerprint, I do not need to worry about the capacity of zero-bias DNN models in distinguishing different classes. A numerical example for quantifying the maximum theoretic number of classes that a zero-bias DNN will reliably learn is given in Figure 6.5b. Secondly, zero-bias dense layer can be easily adapted to existing DNN models via transfer learning.

6.1.1 Abnormality detection in DNN with zero-bias dense layer

The effectiveness of zero-bias DNN for single-shot abnormality detection has been demonstrated in my prior results [71, 72], briefly, it is significant better than regular DNN and comparable to one-class SVM [86]. In this section, I deepen my previous research and present a solution to convert zero-bias DNN models into abnormality detectors with predictable and assurable performance.

6.1.1.1 Deriving Abnormality Detectors from Existing DNN Classifiers

In Figure 4.6b, feature vectors from known classes are closely projected to the vicinity of the corresponding fingerprints. As a result of cosine similarity matching, I come to my first remark:

Remark 6.1. Abnormal data from an unknown novel class are less likely to be projected into any existing classes’ close vicinity as there is no specific fingerprint for these data.
I use the MNIST example to demonstrate Remark 6.1 with the results in Figure 6.3. I train the zero-bias DNN to recognize handwritten digits from 1 to 8 and use digits 9 and 0 as abnormal data. The projection feature vectors of known and abnormal data and fingerprints are depicted in Figure 6.3a. I then replace the abnormal data with pure Gaussian random noise ($N(0, 2)$) and repeat the experiment. Results are depicted in Figure 6.3b.

![Figure 6.3: Stacked diagrams of: a) Voronoi graph of remapped fingerprints. b) Class decision boundaries. c) remapped validation and abnormal data feature vectors. Data are projected to a 2D space using t-SNE algorithm [1].](image)

From these results, abnormalities from an unknown class of the same domain could be even more difficult to detect than pure random noise. Although Figure 6.3a has shown that abnormalities from unknown classes are more sparsely distributed in the unit hyperspherical surface, Remark 6.1 still holds. Therefore, I can derive a basic principle (depicted in Figure 6.4) to convert a zero-bias dense layer enabled DNN classifier into an abnormality detector:

**Remark 6.2.** I can model the spatial distribution and boundaries of normal data in the hyperspherical surface. Then the incoming feature vectors that are out of normal data boundaries are regarded as abnormalities. For simplicity, I only need to find a hard cut-off distance for each class representative fingerprint (suppose each class fingerprint is at the centroid of its corresponding feature cluster).
For a given DNN model with zero-bias dense layer, I model the boundaries of different classes as follows:

**Step 1**: The training set is utilized to learn the boundaries of known classes while the validation set will be mixed with abnormal data ($A_0$) to measure the performance of converted abnormality detector.

**Step 2**: I pass accurately classified data of each known class in $C_1$, denoted as $KX_i$, through the DNN model and obtain the compressed feature vectors before fingerprint matching, denoted as:

$$Y_0[F_{n-1}(KX_i)] = W_0F_{n-1}(KX_i) + b \quad (6.1)$$

where $W_0$ and $b$ are defined in Equation 5.1, $F(\cdot)_{n-1}$ denotes all network layers before the fingerprint matching, $Y_0[F_{n-1}(KX_i)]$ denotes feature vectors of accurately classified data in $KX_i$. 
**Step 3:** Calculate the centroid $c_i^0$ and covariance matrix ($P_i$) of $KX_i$ as:

$$c_i^0 = \text{mean}(Y_0[F_{n-1}(KX_i)])$$

$$P_i = \text{cov}(Y_0[F_{n-1}(KX_i)], Y_0[F_{n-1}(KX_i)])$$

**Step 4:** Calculate the Mahalanobis distances [87] from the class centroid $c_i^0$ to all accurately classified feature vectors. Then I use the maximum value as a cut-off distance $CO_i$ of class $KX_i$:

$$CO_i = \max D_m[Y_0[F_{n-1}(KX_i)], c_i^0]$$ (6.3)

where $D_m(\cdot, c_i^0)$ denotes the feature vectors’ Mahalanobis distances to $c_i^0$.

**Step 5:** Abnormality detection using cut-off boundaries on input data $X$) is formally defined as:

$$D(X) = \begin{cases} 
1 & \exists i, \ D_m[Y_0[F_{n-1}(X)], c_i^0] \leq CO_i \\
0 & \text{Otherwise} 
\end{cases}$$ (6.4)

These steps convert zero-bias DNNs into abnormality detectors with binary outputs. In essence, this is a discretization process that transforms the continuous response characteristics of DNNs into two simple Bernoulli distributions. Herein, I will use the term zero-bias abnormality detector alternatively. In essence, I construct statistical models for each class to describe the distribution of corresponding normal data and a hard cut-off distance to form its boundary (denoted as dashed purple lines in Figure 6.4). Please note that other distance functions or modeling methods such as the Local Outlier Factor [88] can also be applied.
Suppose that each fingerprint governs a non-overlapped subregion with a maximum acceptable deviation angle, $\sigma$, for normal data. This subregion is named $\sigma$-cap, I can analytically evaluate the area of each $\sigma$-cap, and its occupied area ratio, $r_0(m)$, as:

\[
A_c(m) = \frac{1}{2} A_u(m) r^{n-1} I_{(2rh-h^2)/r^2} \left( \frac{m - 1}{2}, \frac{1}{2} \right) \\
A_u(m) = \frac{2\pi^{n/2}}{\Gamma(n/2)} \\
r_0(m) = \frac{A_c(m)}{A_u(m)} = \frac{1}{2} I_{(2h-h^2)} \left( \frac{m - 1}{2}, \frac{1}{2} \right) 
\]

where $A_c(m)$ is the area of a $m$-D $\sigma$-cap. $A_u(m)$ is the surface area of the $m$-D unit hypersphere. Additionally, I have $r = 1$ and $h = r - r\cos(\sigma)$. $I$ and $\Gamma$ are the regularized incomplete beta function and the gamma function, respectively. A numerical result is given in Figure 6.5. As depicted, both a smaller $\sigma$ and larger number of feature dimensions ($N_1$) increase the capacity and interclass distinguishability of the zero-bias DNN. Moreover, even if I eliminate some information of feature vectors in zero-bias DNNs, I do not have to worry much about their learning capacity and remaining space for abnormalities, as long as the feature dimension $N_1$ is large.

6.1.1.2 Theoretic Performance Analysis of the Single-Shot Binary Abnormality Detector

I introduce the hard cut-off distances from class fingerprints. Therefore, a binary abnormality detector converted from zero-bias DNN becomes a binary classifier. I derive two important properties of this type of zero-bias abnormality detector regarding false positive and false negative rates. This step is to determine the range of parameters of the coverted binary abnormality detector specified in Section 6.1.1.1, therefore, I define it as the parametrization process.
The accuracy of zero-bias DNN models on known classes can be obtained after training. From the perspective of decision boundary and class boundary, the scenarios that lead to classification error are depicted in Figure 6.6. As depicted, the feature vectors C and D are projected into the wrong class boundaries but out of the boundaries of normal data. Meanwhile, E and F are projected into the normal data boundaries of wrong fingerprints.

Suppose that E and F in Figure 6.6 are moved out of the normal data boundaries.

Figure 6.5: The coverage ration per class and maximum number of distinguishable class in zero-bias DNN.
The false positive rate of abnormality detection reaches its upper bound and equals the classification error $\alpha$. Furthermore, if C and D are moved into normal data boundaries, the false positive rate equals zero. Therefore, the range of false-positive rate of zero-bias abnormality detector is actually determined:

**Remark 6.3** (Range of the false positive rate). Suppose that the classification error of the zero-bias DNN is $\alpha$, as long as my statistical model can closely follow the boundary of normal data, the false positive rate of converted abnormality detector is less than or equals to $\alpha$. Denoted as:

$$FPR \leq \alpha \quad (6.6)$$

Suppose that in a regular case, the feature vectors of abnormalities are mixed with normal data and uniformly distributed on the surface of the unit hypersphere, in this case, the maximum false negative rate is reached.

**Remark 6.4** (Range of false negative (true positive) rates). The upper bound of the false negative rate under uniformly distributed abnormalities, equals to ratio of the
occupied regions’ area of normal data divided by the total surface area of the unit hypersphere, denoted as:

\[
RU_{FNR} = \frac{\sum_{i=1}^{N_c} S_{hsp}^i(N_1)}{A_{hsp}(N_1)}
\]

With \(FNR \leq RU_{FNR}\), \(TPR \geq 1 - RU_{FNR}\) \hspace{1cm} (6.7)

where \(N_c\) is the number of known classes, \(N_1\) and \(A_{hsp}(N_1)\) are the dimension and surface area of the unit hypersphere, respectively. \(S_{hsp}^i(N_1)\) is the surface area of normal data of the \(i\)th class.

Algorithm 1 Estimating \(RU_{FNR}\)

1. function \(RU_{FNR}(N_1, N_c, M, List[CO], List[c_0])\)  
2. \(HX \leftarrow UniformHypersphereRand(N_1, M)\) \hspace{1cm} \(\triangleright\) Please refer to [89]  
3. \(chx \leftarrow 0\)  
4. for \(k \leftarrow 1 \ldots M\) do  
5. \hspace{1cm} for \(i \leftarrow 1 \ldots N_c\) do  
6. \hspace{2cm} if \(D_m[HX_k, c_0] \leq CO_i\) then  
7. \hspace{3cm} \(chx \leftarrow chx + 1\)  
8. \hspace{2cm} end if  
9. \hspace{1cm} end for  
10. \hspace{1cm} end for  
11. return \(\frac{chx}{M}\)  
12. end function

Analytically calculating \(S_{hsp}^i(N_1)\) is difficult since the shapes of these occupied subregions are unknown. Therefore, I use Monte Carlo method to estimate \(RU_{FNR}\) directly. Corresponding pseudo code is presented in Algorithm 1, specifically, I generate \(M\) random points uniformly distributed on the surface of the unit hypersphere and count the ratio of points that are captured by the normal data boundaries of fingerprints. The captured rate directly indicates the value of \(RU_{FNR}\). Empirically, I set \(M = 20,000\).
6.2 Zero-bias DNN for Quickest Abnormal Event Detection

In this section, I will introduce the method to integrate a single-shot zero-bias binary abnormality detector with quickest detection algorithm.

6.2.1 Sequential Formalization And Detectability

Given the theoretic analysis of zero-bias abnormality detector in section 6.1.1.2, I can model the response of zero-bias DNNs as switching between two probability distributions before and after the appearance of an abnormal event, namely $P_0$ and $P_1$, respectively. Since I have converted the zero-bias DNN into a binary abnormality detector, I can formulate $P_0$ and $P_1$ into two Bernoulli Distributions [90]:

$$P_0(I_k) = FPR^I_k(1 - FPR)^{1-I_k}$$
$$P_1(I_k) = (1 - FNR)^I_k FNR^{1-I_k} = (TPR)^I_k(1 - TPR)^{1-I_k}$$

(6.8)

where $I \in \{0, 1\}$ is the binary output of the abnormality detector with $I_k = D(X_k)$. $FPR$ can be derived on existing data, and the range of $FNR$ and $TPR$ from section 6.1.1.2. As long as the $P_0$ and $P_1$ are different, the abnormal event causing drifts from $P_0$ to $P_1$ can be sequentially detected. I have the following determinant under regular scenarios:

**Remark 6.5** (Sequential detectability). Abnormal events are assured to be sequentially detectable if the binary zero-bias detector’s true-positive rate ($TPR$) lower bound $(1 - RU_{FNR})$ are greater than the false-positive rate ($FPR$) upper bound ($\alpha$).
Remark 6.4 shows that the true positive and false negative rates are within different ranges, if $1 - RU_{FNR} \leq \alpha$, the two variables’ spanning ranges are partially overlapped (depicted in Figure 6.7) and I may encounter an extreme case: $TPR = FPR$. Therefore, the abnormal event is only conditionally detectable. Please note that we can definitely use the Kullback–Leibler divergence [91] to measure the difference of the two Bernoulli distributions, but in this dissertation, our own definition can be more intuitive and specific.

![Figure 6.7: Range of true-positive and false-positive rates.](image)

6.2.2 Quickest Detection Algorithm

With Remark 6.5, I can use Quickest Detection algorithm to detect the appearance of an abnormal event with the lowest latency at a given false alarm run length. I will present both the Bernoulli Generalized Likelihood Ratio Chart and its approximation, the multiple Bernoulli CUSUM Chart, respectively. Compared with the existing nonparametric solutions, I discretize the continuous probabilistic function space and transform the problem into a parametric sequential hypothesis testing problem.
Using Bernoulli Generalized Likelihood Ratio (GLR) Chart \[92\] to sequentially detect abnormal events. I have:

\[
R_k = \max_{0 \leq \tau \leq k-1, 0 \leq TPR \leq 1} \ln \frac{\prod_{i=\tau+1}^k TPR^I_k (1 - TPR)^{I_k}}{\prod_{i=\tau+1}^k FPR^I_k (1 - FPR)^{1 - I_k}}
\]

\[= \max_{0 \leq \tau \leq k-1} (k - \tau) \ln \left[ \frac{\hat{TPR} \cdot \hat{TPR} (1 - FPR)}{FPR (1 - \hat{TPR})} + \ln \frac{1 - \hat{TPR}}{1 - FPR} \right] \]

where \(\hat{TPR} \approx TPR \in [1 - RU_{FNR}, 1)\) is the estimated true positive rate of zero-bias abnormality detector and \(\tau\) is the estimated time when an abnormal event happens. \(\hat{TPR}\) is dynamically estimated as follows:

\[
\hat{TPR} = \min \left\{ B_1, \max \left[ 1 - RU_{FNR}, \frac{\sum_{i=\tau+1}^k I_k}{k - \tau} \right] \right\} \quad (6.9)
\]

where \(B_1 = 1 - \varepsilon\) is the maximum possible value of \(TPR\) and \(\varepsilon\) is a tiny positive number to assure \(\hat{TPR} < 1\). An alarm is triggered if \(R_k > h_{GLR}\) and \(h_{GLR}\) is a pre-defined threshold. \(h_{GLR}\) can be chosen as suggested in: \[92\]:
be closely approximated with a countable set of Bernoulli CUSUM Charts, where the identical detection threshold $h_{CUSUM}$ is shared among them and $h_{CUSUM} = h_{GLR}$ [92, 94]. The approximated range of $TPR$ is covered by each CUSUM chart is:

$$\hat{TPR}_i = 1 - RU_{FNR} + \frac{TPR_{max} \cdot i^2}{U^2}$$

(6.11)

where $U$ is the total number of CUSUM charts, in which greater than 100 is recommended, $i$ denotes the index of each chart. $TPR_{max}$ is the max possible value of the true positive rate that is less than 1. $1 - RU_{FNR}$ denotes the lower bound of the true positive rate. Therefore, given an average run length between false alarms, $ARL$, I have the worst case average detection delay as:

$$\bar{T}_{GLR} = \bar{T}_{CUSUM} \sim \frac{h_{CUSUM}}{I(P_1, P_0)}$$

(6.12)

Please note that I use the characteristic of multiple Bernoulli CUSUM charts to demonstrate the properties of detection delay.

### 6.3 Zero-bias Deep Learning for Quickest Identity Spoofing Detection

#### 6.3.1 Introduction

Deep Learning (DL) has reformed the ecosystem of the Internet of Things (IoT). On the one hand, they have been successfully applied in smart devices for accurate recognition of complicated inputs [95–97]. On the other hand, deep learning models do not require high-quality features and reduce the time-consuming feature engineering in conventional
machine learning schemes [98, 99]. As a representative technology within the scope of DL, Deep Neural Network (DNN) classifiers aim to use hierarchically stacked convolution layers to extract latent features to make accurate decisions.

Although DL and DNNs are successful in general purpose applications, applying DNNs in safety-critical systems requiring assured performance is still controversial. Firstly, DNNs perform well on known subjects but cannot distinguish unseen abnormal data. Abnormal signals, such as cyberattacks, are required to identify in real-time with constrained false alarms [100]. Secondly, deep neural networks lack explainability, while applications in safety-critical systems require making accurate decisions with known and explainable behaviors. Thirdly, in safety-critical systems, classifiers are supposed to evolve efficiently within a manageable behavior. The three obstacles impede the deployment of DL and DNNs in IoT of safety-critical systems. Compared with DNNs, the nearest neighbor matching algorithms naturally overcome these obstacles and gain popularity in safety-critical systems.

To address the first challenge, existing works use deep Autoencoders or Generative Adversarial Networks (GANs) to capture the latent features of the domain-specific inputs by compressing and accurately reconstructing them. However, training deep autoencoders or GAN models is even more computationally expensive than training DNN classifiers on a specific domain. Moreover, autoencoders or GAN models do not guarantee to respond in time with constrained false alarms [101]. Despite that existing Quickest Detection (QCD) algorithms can detect changes with minimum latency under constrained false alarms, they are neither sufficient in handling complicated inputs nor can provide mathematically assured performance when directly combined with Black Box models. For the second problem, the eXplainable AI (XAI) has been proposed [102]. In this application, I use the zero-bias neural network to transparentize the decision process and decision
boundary. Finally, to support dynamic evolving DNN models, I have invented the CSIL algorithm as in Chapter 5. Therefore, I will focus mainly on the zero-bias deep neural network enabled quickest detection scheme.

### 6.3.2 Problem definition

The system model of my proposed framework is depicted in Figure 6.8. I aim to use a deep learning model to process heterogeneous data from IoT and spot the abnormal data. I then use the quickest event detection algorithm to detect ongoing abnormal events with minimum latency.

![Figure 6.8: System model of zero-bias deep learning enabled quick and reliable abnormality detection in IoT.](image)

In IoT, the system states are highly correlated with time-dependent events, e.g., abnormal events or normal operations. I define that abnormalities are suspicious data caused by abnormal events. Intuitively, abnormalities could trigger variation of specific indication metrics. Analyzing the drift or variations of these metrics, abnormal events can be detected sequentially. Therefore, I can convert the real-time abnormality detection
problem into an online sequential event detection scheme, in which a surveillance oracle can sequentially collect its target system’s state signals or heterogeneous data denoted as:

\[ X = \{X_1, X_2, \ldots X_j \ldots X_{j+m} \ldots \} \] (6.13)

where \( X_j \) denotes a state variable or record in vector form, an abnormal event appears at \( j \) and disappear at \( j+m \). Real-time abnormal event detection requires triggering an alarm before \( j+m \) with minimum assured latency.

Well-known methods are provided in the Quickest Change Detection (QCD) theory. For example, in the Cumulative Sum (CUSUM) Control Chart algorithm, a likelihood ratio test is employed to sequentially process the observed data at each timestamp \( k \), denoted as:

\[ g(k) = \ln \left( \frac{P_1(X_k)}{P_0(X_k)} \right) \] (6.14)

where \( g(k) \) is a sufficiency metric, \( P_0(\cdot) \), \( P_1(\cdot) \) denotes the probabilistic density functions of abnormal and abnormal states, respectively. A constrained cumulative sum of sufficiency metrics is used as an indicator, denoted as:

\[ S(k) = \max(0, S(k-1) + g(k)) \] (6.15)

An alarm will be sent once \( S(k) \) is greater than a predefined threshold, \( h \). The CUSUM algorithm has been proved to provide the lowest worst-case detection latency at specific false alarm intervals. However, CUSUM-style quickest detection algorithms can hardly handle high-dimension data, where \( P_0(\cdot) \) and \( P_1(\cdot) \) are difficult to obtain. Even though
some works use DNNs to derive the sufficiency metric $g(k)$ from high dimension data, the DNNs’ uncertain responses when encountering abnormalities make performance assurance a theoretic challenge. To enable deep learning for quick and reliable abnormality detection, the following efforts are needed:

- I need a DNN driven abnormality detection model to process complex data and provide theoretically assured performance. If possible, the deep abnormality detection model should be derived without a large overhead.
- I need to develop an efficient method to jointly apply performance-assured DNN and quickest event detection to provide theoretically guaranteed performance in detecting abnormal events.

### 6.3.3 Performance Evaluation

In this section, I evaluate the performance of the proposed framework in two folds, I first use the wireless signal dataset in Section 5.3.2 to test the proposed method for the class decision boundary and feature vector visualization. Then I use the proposed method to convert the wireless device recognition DNN into an quickest detection model and evaluate its performance for identity spoofing detection.

As presented in my prior work [71, 72], I take the first 1024 samples from the signal of each intercepted message. And convert the 1024 samples into a 32 by 32 by 3 tensor, which incorporate pseudo noise, magnitude-frequency domain, and phase-frequency domain information, respectively.
6.3.3.1 Decision Boundaries in Real-World Zero-Bias DNN

The architecture of my DNN model is depicted in Figure 4.11 with a description of the dataset in Table 6.1. In Figure 6.10 I use two stacked Voronoi diagrams to depict the relation of fingerprints, class boundaries, normal and abnormal data, with the DNN model trained under two scenarios: a) Input signals are polluted by abrupt spike noise due to the signal interference. b) Input signal without abrupt spikes (removed by a Gaussian filter).

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>60% of signals, including 28 aircraft with more than 500 randomly selected raw records for each.</td>
</tr>
<tr>
<td>Validation</td>
<td>40% of signals, including 28 aircraft with more than 500 randomly selected raw records for each.</td>
</tr>
<tr>
<td>Normal data</td>
<td>Validation data that’s not been used to train the network.</td>
</tr>
<tr>
<td>Abnormal data</td>
<td>Signals including 236 aircraft with less than 200 raw signal records during the data collection period.</td>
</tr>
</tbody>
</table>

As shown in Figure 6.10a and 6.10b, normal data are closely distributed within their fingerprints, while abnormalities are sparsely distributed over the feature space. It is interesting to find that if the DNN model is with high accuracy, the abnormalities are less likely to appear in normal data clusters. The DNN signal identifier is with high accuracy within the two scenarios.
According to Remark 6.3 and 6.4, the abnormality detector’s performance is predictable. In reality, this single-shot zero-bias abnormality detector trained on noisy data has a true positive and a true negative rate of 91% and 92%, respectively, which is closely matched with my prediction. However, the zero-bias abnormality detector trained on the filtered data has a true positive and a true negative rate of 99% and 91%. The true negative rate is smaller than the expected value due to the model entering the early stage of overfitting. At this stage, the training set’s feature vectors can no longer provide
sufficient information on the distribution of normal data. Therefore, the estimation of normal data will be misled.

For alleviation, I can set a threshold value for accuracy during training. I derive the cutoff distances, centroids, and covariance matrices at this point. By setting a triggering value of 96% for the validation accuracy on the filtered data, I get a true-positive rate and a false-positive rate of 98% and 95%, respectively. The relation between the performance of the converted abnormality detector and the zero-bias DNN model’s accuracy before conversion is given in Figure 6.11. As predicted, when the accuracy of zero-bias DNN gets higher, the normal data occupies a smaller amount of area on the unit hypersphere surface, and thus produce higher True Positive rates. Meanwhile, lower False Positive rates are achieved when the zero-bias DNN has higher classification accuracy. According to Remark 6.5, even when the accuracy is less than 75%, the identity spoofing attack is still in the sequentially detectable range.

![Figure 6.11: Performance of the converted abnormality detector.](image)

### 6.3.3.2 Quickest Abnormal Event Detection with Zero-Bias DNNs

My model can detect abnormalities (the appearance of an unknown aircraft’s signal) with almost neglectable latency (less than ten samples on average using GLR chart) as a result of both high true-positive and true-negative rates. To further evaluate my proposed
method, I can use numerical simulation results to demonstrate the performance of zero-bias DNN, since its response characteristics prior and after abnormal events are modeled as two Bernoulli distributions in Section 6.2.1. I experiment with a collection of possible values of $h_{GLR}$, $FPR$, and $TPR$ that a zero-bias abnormality detector can encounter. In which $TPR \in [0.6, 0.99]$, $FPR \in [0, 0.4]$ and $h_{GLR} \in [2, 20]$ The relationship between abnormal event detection latency and false alarm rate is depicted in Figure 6.12a. I discover a nice cut-off property, in which the average false alarm rate becomes zero as
I select a proper detection threshold $h_{GLR}$ or $h_{CUSUM}$. After the threshold is properly set. Once the detection threshold is greater than a certain value, 12 in my experiment, the detection delay grows linearly as depicted in Figure 6.12b. A further analysis of the distribution of detection delay with threshold $h_{GLR} \in [12, 20]$ is presented in Figure 6.13, in general, as $TPR/FPR$ gets larger, average detection latency decreases with less sensitive to $h_{GLR}$.

### 6.4 Concluding Remark

In this chapter, I significantly extend the analysis of my previously proposed zero-bias DNN and combine it with the Quickest Detection algorithms to detect abnormalities and time-dependent abnormal events in IoT with the lowest assured latency. I first use the zero-bias neural networks and use Voronoi diagrams to analyze their latent space and derive the region association characteristics. I then provide a solution to convert zero-bias DNN classifiers, which are easier to obtain, into performance assured binary abnormality detectors with assured performance boundaries. Using the converted abnormality
detectors, I model their behavior using Bernoulli distribution, which perfectly adapts to the Generalized Likelihood Ratio Test based Quickest Detection scheme. In this Quickest Detection scheme, the theoretically assured lowest abnormal event detection delay is provided with predictable false alarms. Finally, I demonstrate the framework’s effectiveness using both massive signal records from real-world aviation communication systems and simulated data.
Chapter 7

Concluding Remark and Future Work

7.1 Concluding Remark

In this dissertation, I aim to find a solution to seamlessly bridge the two domains: Quickest Detection and Machine Learning. On the one hand, Quickest Detection has an advantage for detecting events that cause state change even when there is random noise, however, Quickest Detection algorithms are not efficient in handling high dimension data and usually require manual and time-consuming feature engineering. On the other hand, machine learning methods, especially deep learning, are capable of handling heterogeneous data with high dimensions. I finally reach an integral solution with incremental learning capability, the corresponding findings are as follows:

1. **Zero-bias Neural Network**: I thoroughly studied the characteristic of the classification layer of deep neural networks. And I invented the zero-bias dense layer for
zero-bias and equal weight decision. The zero-bias neural network can distinctive identify the known and unknown classes with a clear boundary. In this dissertation, the zero-bias neural networks provides a very convenient way to analyze the topological characteristics the latent space of deep neural networks.

2. **Topological Maturity and Orthogonal Memory Organization**: Based on the zero-bias neural network, I discover and mathematically prove that neural networks organize class representation vectors (class fingerprints) in the latent space with a confusion (interference) minimization manner, in which the class fingerprints are approximating a mutually orthogonal relation. I then provide a metric, the degree of conflicts, to quantitative measure the topological maturity of deep neural networks. This metric provides an important clue for clearly explain catastrophic forgetting phenomenon during incremental learning.

3. **Channel Separation Incremental Learning**: Based on the orthogonal memory organization theorem, I invent a new incremental learning algorithm, CSIL, to facilitate DNNs to evolve incrementally to adapt to operational variation.

4. **Integration of Quickest Detection and Deep Neural Network**: I use zero-bias neural network as an early warning generator for anomalous event detection. I use the Voronoi diagram to analyse the regional association characteristics of the latent space of zero-bias neural network. I found that the zero-bias neural classifiers can be transformed into corresponding abnormality detectors with known performance boundaries. Even better, the response characteristics formulated using Bernoulli Distributions. In this way, I can integrate parametric quickest detection with deep neural networks seamlessly and mathematically provide theoretic assurance on performance boundaries.
Chapter 7. Concluding Remark and Future Work

7.2 Future Works

7.2.1 Unsupervised Deep Learning Driven Quickest Detection Algorithms

In terms of Quickest Detection, my method is a semisupervised learning approach, it is extremely helpful in developing heuristic cyber defense systems in which a small proportion of prior knowledge about the attack scene is required, and those small proportion of prior knowledge is useful for constructing the initial topology of the latent space.

However, in some more challenging scenarios, I face the zero-knowledge situation at the beginning. In these cases, the system needs to employ unsupervised learning algorithms to automatically explore and build the initial topology. In my plan, I first use the deep autoencoder with the zero-bias dense layer as the bottleneck layer to explore the existing normal data space. Then I can use the zero-bias bottleneck layer and the extracted encoder to form an early warning generator. Such an early warning generator would be trained incrementally as what I have done in this dissertation. I believe the Converging Point Theorem (Theorem 5.11) would be a good indicator for us to evaluate whether an autoencoder maturely trained.

7.2.2 Incremental Learning With Adaptive Network Expansion

In this dissertation, I have used the characteristics of zero-bias neural network and latent space optimization to uncover the essence of catastrophic forgetting and provide a better incremental learning algorithm. It should be noted that I have locked the prior layers as in other works [103–105]. However, locking prior layers could lead to severe problems when the prior layers are becoming obsolete. Unlocking or expanding the prior layers
would cause problems when the whole topological characteristics of the latent space will change and even the fingerprints of old classes have to be adjusted dynamically and simultaneously. In essence, this will be a bi-level optimization problem. Even though the deep learning society is eagerly working towards the problem as in [78], incremental learning with full architecture learning is still very challenging.

In my future work, I will borrow some ideas from the Deep Model Consolidation [78]. I will initialize two networks with an identical structure and then train the two networks on different tasks, I will combine the techniques of deep model consolidation and channel separation incremental learning to develop a more reliable incremental learning strategy.
Bibliography


[2] Jian Wang, Xuejun Yue, Yongxin Liu, Houbing Song, Jiawei Yuan, Tianyu Yang, and Remzi Seker. Integrating ground surveillance with aerial surveillance for enhanced amateur drone detection. In *Disruptive Technologies in Information Sciences*, volume 10652, page 106520E. International Society for Optics and Photonics, 2018.


