Liquid NeurIoTic: Liquid Neural Network-Based Intrusion Detection System for Internet of Things in Healthcare Networks

by John Michael Kramarczyk

B.A. in Business Administration, May 2020, North Park University
M.S. in Data Science with specialization in Artificial Intelligence, June 2022, Northwestern University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

August 31, 2024

Praxis directed by

Ould Vadel
Professorial Lecturer in Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that John Michael Kramarczyk has passed the Final Examination for the degree of Doctor of Engineering as of July 25, 2024. This is the final and approved form of the Praxis.

Liquid NeurIoTic: Liquid Neural Network-Based Intrusion Detection System for Internet of Things Networks

John Michael Kramarczyk

Praxis Research Committee:

Dr. Ould Vadel, Professorial Lecturer in Engineering and Applied Science, Praxis Director

Dr. Amir Etemadi, Associate Professor of Engineering and Applied Science, Committee Chair

Dr. Alexis Starr, Professorial Lecturer in Engineering and Applied Science, Committee Member

© Copyright 2024 by John Michael Kramarczyk
All rights reserved

Dedication

The author dedicates this research to his extraordinary mother, Son Kramarczyk, whose selfless love and guidance have made this journey possible–without her, none of this would be achievable.

The author also dedicates this work to his beloved wife, Ginsey, whose unwavering love and encouragement have fueled his pursuit of success, and to his joyful daughter, Yoella, whose infectious laughter and vibrant spirit brighten the lives of those around her.

Acknowledgments

The author extends his gratitude to his advisor, Dr. Ould Vadel, for his support and direction throughout the research phase. The author would also like to thank Dr. Hamza Alsarhan for his guidance throughout the praxis proposal phase.

Abstract of Praxis

Liquid NeurIoTic: Liquid Neural Network-Based Intrusion Detection System for Internet of Things in Healthcare Networks

Over 249.09 million individuals were affected by healthcare cyberattacks between 2005 and 2019. Alarmingly, a 95% probability exists that distributed denial-of-service (DDoS) attacks on the Internet of Things in Healthcare (IoT-H) devices pose a life-threatening risk. Despite notable progress in the field, current solutions for neural networks (NN) with IoT-H networks are flawed by issues regarding scalability and IoT device resource management. As a viable alternative to conventional NNs in detecting intrusions in IoT-H networks, this praxis introduces Liquid NeurIoTic, a liquid neural network (LNN)-based intrusion detection system for IoT-H networks.

This praxis used the CICIoT2023 dataset to train and validate the LNN studied by identifying and categorizing distributed denial-of-service, denial-of-service, Mirai, spoofing, and reconnaissance attacks. This praxis also provided a novel technique to convert network traffic features into cyclical waveforms to enable time-series representations for enhanced integration into an LNN model. As a result, Liquid NeurIoTic reduced the number of neurons by as much as 40% and the number of epochs by as much as 64% compared to the best-performing artificial and recurrent neural networks. It also achieved a 61.81% reduction in training time. Additionally, Liquid NeurIoTic stood out in the broader NN research landscape because it used fewer neurons, epochs, and parameters than models in the existing literature and exceeded performance in accuracy, recall, precision, and F1-score metrics.

Table of Contents

Dedication ..... **iv**

Acknowledgments ..... **v**

Abstract of Praxis ..... **vi**

Table of Contents ..... **vii**

List of Figures ..... **xi**

List of Tables ..... **xii**

List of Symbols ..... **xiii**

List of Acronyms ..... **xvi**

Chapter 1— Introduction **1**

1.1 Background 1

1.2 Research Motivation 4

1.3 Problem Statement 5

1.4 Thesis Statement 6

1.5 Research Objectives 6

1.6 Research Questions and Hypotheses 7

1.7 Scope of Research 8

1.8 Research Limitations 8

1.9 Organization of Praxis 8

Chapter 2— Literature Review **10**

2.1 Introduction 10

2.2 Vulnerabilities in Internet of Things Devices in Healthcare 10

2.3 Prevalent Cybersecurity Threats in the Internet of Things in Healthcare Networks 12

2.3.1 Denial-of-Service and Distributed Denial-of-Service Attacks 12

2.3.2 Mirai Attack 14

2.3.3 Spoofing Attack 15

2.3.4 Reconnaissance Attack 16

2.4 Categories of Intrusion Detection Systems 17

2.5 Neural Network-Based Intrusion Detection Systems for Internet of Things in Healthcare Networks 20

2.6 Liquid Neural Networks 26

2.6.1 Foundations of Liquid Neural Networks 26

2.6.2 Liquid Time-Constant Networks 27

2.6.3 Closed-Form Continuous-Depth Networks 31

2.6.4 Neural Circuit Policies 32

2.7 Summary and Conclusion 34

Chapter 3— Methodology 36

3.1 Introduction 36

3.2 Canadian Institute for Cybersecurity Dataset 37

3.2.1 Dataset Collection and Integration 39

3.2.2 Data Structure 39

3.3 Exploratory Data Analysis 41

3.4 Data Preparation 42

3.4.1 Grouping Attacks by Class 42

3.4.2 Feature Importance Selection 43

3.4.3 Splitting Data 45

3.4.4 Stratified Random Sampling 46

3.4.5 Random Undersampling 47

3.4.6 Data Distribution Similarity Test 48
3.4.7 Feature Scaling 49
3.4.8 Converting Standardized Features Into Waveforms 50
3.5 Neural Network Models 51
3.5.1 Artificial Neural Networks 52
3.5.2 Recurrent Neural Network 54
3.5.3 Liquid Neural Network 56
3.5.3.1 Closed-Form Continuous-Time Networks 57
3.5.3.2 Neural Circuit Policies 58
3.6 Neural Network Hyperparameters 61
3.7 Performance Evaluation Metrics 62
3.8 Summary and Conclusion 64
Chapter 4— Results and Discussions 66
4.1 Introduction 66
4.2 Exploratory Data Analysis 67
4.2.1 Identify Data Types 67
4.2.2 Uncovering Null Values 68
4.2.3 Collecting Summary Statistics 68
4.2.4 Feature Importance Selection 69
4.3 Data Preparation 71
4.3.1 Grouping Attacks by Class 71
4.3.2 Splitting Data 73
4.3.3 Stratified Random Sampling 73
4.3.4 Random Undersampling 74

4.3.5 Data Distribution Similarity Test Using Kolmogorov-Smirnov..... 76
4.3.6 Feature Scaling and Conversion into Waveforms 77
4.3.7 One-Hot Encoding of Classes..... 78
4.4 Neural Network Models Development 79
4.4.1 Optimal Hyperparameters and Architecture 79
4.5 Neural Network Model Results 83
4.5.1 Artificial Neural Network Results 83
4.5.2 Recurrent Neural Network Results 84
4.5.3 Liquid Neural Network Results 85
4.5.4 Superior Results of Liquid Neural Networks 86
4.5.5 Liquid NeurIoTic Compared to Existing Literature 88
4.6 Summary and Conclusion 89
Chapter 5— Conclusions..... 91
5.1 Discussion 91
5.2 Final Conclusions..... 92
5.3 Contributions to Body of Knowledge 93
5.4 Recommendations for Future Research 94
References..... 96
Appendix A..... 122

List of Figures

Figure 3-1: Liquid NeurIoTic Architecture 36

Figure 3-2: CfC Neuron Architecture 58

Figure 3-3: Example NCP Architecture with Flatten and Dense Layers 59

Figure 4-1: Distribution of 33 Subtypes Grouped Into Seven Main Classes 72

Figure 4-2: Original Distribution of CICIoT2023 Classes 73

Figure 4-3: Validation Set Distribution After SRS 74

Figure 4-4: Test Set Distribution After SRS 74

Figure 4-5: Training Set Balanced After Random Under Sampling 75

Figure 4-6: Validation Set After Removal of Web and Brute Force Attack Classes 76

Figure 4-7: Test Set After Removal of Web and Brute Force Attack Classes 76

Figure 4-8: DDoS and Recon Attacks Converted Into Stacked Waveforms 78

Figure 4-9: Optimal ANN Architecture With 30 Neurons 81

Figure 4-10: Optimal RNN Architecture With 20 Neurons 81

Figure 4-11: Optimal NCP-CfC Architecture With 18 Neurons 82

List of Tables

Table 3-1: CICIoT2023 Dataset Features with Descriptions 40

Table 3-2: ANN Configurations and Sizes 52

Table 3-3: RNN Configurations and Sizes 55

Table 3-4: NCP-CfC Configurations and Sizes 61

Table 3-5: Hyperparameter Values of Developed ANN, RNN, and LNN Models 61

Table 4-1: Data Types in the CICIoT2023 Dataset 67

Table 4-2: Summary Statistics for the Features of the CICIoT2023 Dataset 68

Table 4-3: Mutual Information Scores of CICIoT2023 Dataset Features 70

Table 4-4: Kolmogorov-Smirnov Test Results 77

Table 4-5: One-Hot Encoded Representation of All Classes 79

Table 4-6: Optimal Hyperparameters for Each Model Developed 80

Table 4-7: Top ANNs Before and After Liquid NeurIoTic Waveform Conversion 83

Table 4-8: Top RNNs Before and After Liquid NeurIoTic Waveform Conversion 84

Table 4-9: Top NCP-CfCs Before and After Liquid NeurIoTic Conversion 85

Table 4-10: Superior Results of Liquid Neural Networks 86

Table 4-11: Superior Results of Liquid NeurIoTic Compared to the Existing Literature 88

Table A1: Top 25 ANN Results 122

Table A2: Top 25 RNN Results 123

Table A3: Top 25 NCP-CfC Results 124

List of Symbols

b	bias term
c	class belonging to the set of C
c	number of accurately predicted samples
C	set of all classes
C_{m_i}	membrane capacitance (potential to change state) of neuron i
cos	generates cosine waveforms
E_{ij}	excitatory or inhibitory synapses between neurons i and j
f	function that determines how an RNN is updated
f	liquid time-constant network
$F_{1,n}(x)$	eCDF of sample one with n observations
$F_{2,m}(x)$	eCDF of sample two with m observations
FN_c	number of false negative predictions for class c
FP_c	number of false positive predictions for class c
g	recurrent neural networks
h	recurrent neural networks
$H(X\|Y)$	conditional entropy of X given Y
$H(X)$	entropy of X
I	input sequence
$I(X;Y)$	mutual information between X and Y
j	neuron which gives signals or influences to neuron i
k	class from 1 to k

n	size of the entire sample
n	total number of inputs to the neuron
n_i	sample size of the stratum
N	total population size
N_i	total number of elements in the i^{th} stratum
p_e	expected agreement when both annotators assign labels randomly
p_k	instances where class k was predicted
p_o	empirical probability of agreement
$Precision_{micro}$	$\frac{Total\ True\ Positives}{Total\ True\ Positives+Total\ False\ Positives}$
$Recall_{micro}$	$\frac{Total\ True\ Positives}{Total\ True\ Positives+Total\ False\ Negatives}$
sup_x	supremum or largest value over all possible values of x
t_k	actual occurrences of class k
TN_c	number of false negative predictions for class c
TP_c	number of true positive predictions for class c
w_c	$\frac{Number\ of\ samples\ in\ class\ c}{Total\ number\ of\ samples}$
w_i	weight associated with the i-th input
w_{ij}	synaptic weight (connection strength) from neuron i to j
x_{leak_i}	potential of neuron i when not activated
x_i	i-th input
x_i	i-th standardized input value
$x_i(t)$	state of neuron i at any given time t
$\dot{x}_i$	rate of change of state $\dot{x}_i$ of neuron i over time

$\sigma_i\left(x_j(t)\right)$	sigmoid activation function applied to neuron j at time t
τ_i	time-constant of neuron i
$\odot$	Hadamard product that combines the gated outputs of g and h
2π	scales the standardized input data into a full circle
s	number of samples
sin	generates sine waveforms
t	input to the RNN at time t
t	time
x	hidden state
$x(t)$	state of the RNN at time t
$x(t+1)$	updated state of the RNN at the next time step $t+1$
z	z-score or standardized data points
θ	parameters of neural networks
θ	parameters of the function f learned during training
μ	mean of the data points
σ	sigmoid function bounds output between 0 and 1
σ	standard deviation of the data points
ϕ	activation function

List of Acronyms

ACK	acknowledge
Adam	Adaptive Moment Estimation
AIDS	anomaly-based intrusion detection system
ANN	artificial neural network
API	application programming interface
ARC	Argonaut reduced instruction set computer core
ARP	address resolution protocol
AUC-ROC	area under the receiver operating characteristic
AVG	average
BPTT	backpropagation through time
CAGR	compound annual growth rate
CCE	categorical cross entropy
CfC	closed-form continuous-time
CIC	Canadian Institute for Cybersecurity
CNN	convolutional neural network
CPU	central processing unit
CSI	channel state information
CSV	comma-separated value
CT-GRU	continuous-time gated recurrent unit
CT-RNN	continuous-time recurrent neural network
CVE	common vulnerabilities and exposures
DDoS	distributed denial-of-service

DE	differential equation
DHCP	Dynamic Host Configuration Protocol
DL	deep learning
DMA	detect malware attacks
DNN	deep neural network
DNS	domain name system
DoS	denial-of-service
DT	decision tree
eCDF	empirical cumulative distribution functions
ELF	executable and linkable format
F1	F beta set to 1
GHz	gigahertz
HIDS	host-based intrusion detection system
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAT	inter-arrival times
ICMP	Internet Control Message Protocol
IDS	intrusion detection system
IoHT	Internet of Healthcare Things
IoMT	Internet of Medical Things
IoT	Internet of Things
IoT-H	Internet of Things in Healthcare
IP	internet protocol

ISP	internet service provider
K-S	Kolmogorov-Smirnov
KNN	K-nearest neighbors
LiDAR	light detection and ranging
LLC	logical link control
LNN	liquid neural network
LR	logistic regression
LSTM	long short-term memory
LTC	liquid time-constant network
MAC	media access control
MCC	Matthews correlation coefficient
MI	mutual information
MIMO	multiple-input multiple-output
MitM	man-in-the-middle
ML	machine learning
MLOps	machine learning operations
MNIST	Modified National Institute of Standards and Technology
MSE	mean squared error
NCP	neural circuit policy
NIDS	network-based intrusion detection system
NN	neural network
ODE	ordinary differential equation
OS	operating system

PCAP	packet capture
PHI	protected health information
PHP	hypertext processor
PII	personally identifiable information
PM	particulate matter
PSO	particle swarm optimization
QR	quantile regression
RF	random forest
RFID	radio-frequency identification
RGB	red-blue-green
RMSE	root mean squared error
RNN	recurrent neural network
RQ	research question
RUS	random undersampling
SIDS	signature-based intrusion detection system
SMTP	Simple Mail Transfer Protocol
SRS	stratified random sampling
SSH	secure socket shell
STD	standard deviation
SVM	support vector machine
SYN	synchronize
Tanh	hyperbolic tangent
TCP	Transmission Control Protocol

ToN	telemetry operating systems and network
TS	timestamp
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
XML	Extensible Markup Language
XOR	exclusive or

Chapter 1—Introduction

1.1 Background

The Internet of Things (IoT) is a life-changing network of interconnected devices that seamlessly integrates the digital and physical realms to create endless possibilities for smarter living and innovative solutions across diverse industries (Javaid & Khan, 2021). The value of the IoT lies in its ability to automate, optimize, and bring intelligence to various aspects of daily and commercial life (Kelly et al., 2020). The IoT grew exponentially between 2010 and 2023 as the global number of IoT devices surged from 800 million units to 15.1 billion units, a whopping 1,787.5% increase in less than 15 years (Rahmani et al., 2022; Vailshery, 2023). By 2025, projections estimate that the total number of IoT units will escalate an additional 141% to reach 75 billion worldwide (Vailshery, 2023). By 2028, the global IoT market is poised to grow at a compound annual growth rate (CAGR) of 13.6% to reach a valuation of USD 2.23 trillion (Statista Market Insights, 2023). The rapid growth of IoT is fueled by the plummeting costs of sensors and processors, the proliferation of high-speed internet access, and the capability to enhance personalized healthcare, streamline manufacturing, and create intelligent homes (Rahmani et al., 2022).

The Internet of Things in Healthcare (IoT-H) market is also rapidly growing. By 2027, the global IoT-H market is projected to grow at a CAGR of 18.5% to reach a valuation of USD 284.5 billion (Faruqui et al., 2023). Unlike Internet of Medical Things (IoMT)/Internet of Healthcare Things (IoHT) networks, IoT-H networks concern all IoT devices connected to a healthcare provider's network, including non-healthcare devices like cameras, printers, routers, and computers (Ibarra-Esquer et al., 2017; Natarajan et al.,

2023). Due to IoT devices, enhanced patient care and operational efficiency have increased to provide key benefits like real-time clinical monitoring, inventory management with smart RFID tags, optimized resource allocation through patient flow tracking, personalized patient care, reduced healthcare costs, and increased accessibility of healthcare services worldwide (Kelly et al., 2020; Rejeb et al., 2023).

While integrating IoT-H technology has undeniably enhanced patient care and healthcare management, it has concomitantly led to a surge in cybersecurity incidents in the healthcare sector (Liu et al., 2015). Between 2005 and 2019, cyberattacks were the leading cause of all personally identifiable information (PII) and protected health information (PHI) breaches in healthcare, impacting more than 249.09 million individuals (Seh et al., 2020; Williams et al., 2020). These cyberattacks increased the vulnerability of IoT-H networks because every IoT device added to a healthcare network became a new gateway for cyber threats, which inadvertently multiplied the potential entry points for attackers to compromise systems (Mejía-Granda et al., 2023). In other words, the interconnectedness of IoT-H devices exposed them to the same security risks as other networked systems since they contained non-medical devices and were connected to the network in the same manner as standard computers (Mejía-Granda et al., 2023).

The potential for financial gain has primarily driven cyberattacks on healthcare institutions. However, even more alarming is that security breaches in IoT-H devices have threatened clinical care and patient safety, leading to increased mortality rates (Cartwright, 2023; Collier, 2022). For example, IoT-H networks are susceptible to distributed denial-of-service (DDoS) attacks like any non-healthcare IoT network (Alabsi et al., 2023). DDoS attacks involve overwhelming a network by flooding it with traffic

from multiple compromised devices to disrupt the service of a targeted resource (Wang et al., 2018). The interconnected nature of IoT-H devices can amplify such attacks, as cybercriminals harness the collective bandwidth of numerous insecure non-medical IoT devices to generate large-scale assaults that often render essential services inoperative (Alabsi et al., 2023). With a 95% probability of causing fatalities, DDoS attacks on IoT-H devices present a life-threatening scenario (Somasundaram & Thirugnanam, 2021).

A sobering example of the vulnerabilities in healthcare caused by DDoS attacks happened in 2014 at Boston Children's Hospital due to attacks by the hacktivist group Anonymous (Nigrin, 2014). These DDoS attacks disrupted the hospital's network, including affiliated institutions that shared the same internet service provider (ISP), such as Harvard Medical School and Massachusetts General Hospital, causing widespread internet outages for nearly a week (Nigrin, 2014). Throughout the week of outages, the intensity and evolving nature of the DDoS attacks gradually escalated and culminated in a peak where the influx of malicious traffic was substantial enough to hinder legitimate network activity (Nigrin, 2014). This hindrance prevented patient and medical personnel from accessing vital online medical records and services, with some unable to use their online accounts to view appointments, test results, and other vital information (Nigrin, 2014). Financially, the hospital incurred over $300,000 in damages and response costs (Nigrin, 2014).

This cyberattack illustrates a digital age axiom: no system is trivial in the interconnected world of the internet. Every system, irrespective of its scale or access to sensitive information, plays a pivotal role in fortifying the cyber resilience of IoT devices in healthcare. This attack could have been averted if the hospital employed more

sophisticated cybersecurity tools, such as an advanced intrusion detection system (IDS) capable of adapting to the evolving nature of cyber threats like the various forms of DDoS attacks against Boston Children's Hospital that changed over the course of a week.

Therefore, this praxis presents a novel liquid neural network (LNN)-based IDS for IoT-H networks to address the shifting patterns of cyber threats. LNNs offer a dynamic and adaptive architecture capable of real-time learning and classification of IoT malware, which is crucial to keep up with the constantly changing landscape of malware targeting IoT-H devices, such as DDoS attacks. An LNN-based IDS can detect known threats and new patterns of anomalies indicative of malware while providing a predictive and pre-emptive security posture. The key feature of an LNN-based IDS is its ability to continually adapt and learn from new data and evolving threats. Therefore, as new attacks and tactics emerge, the model can continually learn to stay effective, even after training.

1.2 Research Motivation

In 2022, more than 112 million cyberattacks were launched on IoT devices globally (Petrosyan, 2023). This number substantially rose from approximately 32 million reported incidents in 2018, marking an 87% annual growth in IoT malware cases (Petrosyan, 2023). IoT-H networks face a higher likelihood of security challenges than any other industry because many services are ill-equipped to handle the intricacies of medical devices (Argaw et al., 2020; B. Bhushan et al., 2023). Furthermore, medical data is estimated to be 50 times more valuable than other industries (B. Bhushan et al., 2023). Medical records contain sensitive health information and provide avenues for fraudulent access to financial accounts, such as banking and credit card accounts, making them more susceptible to risks like identity theft (Raghupathi et al., 2023).

Furthermore, a major limitation of all IoT devices in healthcare is their restricted storage and computational capabilities, which compromises the inclusion of robust cybersecurity measures like NNs that can keep up with the changing dynamics of attacks (Faruqui et al., 2023). Lastly, threat actors have recently begun to offer DDoS-as-a-service. Groups like Passion group and KillNet have coordinated attacks on healthcare providers in the United States and Europe as part of a retaliatory campaign against nations supporting Ukraine (Davis, 2023; Rohner, 2023). KillNet, in particular, is actively targeting the U.S. health sector and is seeking to expand its reach by recruiting via public channels (Rohner, 2023).

1.3 Problem Statement

The problem statement of this praxis is as follows: *With nearly 50% of all IoT-H devices vulnerable to cyberattacks (Ghubaish et al., 2021), 90% of IoT-H-utilizing healthcare organizations have experienced at least one security breach, which has put patients at risk of identity theft, financial trouble, and life-threatening consequences (Sadhu et al., 2022).*

Cyberattacks are the primary source of all data breaches in healthcare (Jiang & Bai, 2019; Williams et al., 2020). The number of individuals affected by cyberattacks in healthcare surpasses other breach types by 147% to 198% and exceeds their combined total by 93% (Cartwright, 2023; Jiang & Bai, 2019). Cyberattacks in healthcare that expose PII and PHI affect hundreds of millions of patients, costing an average of $7.13 million per attack, 84% higher than other industries (Cartwright, 2023).

1.4 Thesis Statement

The thesis statement of this praxis is as follows: *An LNN-based IDS is required to enhance the cyber resiliency of IoT-H devices.*

Leveraging LNNs in IDSs can enhance the detection and mitigation of data breaches and network disruptions in IoT-H networks due to their continuous learning capability and adaptive response refinement beyond the initial training phase. Their rapid processing, flexibility, and scalability in IoT-H network environments where data patterns are ever-changing and unpredictable can equip security analysts and response teams with a precise diagnosis that allows them to choose the appropriate tools to address the specific challenge at hand effectively.

1.5 Research Objectives

The primary objective of this praxis is to develop and evaluate Liquid NeurIoTic, a novel predictive intrusion detection architecture for IoT-H networks using LNNs. Liquid NeurIoTic is designed to identify and categorize cyberattacks on IoT-H networks. The main objective is divided into three goals:

1. converting network traffic features into cyclical waveforms to enable time-series representations for integration into an LNN model,
2. constructing an LNN model capable of detection and classification of IoT-H network traffic into malicious or benign categories, and
3. constructing an artificial neural network (ANN) and recurrent neural network (RNN) model capable of real-time classification of IoT-H network traffic into malicious or benign categories to benchmark the LNN's performance.

1.6 Research Questions and Hypotheses

This praxis' research questions and hypotheses investigate the potential of Liquid NeurIoTic as a tool for intrusion detection and classification in IoT-H networks. Specifically, this praxis seeks to identify the ideal LNN architecture and features for optimal performance. The goal is to utilize IoT network traffic data guided by the subsequent questions and hypotheses to craft an LNN model to detect cyberattacks, determine their assault mode, and adapt to new patterns after initial training.

Therefore, this praxis addresses the following research questions:

RQ1: Can IoT network traffic features be depicted as cyclical waveforms to enable time-series representation for integration into an LNN?

RQ2: Can an LNN be developed to detect and classify malware on IoT-H networks?

RQ3: Can an LNN outperform an ANN and RNN model in IoT-H malware classification?

Previous research has not investigated RQ1, RQ2, and RQ3. While several studies have partially examined RQ1 by converting network traffic features into waveforms, they have not considered the application of integrating them into LNN models.

Moreover, the three hypotheses formulated to investigate the research questions are as follows:

H1: Converting IoT network traffic features into cyclical waveforms and layering them for input enables time-series representation and enhances their utility.

H2: An LNN can be developed to detect and classify malware in IoT-H networks.

H3: An LNN outperforms ANN and RNN models in IoT-H malware classification.

1.7 Scope of Research

This research investigates the performance of LNNs in detecting and classifying IoT-H malware. Hence, this praxis uses the CICIoT2023 dataset to train and validate the LNN studied, which consists of more than 46,600,000 labeled records of benign and malicious IoT network traffic captured between 2018 and 2023. This research utilizes 3,017,817 records, or 6%, of the CICIoT2023 dataset, which is selected using stratified random sampling (SRS). Random undersampling (RUS) is also applied to evaluate performance using balanced and imbalanced datasets. The novel LNN is evaluated using loss, accuracy, precision, recall, F1-weighted, Matthews correlation coefficient (MCC), and Cohen's kappa.

1.8 Research Limitations

This praxis focuses on developing three neural network (NN) model types: ANNs, RNNs, and LNNs. The field of NN-based IDSs has seen remarkable breakthroughs. Nonetheless, no prior scholarly investigation has pertained to using LNNs in IDS systems for IoT-H networks. Therefore, this praxis centers its attention on and confines its scope to identifying and categorizing five cyber intrusions targeting IoT devices: 1) DDoS, 2) denial-of-service (DoS), 3) Mirai, 4) spoofing, and 5) reconnaissance (Recon).

1.9 Organization of Praxis

Chapter 1 serves as the introduction, outlining the domain under scrutiny, the problem statement, the thesis statement, the research's motivation, objectives, questions, hypotheses, scope, and limitations. Next, Chapter 2 delves into a comprehensive

literature review, capturing the cutting-edge developments pertinent to this praxis' research focus. It also highlights past studies relevant to the methodology applied in this research. Subsequently, Chapter 3 elaborates on the methodology adopted to tackle the issue of intrusion detection and classification within IoT-H networks. It encompasses aspects like the dataset, data preprocessing, feature selection, categorical data encoding, data standardization methods, converting network traffic data into cyclical waveforms, data balancing, and model training and evaluation.

Chapter 4 presents the results derived from Liquid NeurIoTic, the application of the LNN-based IDS methodology. This section concentrates on the viability of employing an LNN model for detecting and classifying cyber threats on IoT-H networks. It assesses the efficacy of the ANN, RNN, and LNN models, pinpoints the optimal features for the LNN model, and examines various performance indicators. Finally, Chapter 5 concludes the work by encapsulating the insights and takeaways from the praxis, underlining its contributions to the academic domain. Additionally, it hints at promising avenues worthy of future exploration.

Chapter 2—Literature Review

2.1 Introduction

This chapter is organized into seven sections exploring scholarly works to build a thorough understanding of the background and context of the problem addressed by this praxis. Section 2.1 provides the structure of Chapter 2, while Section 2.2 examines the proliferation of Internet of Things (IoT) devices and security weaknesses in IoT in Healthcare (IoT-H) devices. Section 2.3 delves into major cyberattacks targeting IoT devices and networks, particularly in healthcare, followed by Section 2.4, highlighting various intrusion detection systems (IDSs), their mechanisms, and the pros and cons inherent to each system. Section 2.5 explores neural network (NN)-based IDSs for IoT-H networks, while Section 2.6 explores the theoretical and practical applications of liquid neural networks (LNNs). Finally, Section 2.7 synthesizes the literature review and identifies a research gap that justifies the necessity for this praxis to create Liquid NeurIoTic, an LNN-based IDS for IoT-H networks.

2.2 Vulnerabilities in Internet of Things Devices in Healthcare

Meneghello et al. (2019) highlighted that the heterogeneity of IoT devices makes standardizing security designs for IoT-H networks challenging. Therefore, current healthcare protocols for managing digital medical records are susceptible to cybersecurity risks like malware (Namasudra et al., 2023). Moreover, Spathoulas et al. (2019) found that the rush to launch products within tight budgets often limited the adoption of more robust security measures due to higher costs. For example, security cameras lack patch update facilities and require full replacement if software is compromised (Bhushan et al., 2023). Al-kahtani et al. (2022) revealed that wireless networks in healthcare, such as IoT-

H networks, are also highly vulnerable to malware due to limited resources and their reliance on universal wireless access. These networks operate on the 2.4 GHz bandwidth, which is unlicensed, making it easier for attackers to disrupt services either temporarily or permanently (Al Kalaa et al., 2017).

Tweneboah-Koduah et al. (2017) demonstrated that the bi-directional wireless communication channel between IoT devices makes the system susceptible to network-based attacks and protocol failures. In other words, if these attacks succeed, they can infiltrate and disrupt operations and access crucial data. For example, most user-interactive IoT applications are designed for web and mobile platforms that primarily utilize application programming interfaces (APIs) as their core architectural interface (Tweneboah-Koduah et al., 2017).

Tweneboah-Koduah et al. (2017) warned that a non-updated API can be vulnerable to a range of security breaches that can jeopardize an entire network. This vulnerability makes IoT-H devices prime targets for cyberattacks aiming to breach IoT-H networks. For instance, in CVE-2016-7413, a specific flaw in the wddx_stack_destroy function in PHP versions prior to 5.6.26 and 7.x before 7.0.11 can be exploited by attackers to disrupt services using a malformed wddxPacket XML document (Tweneboah-Koduah et al., 2017). Additionally, Tweneboah-Koduah et al. (2017) uncovered how CVE-2016-4539 revealed a vulnerability in PHP's xml_parse_into_struct function in versions before 5.5.35, 5.6.x, 5.6.21, 7.x, and 7.0.6, where attackers can disrupt services or cause other unknown damages using manipulated XML data. As such, it is unsurprising that Yu (2023) found that a staggering 59% of IP cameras are vulnerable due to unpatched common vulnerabilities and exposures (CVEs), with 56% of

these posing critical risks. Additionally, they uncovered that 37% of printers harbor unpatched CVEs, with 30% being critically severe, while Voice over Internet Protocol (VoIP) devices, which are less critical with only 2% severity, show security gaps with 53% having unpatched CVEs (Yu, 2023).

Overall, the authors in this section have highlighted that IoT devices in healthcare present significant cybersecurity risks from the lack of standardized security designs and the tradeoff between cost-effectiveness and security. An initial breach in one IoT device in a healthcare setting can act as a gateway to the entire healthcare network, putting the whole connected IoT-H network at risk. Therefore, IoT infrastructures in healthcare must sustain a secure and robust operational state to counteract these vulnerabilities.

2.3 Prevalent Cybersecurity Threats in the Internet of Things in Healthcare Networks

Denial-of-service (DoS), distributed denial-of-service (DDoS), Mirai, spoofing, and reconnaissance (Recon) attacks are major attacks in IoT networks, as revealed by Makhdoom et al. (2019). IoT service gateways have become primary targets for intrusions, and IoT-H networks are comprised of similar devices used in IoT networks. Hence, these attacks can become predominant threats within IoT-H networks, as discussed in this praxis.

2.3.1 Denial-of-Service and Distributed Denial-of-Service Attacks

Kumari and Mrunalini (2022) discovered that DoS and DDoS attacks are the most common and challenging to mitigate because threat actors use fake return addresses to hide their identities, which makes the attack difficult to trace and prevent. Both attacks bombard a device with increasing requests that become virtually impossible to manage,

resulting in a DoS of the targeted system (Kumari & Mrunalini, 2022). Ibrahim et al. (2022) highlighted that the differentiating factor between a DoS and a DDoS attack is that the former uses a single source to execute an attack, while the latter uses multiple sources. However, DDoS attacks are more attractive to threat actors because they can cause more damage but do not require advanced technical skills, resources, and money to execute (Adedeji et al., 2023). Therefore, IoT-H devices frequently become tools for orchestrating DoS and DDoS attacks due to their weak security (Huraj et al., 2020; Ul et al., 2018).

A major type of DoS attack is the Smurf attack, which has become highly concerning because of its widespread and highly scalable nature (Tripathi et al., 2013). A Smurf attack overwhelms a target server with Internet Control Message Protocol (ICMP) requests by exploiting a forged internet protocol (IP) address (Tripathi et al., 2013). Tripathi et al. (2013) demonstrated that the attacker crafts a fake packet with the target server's IP address as its source and dispatches it to a router's IP broadcast address. The packet prompts the router to broadcast the request to all hosts within the network, which then unwittingly sends ICMP responses back to the targeted server, flooding it with traffic (Tripathi et al., 2013).

Similarly, the primary goal of a DDoS attack is to render a service unavailable to its intended users (Mohapatro & Snigdh, 2021). Riskhan et al. (2023) presented evidence that the synchronize (SYN) flood technique is the most aggressive form of DDoS attacks, comprising 81%. A SYN flood attack targets the Transmission Control Protocol (TCP) handshake by sending numerous SYN packets to a device but does not complete the handshake with the final acknowledge (ACK) packet (Guo & Gao, 2022). This process

creates multiple half-open connections, which overwhelms a device or entire network and causes a DoS (Guo & Gao, 2022).

Mohapatro and Snigdh (2021) highlighted that the SYN flood DDoS attack has become highly effective in causing major interruptions in large corporations with advanced cyber defenses, such as Amazon, Netflix, and Twitter (currently known as X). Moreover, Wang et al. (2018) revealed that the latest DDoS attacks have become even more dynamic as attackers can continuously alter their attack vectors to persistently navigate through various defense strategies until they successfully breach network defenses. For example, the traffic flow of a DDoS attack closely resembles that of typical traffic surges, making the detection increasingly challenging (Wang et al., 2023). As a result, the attack inevitably saturates the network bandwidth or depletes system resources, which leads to slowed response times or a complete DoS (Mohapatro & Snigdh, 2021).

2.3.2 Mirai Attack

Shao et al. (2021) revealed that the Mirai attack in IoT-H networks is the most threatening. The Mirai attack has become so destructive that it has disrupted nearly half the internet due to insufficient security practices in hundreds of thousands of IoT devices (Kolias et al., 2017). Mirai was first discovered in 2016 as malware that scans the internet for IoT devices powered by Argonaut reduced instruction set computer core (ARC) processors (Victor et al., 2023). Metongnon and Sadre (2018) demonstrated that Mirai exploits vulnerabilities in devices that retain default factory settings, such as cameras and routers, and gains access through telnet ports 23 and 2323 by employing dictionary attacks. Once accessed, it infects them and turns the devices into remotely controlled bots (Kolias et al., 2017; Metongnon & Sadre, 2018). As a result, each infected IoT device

becomes part of the Mirai botnet and is controlled by an attacker to overwhelm servers and websites with traffic (Kolias et al., 2017).

Mirai attacks can be considered a type of DDoS attack. However, the key difference is that the device owner is unaware that the device is compromised and controlled to execute the attack (Zhang et al., 2020). Metongnon and Sadre (2018) showed that the original version of Mirai utilized stateless TCP SYN packets in its attacks, with the initial sequence number mirroring the IP address of the targeted device. However, Wazzan et al. (2021) uncovered that the Mirai code has evolved since its inception, with attackers regularly updating it to exploit new vulnerabilities, target a broader range of devices, and avoid detection. Furthermore, Tushir et al. (2021) revealed that IoT devices in healthcare are mostly susceptible to Mirai attacks since it is common for IoT-H device owners to neglect to change default credentials and for IoT-H devices not to regularly receive necessary software or security updates. Tushir et al. further detailed how the inherent resource constraints of many IoT-H devices also make implementing standard security protocols to prevent Mirai attacks a substantial challenge.

2.3.3 Spoofing Attack

Another highly threatening attack in IoT-H networks is the spoofing attack. Khan et al. (2022) explained that a spoofing attack occurs when an attacker pretends to be a trusted device or user to access secure information, steal data, or spread harmful software. Tariq et al. (2023) highlighted common techniques in spoofing attacks, including faking DNS, web traffic, IoT cloud services, and network nodes. However, Khan et al. (2022) and Tariq et al. (2023) both revealed that the most frequent attacks involve spoofing media access control (MAC) addresses or IP addresses.

MAC address spoofing is when an attacker changes their device's MAC address to match a legitimate device in a wireless network (Alotaibi & Elleithy, 2016). For example, an attacker may mimic the MAC address of a smart camera in an IoT-H network to intercept data or launch a man-in-the-middle (MitM) attack on a smart medical device (Alotaibi & Elleithy, 2016; Malasri & Lan Wang, 2009). This attack can also overwhelm the network with fake requests from random MAC addresses and drain network resources in a tactic known as resource depletion (Alotaibi & Elleithy, 2016; Malasri & Lan Wang, 2009).

Ramesh et al. (2010) defined IP spoofing as a hacking method where attackers gain unauthorized access to a computer by disguising their messages as coming from a trusted source. Ramesh et al. explained that in this attack, the attacker chooses a target and acquires the IP address of a trusted computer to disrupt the communication of the trusted computer using a tactic like SYN flooding. The attacker observes the interaction between the target and the trusted systems to estimate the sequence numbers in the trusted computer's communication (Ramesh et al., 2010). Subsequently, the attacker alters the message headers to mimic those from the trusted computer and attempts to connect to a secure service or port (Ramesh et al., 2010). If successful, the attacker establishes a hidden backdoor for future access (Ramesh et al., 2010).

2.3.4 Reconnaissance Attack

Zhang et al. (2023) highlighted that Recon attacks are the most common first step in a cyberattack. Approximately 70% of cyberattacks are initiated following Recon scanning activities. A Recon attack aims to look for weaknesses and other important

details about a target's system or network without alerting the target to formulate possible attack methods (Alani & Damiani, 2023; Mazurczyk & Caviglione, 2021).

Neto et al. (2023) highlighted that the two major types of Recon attacks are host discovery and operating system (OS) fingerprinting. In host discovery, an adversary checks if a host is active by probing its IP address (Neto et al., 2023). For example, an adversary begins with the target network's range of IP addresses, uses a ping sweep technique to send packets to each IP address, and waits for responses to identify active hosts (Belalis et al., 2023).

In comparison, OS fingerprinting involves examining data packets from a network to determine the OS and configuration details to help hackers identify and exploit vulnerabilities more easily (Laštovička et al., 2023). OS fingerprinting can be active and passive (Auffret, 2010). In active OS fingerprinting, attackers send a packet to a target and then scrutinize the TCP packet response to identify the OS (Auffret, 2010). Conversely, attackers observe network traffic in passive OS fingerprinting but do not directly interact with it (Auffret, 2010). Shamsi et al. (2016) highlighted that passive OS fingerprinting is more effective than active OS fingerprinting because it is more discreet but much slower.

2.4 Categories of Intrusion Detection Systems

A common solution against the evolving nature of cyberattacks is using an IDS. Ahmad et al. (2021) defined an IDS as a security tool to monitor network or system activities to identify and report malicious actions or policy breaches. The two primary types of IDSs are network-based (NIDSs) and host-based (HIDSs; Ahmad et al., 2021). Vanin et al. (2022) highlighted that NIDSs offer efficient and non-intrusive monitoring

by providing a centralized overview of the network segment in which they are placed. They analyze packet contents and metadata to identify threats, offering the advantage of fewer installations and maintenance points compared to HIDSs, which must be installed on each host (Ahmad et al., 2021; Vanin et al., 2022). However, Han and Pak (2023) warned that NIDSs can impact overall network performance, requiring significant processing power to analyze all passing traffic. Furthermore, Yaacoub et al. (2020) identified that the diversity of IoT devices and protocols in healthcare networks makes it challenging for NIDSs to be configured to accurately monitor all types of traffic without producing many false positives.

Conversely, HIDSs are installed directly on endpoints, like IoT devices or servers, in the healthcare environment to monitor system logs, file access and modifications, and other activities to detect potential intrusions on the host device (Vanin et al., 2022). HIDSs are more versatile for various network setups because they are not dependent on the network architecture (Singh & Singh, 2014). Since HIDSs monitor system internals for precise attacks on individual devices, they are well-suited to catch threats that have bypassed network defenses, including insider threats (Yaacoub et al., 2020).

However, Lu et al. (2022) explained that implementing HIDSs in environments with many IoT devices, like healthcare networks, involves a significant undertaking to install and maintain them on each device. Furthermore, Park et al. (2021) emphasized that HIDSs are limited to monitoring only the device on which they are installed, so they overlook wider network threats that NIDS would detect. Thus, the host device's performance is degraded, which is a critical issue for IoT-H devices that necessitate real-time operation (Park et al., 2021).

IDSs have two main detection methods: signature-based (SIDSs) and anomaly-based (AIDSs; Ahmad et al., 2021). Ahmad et al. (2021) described that SIDSs use known patterns of misbehavior to detect intrusions by using if-else rules to match pre-defined signatures in network traffic or host events. This method is effective against known attacks and tends to have a lower rate of false positives when detecting known threats since it relies on established patterns. However, Ahmad et al. (2021) warned that these signatures must be regularly updated to cover new attacks and remove inefficient ones. Thus, signature databases require regular updates to remain effective against new threats because they cannot detect new, unknown threats or sophisticated attacks that do not match existing signatures (Ahmad et al., 2021). As a result, SIDSs are essentially ineffective against zero-day attacks because there are no known signatures for an attack that has not happened yet (Ahmad et al., 2021).

In contrast, Ahmad et al. (2021) highlighted that AIDSs can detect zero-day attacks by utilizing a baseline profile of normal activity to analyze network data and traffic to flag deviations as potential threats. Rathee et al. (2023) highlighted that AIDSs are useful in an IoT-H environment because new devices are added to the network daily. However, rather than being able to specify the type of attack occurring, AIDSs can only identify deviations from normal patterns (Maseer et al., 2021). Ahmad et al. (2021) explained that this limitation can lead to many false positives, resulting in alert fatigue, since unusual but harmless network behavior can be mistakenly identified as anomalies.

Overall, current scholarly works have indicated that the optimal security posture for IoT-H networks is not strictly a choice between NIDSs versus HIDSs or SIDSs versus AIDSs. Instead, optimal security strategically integrates all these systems and includes

new methods since each approach has advantages and limitations. For example, NIDSs offer extensive network coverage and centralized monitoring, making them ideal for guarding against external threats across multiple devices, while HIDSs provide in-depth host-level security essential for detecting insider threats and ensuring the integrity of vulnerable healthcare software on individual devices. Furthermore, SIDSs excel in identifying and mitigating known threats efficiently, while AIDSs offer the adaptability crucial for spotting novel and sophisticated attacks in the healthcare sector where stakes can be life-threatening. However, Hnamte and Hussain (2023) stated that despite the effectiveness of strategically integrating all of these systems, they still struggle to match the relentless and constantly evolving methods of cyberattacks.

2.5 Neural Network-Based Intrusion Detection Systems for Internet of Things in Healthcare Networks

NNs have revolutionized almost every field with cutting-edge solutions in areas like robotics, reinforcement learning, computer vision, and healthcare (Bonardi et al., 2020; Esteva et al., 2017; Jaderberg et al., 2019; Neftci & Averbeck, 2019; Wei, 2019). Researchers have presented NN-based IDSs as a powerful solution in response to the constantly evolving attack techniques. Chu et al. (2020) highlighted that NNs effectively detect a broad spectrum of cyber threats because they can autonomously learn and identify features via non-linear processing capabilities. As a result, the reliance on domain expertise and the quality of pre-defined features for effective detection is lessened (Chu et al., 2020). Therefore, as IoT-H networks continually expand with the addition of new devices, the adaptability offered by NN-based IDSs has become increasingly attractive (Maseer et al., 2021).

To enhance the detection and mitigation of cyber threats in Internet of Health Things (IoHT) devices, Vijayakumar et al. (2023) introduced a deep neural network (DNN) model. They created a system with a multi-class classification framework to identify four unique cyberattacks in healthcare using the ECU-IoHT dataset: 1) ARP spoofing, 2) DoS, 3) Nmap, and 4) Smurf (Vijayakumar et al., 2023). After being trained on over 500 epochs, their DNN model achieved an accuracy of more than 99%, a precision of 99.3%, a recall of 96.8%, and an F1 score of 90.3% (Vijayakumar et al., 2023). The authors stated that future work should evaluate the effectiveness of the proposed system within a real-time IoHT setting to broaden the system's scalability to detect other cyberattacks in IoHT devices (Vijayakumar et al., 2023).

Ravi et al. (2023a) developed an attention-focused deep learning (DL) methodology that leveraged cross-architecture analysis to detect and categorize malware in healthcare IoT devices. They utilized byte sequences from executable and linkable format (ELF) files and converted them into numeric values through an embedding layer (Ravi et al., 2023a). This embedding layer, coupled with a series of DL layers, integrated attention mechanisms to highlight key byte sequence features essential for precise malware identification and classification (Ravi et al., 2023a). Beyond these capabilities, this methodology also effectively discerned the CPU architecture of the ELF file (Ravi et al., 2023a). Their model achieved 94% accuracy in IoT-H malware detection and 95% in classifying IoT-H malware and CPU architecture (Ravi et al., 2023a).

To automatically detect and classify attacks for IoT-H IDSs, Ravi et al. (2023b) combined network flow features and patient biometrics in the preprocessing stage to feed into a DL-based model they tailored for this data. Additionally, the authors integrated an

attention mechanism within the DL layers to enhance the model's capability to extract optimal features for accurate attack detection (Ravi et al., 2023b). To address the challenge of data imbalance commonly encountered in real-world datasets, Ravi et al. (2023b) employed a cost-sensitive learning strategy during the model's training phase by assigning varying costs to different classes, which led to improved performance outcomes. Their model architecture achieved a 10-fold cross-validation accuracy of 95% when analyzing network features, 89% with patient biometrics, and 99% when both sets of features were combined (Ravi et al., 2023b).

Moreover, Wozniak et al. (2021) developed an RNN-Long Short-Term Memory (RNN-LSTM) model to analyze network traffic in various IoT environments. They implemented this model at the firewall level to process real-time connection data, classify it, and communicate its decision back to the firewall, achieving 99.9957% accuracy (Wozniak et al., 2021). Perhaps most notably, their model did not require extensive computing power and energy post-training, with data processing times in the nanosecond range, which allowed it to run unobtrusively in the background (Wozniak et al., 2021). To improve the system's efficiency and effectiveness, the authors stated they would focus on implementing preprocessing methods involving selective feature reduction to concentrate on the relationships between different features (Wozniak et al., 2021). Wozniak et al. (2021) aimed to discover how to enhance the NN to function more autonomously, regardless of the size and specifics of the training dataset.

To address the challenge posed by the limited computing power and memory of IoMT devices, Chaganti et al. (2022) developed a particle swarm optimization DNN (PSO-DNN) model. Instead of focusing solely on model development, the authors aimed

to refine the feature selection process as a new method to enhance an NN model's effectiveness in attack detection (Chaganti et al., 2022). First, they merged network traffic data with patient sensing information to develop a more comprehensive and robust dataset (Chaganti et al., 2022). Then, to identify and select key features relevant to predicting attacks in IoMT environments, the authors utilized the PSO technique to select eight optimal features from their combined network and patient sensor features (Chaganti et al., 2022). From this, they experimented with six machine learning (ML) models: logistic regression (LR), K-nearest neighbors (KNN), decision tree (DT), AdaBoost, random forest (RF), support vector machine (SVM), and three NN models: DNN, convolutional neural network (CNN), and LSTM (Chaganti et al., 2022). Their DNN model achieved the best performance in detecting network intrusions with 96% accuracy (Chaganti et al., 2022).

In their future work, Chaganti et al. (2022) plan to broaden their IoMT intrusion detection research using ML and DL models to include more attacks like DoS, data injection, and man-in-the-middle (MitM) attacks. They also aim to develop ML operations (MLOps) in alignment with healthcare industry regulations, particularly for real-time data processing of combined patient and network data (Chaganti et al., 2022). Another focus sought to ensure patient data privacy within IoMT networks through data anonymization techniques while studying the correlation between biometric features and attacks (Chaganti et al., 2022).

To address the vulnerability of cloud-stored e-healthcare information to cybersecurity malware, Anand et al. (2021) developed a CNN-based model to detect malware attacks (CNN-DMA). Hence, all malware were transformed into greyscale

images to prepare the data for input into the CNN model (Anand et al., 2021). Anand et al. (2021) converted the malware binary data into 8-bit vectors. They then processed these vectors further to create corresponding greyscale images, which effectively translated the complex malware data into a format suitable for CNN analysis (Anand et al., 2021).

For the initial convolutional layer, their model processed an input image of 32 × 32 × 1 pixels and then fed it through a dense layer, dropout layer, and flatten layer while utilizing batch sizes of 64 over 20 epochs (Anand et al., 2021). In testing its efficacy, the model was applied to the Malimg dataset, which comprised 25 different malware families (Anand et al., 2021). Their model demonstrated a high recall rate of 98.9% for identifying Adialer.c and a precision rate of 95.8% in detecting Agent.FYI (Anand et al., 2021). However, its performance significantly declined when detecting Allaple.A, Allaple.L, and Alueron.gen!J, where it only achieved precision and recall rates between 79% and 92% (Anand et al., 2021). The authors advocated for further research to enhance the accuracy of various DL methods across diverse datasets.

In response to the increased network traffic within IoT healthcare systems that inhibits NIDSs' and AIDSs' adaptability and effectiveness against new attack methods, Akshay Kumaar et al. (2022) introduced a hybrid DL framework called ImmuneNet to potentially keep up with the latest intrusion attacks. ImmuneNet employs a multifaceted approach that includes feature engineering, oversampling for better class balance, and hyperparameter optimization (Akshay Kumaar et al., 2022). ImmuneNet has linear layers at both ends, each with 512 neurons, and a larger hidden layer of 784 neurons, which Akshay Kumaar et al. (2022) selected to match the feature range of the datasets. The

authors initially selected the neuron counts arbitrarily. However, eventually, they chose equal neuron counts in the first and last layers for the residual connection by fine-tuning through grid search (Akshay Kumaar et al., 2022). Overall, the ImmuneNet architecture comprised less than 1 million parameters, which the authors claimed made it practical for integration into medical devices and IoT-H systems (Akshay Kumaar et al., 2022).

To validate ImmuneNet's efficacy, Akshay Kumaar et al. (2022) developed LR, DT, RF, and XGBoost models as baseline models to compare ImmuneNet's performance. All models used datasets from the 2017 and 2018 Canadian Institute for Cybersecurity's Intrusion Detection System (CICIDS) datasets and the Bell Domain Name System (DNS) 2021 dataset, which cover a wide range of real-time and up-to-date cyberattack data (Akshay Kumaar et al., 2022). ImmuneNet achieved the best performance on the CIC Bell DNS 2021 dataset, achieving approximately 99.19% in accuracy, 99.22% in precision, 99.19% in recall, and a 99.2% Area Under the Receiver Operating Characteristic (AUC-ROC) curve (Akshay Kumaar et al., 2022).

For future work, the authors sought to eliminate manual retraining or fine-tuning by developing an NN-based model to independently adjust to new attacks (Akshay Kumaar et al., 2022). They proposed creating a self-supervised learning-based NN to address this challenge and plan to utilize more current datasets with a better class balance (Akshay Kumaar et al., 2022). Additionally, they planned to implement adversarial reweighting strategies to improve the model's fairness (Akshay Kumaar et al., 2022).

To classify cyberattacks within IoMT networks, Khan et al. (2023) presented a blend of ensemble learning and fog-cloud architecture. Their architecture utilized a series of LSTM networks as initial learners, followed by a DT that classified events into attacks

or normal activities. This blended method outperformed precision by 4% when compared against HealthGuard by Newaz et al. (2019), Support Vector Machine IDS by Begli et al. (2019), Stacked Autoencoder IDS by He et al. (2019), and Ensemble-ML by Kumar et al. (2021; see Khan et al., 2023).

2.6 Liquid Neural Networks

This section comprises four subsections to provide a comprehensive background, including different types of LNNs and the advancements made by the research community in this domain. This praxis refers to an LNN as any NN fundamentally based on the liquid time-constant network (LTC), such as those created by Hasani et al. (2018), Hasani et al. (2021), Lechner et al. (2020), and Hasani et al. (2022). Section 2.6.1 provides a foundational background of LNNs, Section 2.6.2 demonstrates the practical applications of LTCs, Section 2.6.3 covers a more efficient LNN called closed-form continuous-depth (CfC) networks, and Section 2.6.4 dives into a compact wiring of LTCs called neural circuit policies (NCPs).

2.6.1 Foundations of Liquid Neural Networks

The concept of LNNs was first introduced by Hasani et al. in 2018, who demonstrated their status as universal approximators. A universal approximator is a model or system, or in this case, an NN, that can approximate any function, no matter how complex, given certain conditions (Moon, 2021). Hence, an LNN can adapt to and mimic the behavior of any dataset to make predictions, regardless of its complexity, such as predicting continuous or discrete values. The term *liquid* characterizes the network's flexibility to adapt its perception of time (Hasani et al., 2018).

For example, in simpler models, like an RNN, time is processed in fixed steps, like days or seconds (Rezk et al., 2020). However, in an LNN, the time intervals change based on the system's current state as the time-constant is coupled with the hidden state, as Hasani et al. (2018) demonstrated, which allows the model to be extremely flexible or liquid to provide an optimized ability to predict or anticipate unpredictable changes in data (Hasani et al., 2018). This foundational work by Hasani et al. was essential for LNNs because it proved they could theoretically represent any dynamical system. Hasani et al.'s work highlighted the potential superiority of LNNs in managing unpredictable, time-series tasks that were challenging for traditional temporal-based models like RNNs to handle.

2.6.2 Liquid Time-Constant Networks

In 2021, Hasani et al. (2021) validated the theoretical scope of LNNs by developing LTCs for practical applications. LTCs' architecture resembles linear first-order dynamical systems (Hasani et al., 2021), so the NN's neurons are linear and connected through non-linear gates (Hasani et al., 2021). For example, traditional feedforward networks (i.e., RNNs) use layers of neurons and connections that perform non-linear calculations to learn the patterns and dynamics of data or how things change over time (Wang et al., 2019).

Hasani et al. (2021) demonstrated that the approach with LTCs is different. Instead of using typical non-linear functions at each neuron to calculate its outputs, LTCs use a linear model such as first-order ordinary differential equations (ODEs), which are simpler to calculate (Hasani et al., 2021). These are the linear first-order dynamical systems. Then, LTCs control and adjust these linear models using non-linear functions or

interlinked gates (Hasani et al., 2021). This architecture is like having the actions of several simple machines coordinated by a more complex control system. Thus, contrary to traditional NNs, which sum input values and apply an activation function at each neuron, LTCs process input data at each neuron through an ODE and interlink these neurons with a similar non-linear activation function used by traditional NNs, such as a hyperbolic tangent—this allows for a more liquid model to capture the most intricate nuances of the data (Hasani et al., 2021).

For reference, differential equations (DEs) mathematically describe how things change over time and are commonly used in physics and engineering. Specifically, ODEs are simply DEs that describe how things change over time for one independent variable (Miao et al., 2014). Hasani et al. (2021) designed LTCs to use ODE solvers to compute ODEs. Thus, solving the ODE provides the LNN with the function to learn how the data changes over time (Hasani et al., 2021).

ODE solvers in LTCs have two types: variable step, which adjusts based on the data, and fixed step, which uses a constant step size (Hasani et al., 2021). Hasani et al. (2021) identified that LTCs' performance highly depends on the selected solver. Additionally, to learn from the time-series data effectively, LTCs use backpropagation through time (BPTT; Hasani et al., 2021). BPTT is a training method used in sequential models, like RNNs, where the network is unrolled through time, and the error is propagated back through each step (Bellec et al., 2020).

Hasani et al. (2021) demonstrated that LTCs outperform traditional and advanced RNNs in time-series prediction tasks with a 5% to 70% performance improvement in tasks like hand gesture segmentation, room occupancy detection, human activity

recognition, and computer vision classification. Additionally, when performing identical tasks, Hasani et al. (2021) revealed that LTCs are more lightweight since they require fewer parameters than advanced RNN models like continuous-time RNN (CT-RNN), ODE-RNNs, LSTM, and CT-gated recurrent unit (CT-GRU) networks.

Several separate studies have demonstrated the efficacy of LTCs. For example, Yin et al. (2021) achieved a 10% to 40% performance improvement over RNN and auto-regressive models when using LTCs to predict continuously changing channel state information (CSI) in multiple-input multiple-output (MIMO) communication systems (Yin et al., 2021). Moreover, Shang and Hu (2022) achieved superior performance with LTCs over CNN and LSTM models when calibrating red-blue-green (RBG) cameras and 3D light detection and ranging (LiDAR) sensors in real time.

Quan et al. (2022) outperformed advanced RNNs, such as neural ODE, LSTM, CT-RNN, and CT-GRU models, when classifying handwashing videos using LTCs (Quan et al., 2022). Specifically, they constructed a random forest (RF) model using multiple LTCs, with each LTC network corresponding to a tree to classify the quality of handwashing. The authors utilized a publicly available handwashing dataset from the World Health Organization for recognizing individual handwashing steps and a custom dataset from a hospital in Vietnam with the full handwashing process (Quan et al., 2022).

Quan et al.'s (2022) model achieved an overall accuracy of 98.9% in recognizing individual handwashing steps. Recognizing each handwashing step achieved accuracy between 97.32% and 99.61%. For the full handwashing process, it recognized 78% (Quan et al., 2022). The authors emphasized that future research should focus on discovering

more efficient ODE solvers to minimize memory usage and computing durations (Quan et al., 2022).

Nielsen et al. (2023) demonstrated that LTCs can accurately foresee the occurrence and duration of time-series data. Hence, they can adapt to data changes without needing specific training data for each scenario. LTCs achieved over 97.85% accuracy in predictions in time-series data when predicting blockages in millimeter wave (mmWave) links (Nielsen et al., 2023).

Gajjar et al. (2023) evaluated the effectiveness of LNNs in forecasting highly unpredictable data by utilizing LTC networks to analyze and predict stock prices of Adani Ports, Reliance, and Tata Steel on the National Stock Exchange of India. Their experiments revealed that LTCs consistently outperformed RNN, LSTM, and GRU models (Gajjar et al., 2023). When Gajjar et al. compared LTCs against the best-performing temporal models, they achieved an average of 427.41% better mean squared error (MSE) scores, 128.72% better root mean squared error (RMSE) scores, and 22.80% better R^2 scores (Gajjar et al., 2023). However, despite these results, Gajjar et al. found that LTCs were more than 15 times slower during training and more than eight times slower during testing compared to advanced RNN models, revealing a significant area of concern for scalability (Gajjar et al., 2023). As a result, they emphasized that future work should involve improving the speed of LTCs.

2.6.3 Closed-Form Continuous-Depth Networks

Given that LTC performance is highly dependent on the choice of ODE solvers, Hasani et al. (2021) revealed that although advanced ODE solvers improved LTCs' effectiveness, the solvers ultimately slowed their performance (Hasani et al., 2021). Addressing this resource constraint of LTCs, Hasani et al. (2022) created closed-form continuous depth (CfC) networks. CfC networks are essentially LTCs that do not require numerical ODE solvers, so they save time and increase performance (Hasani et al., 2022). CfC networks achieve these improvements by calculating a tightly bounded approximation of a crucial integral within LTC dynamics, leading to a closed-form solution (Hasani et al., 2022), which is a way to solve a mathematical problem with an exact answer expressed as a simple formula (Borwein & Crandall, 2013). Closed-form solutions offer straightforward equations that give simple, exact answers each time, regardless of the input data.

Hasani et al. (2022) used the closed-form formula to directly find the solution to the ODEs, which eliminated the need to use the computationally expensive ODE solver. They achieved nearly 90% faster data processing than with LTCs. Hasani et al. validated their work by classifying bit streams using the bit-stream exclusive-or (XOR) dataset, performing event-based sequential image processing using the MNIST dataset, and predicting the mortality of 8,000 patients admitted to an intensive care unit using the PhysioNet Challenge 2012 dataset (Hasani et al., 2022). Their experiments achieved higher accuracies with lower training times per epoch with CfC networks compared to advanced RNN models.

Furthermore, Qiu et al. (2023) demonstrated the increased performance of CfCs while using them with quantile regression (QR) to forecast wind speed. When benchmarked against QR-time convolution network (QR-TCN), QR-LSTM, QR-GRU, and QR-NN models, the QR-CfC model achieved a 21.89% reduction in the mean absolute percentage error (Qiu et al., 2023). Additionally, Qiu et al. experienced a decrease in the continuous ranked probability score by 11.14% using the CfC model.

2.6.4 Neural Circuit Policies

To make LNNs more compact and interpretable, Lechner et al. (2020) developed NCPs with a four-layer sparse neural architecture in which the overall architecture mimics the structure found in the Caenorhabditis elegans (*C. elegans*) nematode's nervous system with its neurons defined by LTCs (Gleeson et al., 2018; Lechner et al., 2019; Lechner et al., 2020). A key advantage of this architecture is that it requires fewer neurons and learning parameters than other DL-based models, including CT-RNNs, LSTMs, GRUs, and CNNs (Lechner et al., 2020). Even though the NCP network's size is significantly smaller than advanced RNN models, it continually outperforms them.

For example, Lechner et al. (2020) developed state-of-the-art NCP, CT-RNN, LSTM, and CNN models with the same convolutional head and trained and evaluated them in identical conditions in an end-to-end autonomous driving task (Lechner et al., 2020). Both the LSTM and CT-RNN had 64 neurons, while the NCP had only 19 neurons. The NCP model was 63 times smaller than a state-of-the-art CNN network, while the NCP's control network was 970 times sparser than the LSTM network and 241 times sparser than the CT-RNN network (Lechner et al., 2020). Additionally, the

trainable parameter space of an NCP's RNN compartment was 233 times smaller than the LSTM network and 59 times smaller than the CT-RNN (Lechner et al., 2020).

Several studies have highlighted the efficacy of NCPs. For instance, Tylkin et al. (2022) demonstrated the scalable superiority of NCP models by training a 200 KB-sized NCP to control a DJI Tello drone and autonomously avoid obstacles. The NCP was so compact and efficient that Tylkin et al. (2022) deployed it onto a Raspberry Pi Zero, which weighed less than 10 grams (Tylkin et al., 2022).

Loy-Benitez et al. (2022) developed an NCP network to model particulate matter 2.5 (PM2.5) concentration in subway environments using indoor and outdoor air variables to accurately detect PM2.5 levels that were unhealthy for sensitive groups. Compared with DNN, RNN, LSTM, GRU, and residual network models, the NCP network improved accuracy by 27–30% (Loy-Benitez et al., 2022). To practically validate its performance, Loy-Benitez et al. deployed the NCP network in a real-world ventilation control system of a subway station. Hence, they found that the NCP network reduced unhealthy PM2.5 levels to moderate levels, which improved air quality management and minimized energy use for ventilation. Their work also revealed that the NCP network experienced the lowest increase in energy consumption among the tested methods (Loy-Benitez et al., 2022).

Razzaq and Hongwei (2023) revealed that the NCP's architecture enhanced computational efficiency and scalability by having a significantly lower number of trainable parameters. Razzaq and Hongwei developed an NCP model for autonomous driving tasks, demonstrating that the NCP architecture was remarkably resilient against noise interference (Razzaq & Hongwei, 2023). Hence, the NCP was unaffected by

perturbations caused by different weather and visual conditions, such as glare, direct sunlight, and shadows (Razzaq & Hongwei, 2023). Razzaq and Hongwei closely observed the activities of each neuron of the NCP during inference. They recorded that despite variations in activations at other neuron levels, the NCP model could filter motor neuron-level noise for more accurate outcomes (Razzaq & Hongwei, 2023). Finally, Chahine et al. (2023) conducted a series of autonomous flying experiments in scenes with a series of untrained models; the NCP model achieved approximately 20% better performance than state-of-the-art LSTM, GRU, and ODE-RNN models (Chahine et al., 2023).

2.7 Summary and Conclusion

This literature review highlighted the advancements made by the research community in creating a range of IDS solutions to protect against malware in IoT-H networks. Researchers have explored various NN-based IDSs to adapt to the evolving challenges of IoT networks in healthcare, such as limited computational resources and diverse device ecosystems. Despite notable progress in the field, current solutions concerning NNs in IoT-H networks are flawed by issues regarding scalability and IoT device resource management. The resource-heavy nature of many NN models renders them impractical for deployment on smaller, computationally limited IoT devices. Furthermore, the evaluation of NN models in numerous studies has been problematic because of the overreliance on inadequate metrics like accuracy, which has distorted the representation of their true effectiveness.

This chapter included a comprehensive review of the existing literature on LNNs to highlight their effectiveness as a promising alternative to conventional NN models in

detecting intrusions in IoT-H networks. The speed, adaptability, and scalability of LNNs make them an attractive choice for classification tasks where data patterns are inconsistent and unpredictable and in scenarios where NN-based IDSs struggle to keep up with constantly evolving attack vectors. No study has used LNNs for cybersecurity tasks like intrusion detection. Therefore, an LNN-based IDS is required to enhance the cyber resiliency of IoT-H devices.

Chapter 3—Methodology

3.1 Introduction

This chapter outlines the procedures, tools, and methodologies utilized in the development of Liquid NeurIoTic, a liquid neural network (LNN)-based intrusion detection system (IDS) for the Internet of Things in Healthcare (IoT-H). It includes an in-depth examination of Liquid NeurIoTic's architecture: data collection and integration procedures; feature importance selection process; data sampling techniques; data standardization; encoding and formatting techniques; training, validation, and testing techniques; and a performance evaluation process. Liquid NeurIoTic's architecture is illustrated in Figure 3-1.

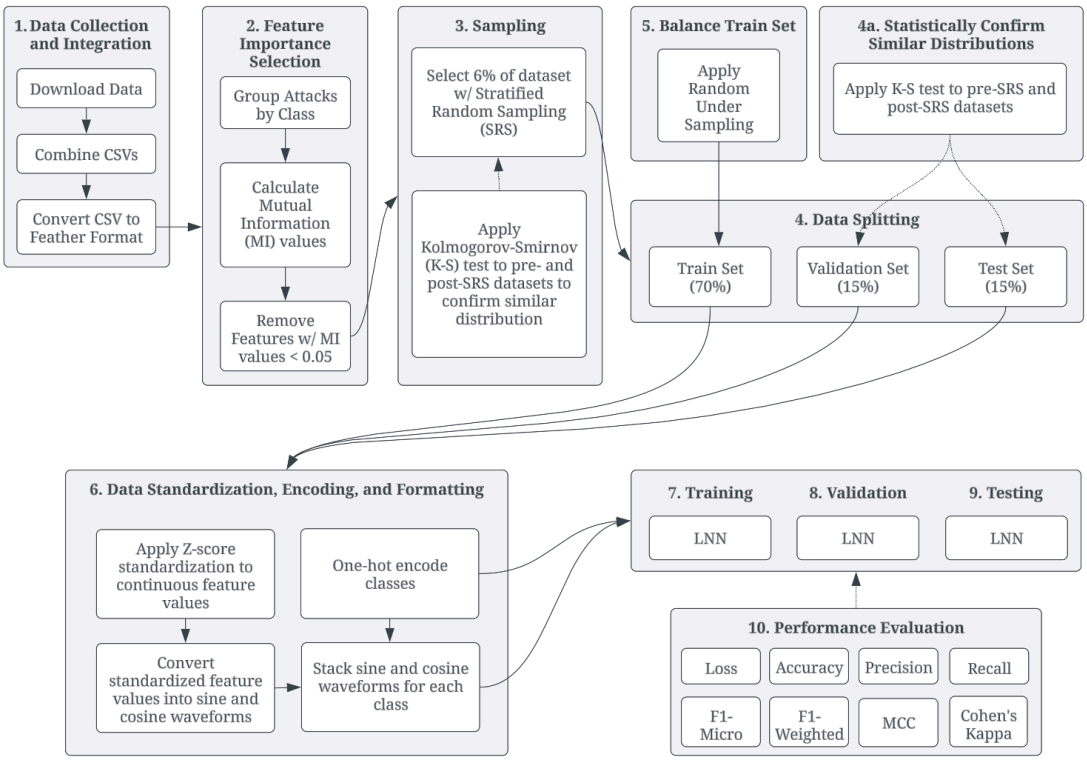

Figure 3-1: Liquid NeurIoTic Architecture

This chapter also establishes the framework for accepting or rejecting the research hypotheses. This praxis developed and evaluated an artificial neural network (ANN) and a recurrent neural network (RNN) to establish a comprehensive performance benchmark of LNNs. Section 3.1 provides the introduction and outline, while Section 3.2 discusses the dataset and its justification. Section 3.3 discusses the exploratory data analysis (EDA) methods used.

Section 3.4 details the data preprocessing techniques, which include attack grouping, feature selection, data splitting, data sampling, data distribution similarity testing, and feature scaling. Next, Section 3.5 examines the architectures of the developed ANN, RNN, and LNN models. Section 3.6 examines the hyperparameters of each model, while Section 3.7 covers the metrics used to evaluate each model's performance.

3.2 Canadian Institute for Cybersecurity Dataset

The Canadian Institute for Cybersecurity (CIC) Internet of Things (IoT) 2023 (CICIoT2023) dataset was selected to develop Liquid NeurIoTic. Neto et al. (2023) created this novel and comprehensive IoT attack dataset to enhance the development of security analytics applications, like NN-based IDSs, across any industry for real-world IoT operations. This dataset offers unparalleled robustness by simulating 33 unique attacks across seven distinct categories within an IoT network comprised of 105 devices. This level of comprehensiveness is not found in other healthcare or non-healthcare IoT datasets, as most studies have not fully incorporated a comprehensive network topology with real IoT devices (Neto et al., 2023). The seven distinct attack types are distributed denial-of-service (DDoS), denial-of-service (DoS), reconnaissance (Recon), web-based, brute force, spoofing, and Mirai (Neto et al., 2023).

These seven attack types encompass a total of 33 subtypes. Under DDoS, the subtypes include RSTFINFlood, PSHACK_Flood, SYN_Flood, UDP_Flood, TCP_Flood, ICMP_Flood, SynonymousIP_Flood, ACK_Fragmentation, UDP_Fragmentation, ICMP_Fragmentation, SlowLoris, and HTTP_Flood. DoS includes UDP_Flood, SYN_Flood, TCP_Flood, and HTTP_Flood. Recon includes OSScan, PortScan, VulnerabilityScan, and HostDiscovery. Mirai features greeth_flood, greip_flood, and udpplain. Spoofing includes DNS_Spoofing and MITM-ArpSpoofing. Web-based includes BrowserHijacking, Backdoor_Malware, XSS, Uploading_Attack, SqlInjection, and CommandInjection. Brute Force encompasses the DictionaryBruteForce subtype.

This dataset is exceptionally relevant given the projected growth of non-medical IoT connections in healthcare as IoT-H networks increasingly comprise a diverse array of non-medical smart IoT devices that can serve as gateways to the entire network (Neto et al., 2023). Additionally, this dataset is comprehensive in scope and strategically designed to facilitate the advancement and refinement of machine learning (ML) models, like LNNs, in IoT-H malware detection (Neto et al., 2023). For example, Khan and Alkhathami (2024) used this dataset to develop various ML models to enhance IoT-H security, including random forest (RF), AdaBoost, logistic regression (LR), perceptron, and deep neural networks (DNNs). Abbas et al. (2024) utilized this dataset to train NNs in detecting cyberattacks within various IoT networks, achieving 96.56% accuracy with an RNN. Jony and Arnob (2024) employed the same dataset to create a long short-term memory (LSTM)-based neural network (NN) that detected known and emerging

cyberattack patterns across various IoT environments and achieved 98.75% accuracy with an F1 score of 98.59%.

3.2.1 Dataset Collection and Integration

The CICIoT2023 dataset was released for public access online by Neto et al. (2023) through an Apache server running on Ubuntu. The authors recorded real-time traffic data during the attacks and captured them in packet capture (PCAP) files. To process this data, the authors utilized Mergecap, a tool designed to merge multiple PCAP files into a single file. Next, they used PySpark, a Python library for data management; TCPDump, a command line tool to divide the PCAP files into smaller segments; and DPKT, a Python library for packet feature extraction. The features were extracted into 169 comma-separated value (CSV) files for specific use in ML development, which resulted in a total of 46,686,579 records and an aggregation of 13.75 GB in size.

For this praxis, all 169 CSV files were downloaded and imported into Python using Pandas, a Python library known for its simplicity and efficiency with large datasets. Each CSV file was stored in a data frame. Pandas was used to seamlessly merge every data frame into a comprehensive data frame that contained every record.

3.2.2 Data Structure

The CICIoT2023 dataset consists of 46 feature columns with the independent variables and one label column containing the dependent variables. Independent variables are the inputs in the dataset used to predict the dependent variable or attack. A dependent variable is the dataset's output that serves as the target prediction for an ML model. Each row represents a record of traffic data categorized as benign traffic or an attack type from

one of this dataset's seven major attack categories. The independent variables and their descriptions are shown in Table 3-1 (Neto et al., 2023) below:

Table 3-1: CICIoT2023 Dataset Features with Descriptions

Feature	Description
flow duration	Duration of the packet's flow
Header Length	Header Length
Protocol Type	IP, UDP, TCP, IGMP, ICMP, Unknown
Duration	Time-to-Live (ttl)
Rate	Rate of packet transmission in a flow
Srate	Rate of outbound packet transmission in a flow
Drate	Rate of inbound packet transmission in a flow
fin flag number	Fin flag value
syn flag number	Syn flag value
rst flag number	Rst flag value
psh flag number	Psh flag value
ack flag number	Ack flag value
ece flag number	Ece flag value
cwr flag number	Cwr flag value
ack count	Number of packets with ack flag set in the same flow
syn count	Number of packets with syn flag set in the same flow
fin count	Number of packets with fin flag set in the same flow
urg count	Number of packets with urg flag set in the same flow
rst count	Number of packets with rst flag set in the same flow
HTTP	Indicates if the application layer protocol is HTTP
HTTPS	Indicates if the application layer protocol is HTTPS
DNS	Indicates if the application layer protocol is DNS
Telnet	Indicates if the application layer protocol is Telnet
SMTP	Indicates if the application layer protocol is SMTP
SSH	Indicates if the application layer protocol is SSH
IRC	Indicates if the application layer protocol is IRC
TCP	Indicates if the transport layer protocol is TCP
UDP	Indicates if the transport layer protocol is UDP
DHCP	Indicates if the application layer protocol is DHCP
ARP	Indicates if the link layer protocol is ARP
ICMP	Indicates if the network layer protocol is ICMP
IPv	Indicates if the network layer protocol is IP
LLC	Indicates if the link layer protocol is LLC
Tot sum	Summation of packet lengths in flow
Min	Minimum packet length in the flow
Max	Maximum packet length in the flow
AVG	Average packet length in the flow
Std	Standard deviation of packet length in the flow
Tot size	Packet's length
IAT	The time difference with the previous packet
Number	The number of packets in the flow
Magnitue	(Average lengths of incoming packets + average lengths of outgoing packets) ** 0.5

Radius	(Variance of lengths of incoming packets + variance of lengths of outgoing packets) ** 0.5
Covariance	Covariance of the lengths of incoming and outgoing packets
Variance	Variance of lengths of incoming packets *divided by* variance of lengths of outgoing packets
Weight	Number of incoming packets × Number of outgoing packets

3.3 Exploratory Data Analysis

Exploration is a fundamental aspect of the scientific process because it is crucial in validating scientific conclusions (Jebb et al., 2017). Therefore, exploratory data analysis (EDA) provided a deep understanding of each feature's data type of the CICIoT2023 dataset to guide the data preprocessing necessary for the training, validating, and testing phases of all NN models. This analysis involved row counting; computing mean values, standard deviations, minimum and maximum values; and inspecting the 25th, 50th, and 75th percentiles. As a result, the scale of feature values was revealed, which aided in identifying irregularities and missing data.

A crucial part of the EDA in this praxis was identifying the most relevant features. Hence, a non-linear feature importance selection method called mutual information (MI) was calculated for every feature to quantify how much information a particular feature shared with the target variable. This calculation revealed the relative quality of each feature as a predictor of the target variable. For example, a higher MI value between a feature and the target variable implied that the feature was more relevant for predicting or explaining the target variable (Seok & Seon Kang, 2015).

This praxis employed MI because it could capture linear and non-linear dependencies between variables, unlike the Pearson correlation coefficient, which is limited to capturing linear relationships (Macedo et al., 2022). Hence, the MI was a powerful tool for selecting features in the CICIoT2023 dataset where relationships

between the variables were not linear. The entire feature selection process is discussed in detail in Section 3.4.2.

3.4 Data Preparation

Data preparation was an essential step taken in this praxis before NN model development to ensure accurate and reliable outcomes of model performance. This step involved refining the raw data quality to ensure the LNN was built on a solid foundation. Furthermore, it ultimately optimized LNN model performance as the model focused exclusively on key features that drove performance. The main preprocessing techniques included grouping attacks by class, feature importance selection, splitting data, stratified random sampling (SRS), random undersampling (RUS), feature scaling, and converting scaled features into waveforms.

3.4.1 Grouping Attacks by Class

Given that the primary objective of this study was to develop an LNN-based IoT-H malware detection system generalized across various instances, all 33 attack subtypes in the target variable column were grouped into one of their seven respective main attack types: DDoS, DoS, Recon, web-based, brute force, spoofing, and Mirai. This grouping was taken because NN models developed to recognize larger, more inclusive categories tend to exhibit superior generalization capabilities when encountering new and unfamiliar data (Dablain et al., 2023). By focusing on the overarching patterns of malicious activity, NN models become more resilient and adaptable to emerging threats than those concentrating on narrow, specific subcategories. This trait was particularly valuable for IoT-H malware detection, where the landscape of attacks is in constant flux (Affia et al., 2023). Furthermore, targeting the main categories rather than the subcategories simplified

the NN model, which led to faster training times and lower computational overhead. Lastly, by categorizing malware into broader categories, security teams could prioritize and allocate resources more effectively without becoming bogged down in the specifics of numerous subcategories.

3.4.2 Feature Importance Selection

MI was used to assess the relationship between each feature and the target attack class. MI has been a widely utilized method for feature selection to enhance the performance of NN-based IDSs in various IoT environments, including IoT-H. This application was well-documented by Alalhareth and Hong (2023), Al-Sarem et al. (2021), and Rihan et al. (2023), who demonstrated its effectiveness across different settings. MI is a numerical value that offers insight into the degree of association between two variables (Young et al., 2023). This numerical association helped identify the most relevant features in the CICIoT2023 dataset to boost the efficiency and effectiveness of the NN models. Employing MI helped sharpen the models' predictive capabilities by only retaining features with the highest MI for the target variable.

A unique aspect of MI is its ability to detect linear and non-linear associations between features and the target variable, which offers a significant advantage given the complex nature of IoT-H environments. Utilizing a streamlined set of the most relevant features, MI inherently minimizes the risk of overfitting. For example, models honed with such relevant data are more adept at generalizing to novel, unseen data, which provides a critical advantage for anticipating emerging attacks in the IoT-H landscape. Additionally, retaining only relevant features leads to more efficient computational processes, reduced training times, and lower memory requirements (Alalhareth & Hong,

2023), which is particularly advantageous in handling large-scale IoT-H network traffic data. Moreover, a model built on a pared-down set of features offers greater interpretability, which makes it easier to understand the influence of each feature on the model's predictions and, by extension, provides valuable insights into malware behavior.

Therefore, in this praxis, the MI between each feature variable denoted as X, and the target variable represented by Y, was formally defined in Equation 1 (Hernández & Samengo, 2019):

$$I(X;Y) = H(X) - H(X|Y), \tag{1}$$

where

$I(X;Y)$ = MI between X and Y,

$H(X)$ = entropy of X, and

$H(X|Y)$ = conditional entropy of X given Y.

The MI score ranged from 0 to 1 (Young et al., 2023). A score of 0 signified the absence of MI between two variables, which indicated that the feature lacked predictive power regarding the target variable (Young et al., 2023). Conversely, a score of 1 denoted perfect MI, which highlighted the feature's strong predictive capacity to anticipate the outcomes of the target variable (Young et al., 2023). If one of the two variables remained constant, the MI value was 0 since a static variable could provide no insight into a fluctuating variable (Young et al., 2023). Conversely, if both variables exhibited variation, and changes in one variable offered predictive insights into changes in the other, the MI value increased (Young et al., 2023). Whether the variables changed in the same or diametrically opposite manner did not matter since both scenarios yielded

identical MI values, which indicated that changes in one variable mirrored changes in the other (Young et al., 2023).

3.4.3 Splitting Data

The dataset was split into three subsets to evaluate the performance of the NN models: 70% for training, 15% for validation, and 15% for testing. Data splitting is a widely adopted practice for validating and training various NNs. This technique was extensively documented in research by Wozniak et al. (2021), Chaganti et al. (2022), Vijayakumar et al. (2023), and Khan and Alkhathami (2024). These studies employed data splits ranging from 70% to 80% for training and 20% to 30% for validation and testing, which showcased its critical role in NN development.

Most of the data was given to the training set because it played a pivotal role in teaching the model to recognize underlying patterns and features. The training set repeatedly presented the same data to the model to enhance its understanding of the data's complexities and accurately forecast outcomes for new, unseen samples. The validation set was utilized to gauge the model's performance as it learned. This iterative evaluation provided a feedback mechanism to improve the model's hyperparameters and settings during training.

The validation process was considered the model's guide. It indicated whether the model's training aligned with the desired outcomes to prevent overfitting. Overfitting occurs when a model excels in training data predictions but cannot accurately extend them to new, unseen data (Chu et al., 2021).

Lastly, the test set served as the final arbiter of the model's training journey. It was utilized to evaluate the model post-training by offering an objective measure of its

final performance. Therefore, splitting the data into training, validation, and test sets provided this praxis with a solid foundation to develop LNN models proficient in learning during training and reliable and generalizable in their predictive capabilities post-training.

3.4.4 Stratified Random Sampling

Given the extensive size of the CICIoT2023 dataset, this praxis utilized a smaller, representative sample to enhance training speed and efficiency, which was achieved during the data-splitting process through SRS because the selected sample preserved the original dataset's distribution and its key population characteristics. SRS is a well-established technique when preparing data for NN training and was well-documented in Sadaiyandi et al.'s (2023) and Ullah et al.'s (2022) studies. These studies highlighted its effectiveness in generating the required number of samples, ensuring precise categorization for each sample, and maintaining representational accuracy across categories.

SRS divides a population into distinct subgroups known as strata based on the target variable and conducts random sampling within each stratum until each subgroup is adequately represented in the sample (Nguyen et al., 2021). As a result, the distribution of target classes across each split is preserved, which ensures that the samples' proportional representation remains consistent in maintaining the dataset's integrity during the splitting process (Nguyen et al., 2021). The SRS process begins by dividing the entire population into unique, non-overlapping groups. Subsequently, a specific sample size is determined. The proportion of elements to be sampled from each group is calculated using the SRS formula in Equation 2 (S. Bhushan et al., 2023):

$$n_i = \frac{n}{N} \times N_i, \quad (2)$$

where

n_i = sample size of the stratum,

n = size of the entire sample,

N = total population size, and

N_i = total number of elements in the i^{th} stratum.

Based on these proportions, elements were randomly selected for each group to ensure each was included in only one group until the predetermined sample size was achieved.

3.4.5 Random Undersampling

RUS was employed in the training set to address the problem of class imbalance by equalizing class distribution. Thus, each class had the same number of observations as the minority class, which provided a fair and balanced dataset for the model to use for training. RUS is a common sampling technique employed to reduce the number of certain classes and balance a dataset for NN training.

For example, Khan and Alkhathami (2024) employed RUS on the CICIoT2023 dataset to ensure a balanced representation of training samples for their NN-based IoT IDS. Datasets are imbalanced when the observations in one class far exceed those in other classes, which can cause NN models to favor the majority class during the training process and perform poorly. RUS randomly chooses a subset from the majority classes and decreases its observation count until it matches the minority class size (Fujiwara et al., 2020).

This method was carried out in this praxis without replacement, which meant that once a majority class observation was chosen, it was excluded from any future selection.

This approach maintained the original data's integrity without introducing duplicates since each observation from the majority class was given a single chance to contribute to the balanced dataset. RUS was not utilized on the validation and test sets to replicate the real-world class imbalance that LNNs will encounter.

3.4.6 Data Distribution Similarity Test

A two-sample Kolmogorov-Smirnov (K-S) test was applied to each feature variable in the pre-SRS and post-SRS datasets for goodness of fit. This test verified that the new class samples had the same distributions as the original dataset. The K-S test is a common method used to test the similarity of two distributions.

For example, Weinger et al. (2022) utilized the K-S test on their data samples before training their NNs to verify whether two samples shared identical distributions. The K-S test is non-parametric and determines if two samples have the same underlying distribution (Hassani & Silva, 2015). This non-parametric test can handle data with any distribution shape.

In this praxis, the K-S test's two-sided nature allowed it to detect distribution differences across both tails, which was particularly valuable for analyzing the inherently variable nature of IoT-H network traffic. The K-S test evaluated the maximum difference between the two samples' empirical cumulative distribution functions (eCDFs) to detect significant discrepancies (Hassani & Silva, 2015). The eCDFs provided a snapshot of the data distribution to understand the spread and central tendency of the observed values. This difference in the two samples was measured against a critical value from the K-S distribution table.

Moreover, a *p*-value was also calculated to assess statistical significance (Hassani & Silva, 2015). The K-S test results ranged from 0 to 1, with values close to 0 suggesting identical distributions and values near 1 indicating significant differences (Hassani & Silva, 2015). *P*-values, also ranging from 0 to 1, assessed the probability that observed differences are due to chance. A *p*-value below 0.05 typically denotes significant distribution differences, while values above 0.05 suggest similarity. Therefore, a significance threshold for the *p*-value in this praxis was set at 0.05. The two-sample K-S test began by first calculating the eCDFs and then inserted them into the two-sample K-S test functions as $F_{1,n}(x)$ and $F_{2,m}(x)$, as shown in Equation 3 (Hassani & Silva, 2015):

$$D_{n,m} = sup_x \left|F_{1,n}(x) - F_{2,m}(x)\right|, \tag{3}$$

where

sup_x = supremum or largest value over all possible values of x,

$F_{1,n}(x)$ = eCDF of sample one with n observations, and

$F_{2,m}(x)$ = eCDF of sample two with m observations.

3.4.7 Feature Scaling

All continuous variables were normalized using *z*-score standardization to ensure uniform influence of all features in model training, as shown in Equation 4 (Andrade, 2021), to accommodate the dynamic nature of IoT-H malware:

$$z_i = \frac{(x_i - \mu)}{\sigma}, \tag{4}$$

where

z = *z*-score or standardized data points,

x = original data points,

μ = mean of the data points, and

σ = standard deviation of the data points.

This technique has been commonly employed in the existing literature to standardize data for NN development.

For example, Beshah et al. (2024) and Khan and Alkhathami (2024) applied this method to the CICIoT2023 dataset, the same dataset used in this praxis, to standardize the data on a uniform scale. This method transformed every feature to have a mean of 0 and a standard deviation of 1. This praxis chose this technique to prevent features with higher values from overpowering the models while preserving the original data distribution. This preservation of data distribution ensured that the LNN models effectively adapted to and identified shifts in network behavior without being misled by overly large values in the input data, which is critical in IoT-H network environments where network dynamics constantly evolve.

3.4.8 Converting Standardized Features Into Waveforms

As IoT networks frequently exhibit peculiar behaviors and patterns, each feature record was transformed into waveforms to encode more information about the temporal relationships and patterns within the data. This technique provided more context for the NNs to capture a nuanced understanding of complex malware behaviors that were not apparent in static data. Prior studies have demonstrated the viability of this approach. For instance, Dobrojevic et al. (2023) enhanced and repositioned ToN-IoT datasets using sine and cosine functions. Similarly, Xiong et al. (2023) converted IoT network traffic data into text format and applied sine and cosine functions to create positional encodings to capture the relative positioning of words.

The conversion to waveforms in this praxis was accomplished through a scaling operation that multiplied each z-score standardized value by 2π and then applied the sine and cosine functions to these scaled values. The sine and cosine transformations are shown in Equations 5 and 6, respectively:

$$Sine\ Waveform = sin(x_i \times 2\pi), \tag{5}$$

$$Cosine\ Waveform = cos(x_i \times 2\pi), \tag{6}$$

where

x_i = i-th standardized input value,

2π = scales the standardized input data into a full circle,

sin = generates sine waveforms, and

cos = generates cosine waveforms.

This transformation converted each standardized feature into a pair of (x, y) coordinates on a unit circle, where the cosine waveform was the x-coordinate, and the sine waveform was the y-coordinate. As a result, two new features were generated for each feature in the dataset: one representing the sine of its standardized value and the other representing the cosine. Both values were stacked together to create a two-dimensional data structure for input into the LNN models. No existing scholarly literature has applied this technique to IoT network traffic data, particularly for training LNNs.

3.5 Neural Network Models

Given that LNNs are fundamentally NNs, this praxis developed simple artificial neural networks (ANNs) to establish a baseline performance for NN classifiers. Additionally, RNNs were developed to serve as a baseline comparison for temporal-

based models, given that LNNs are also advanced temporal models. This section examines these three models.

3.5.1 Artificial Neural Networks

Given the exceptionally compact footprint of LNNs, this praxis developed simple ANNs to benchmark the performance of extremely small-footprint NNs. ANNs use digital signals to process data feedforward using three key layers: the input layer, one or more hidden layers, and the output layer (Chen & Liu, 2022). Each layer comprises neurons connected to neurons in its consecutive layer with specific weight thresholds representing the strength of each neuron's digital synapses (Han et al., 2018). For example, a positive weight value activates another neuron, while a negative or zero weight value suppresses it.

In this praxis, three simple ANNs were developed, each adhering to a uniform three-layer architecture comprising one input layer, one hidden layer, and one output layer. Each ANN had a different number of neurons within the hidden layer, with configurations of 10, 20, and 30 neurons. This approach was adopted to achieve the smallest possible footprint for comparison. The total number of trainable parameters affecting the model's total size is shown in Table 3-2.

Table 3-2: ANN Configurations and Sizes

Model	Hidden Layer Neurons	Total Parameters	Size
ANN 1	10	476	1.86 KB
ANN 2	20	946	3.70 KB
ANN 3	30	1,416	5.53 KB

Within each neuron of the ANN's hidden layer, the rectified linear unit (ReLU) activation function was employed due to its effectiveness in promoting non-linear properties without affecting the scalability of the network (Parhi & Nowak, 2020). ReLU enabled the model to handle non-linear data and large numbers efficiently by squishing values within a certain range, for instance, between 0 and 1 (Parhi & Nowak, 2020). The specific activation function selected determined the exact range according to the task. The SoftMax activation function was chosen for the output layer because it excelled in handling multi-class classification tasks by converting logits into probabilities summing to 1. The general mathematical form of an ANN's neuron output is demonstrated in Equation 7:

$$y = \phi\left(\sum_{i=1}^{n}(w_i \times x_i) + b\right), \tag{7}$$

where

ϕ = activation function,

w_i = weight associated with the i-th input,

x_i = i-th input,

b = bias term, and

n = total number of inputs to the neuron.

Notably, the ANN is essentially a linear regression model without the activation function (Han et al., 2018). For example, a linear model defined as $y = Wx + b$ with input $x = 100$ and $y = 200$ values intuitively knows that $W = 2$ and $b = 0$ (Han et al., 2018). In the same vein, when examining similar x and y values, the ANN seeks optimal weight (W) and bias (b) values but through a more advanced trial-and-error method that incorporates a loss function, backpropagation, and an optimizer (Han et al., 2018).

This entire trial-and-error process began by initializing random W and b values and then calculating the loss function to measure prediction accuracy (Han et al., 2018). Through backpropagation, the W and b values were iteratively adjusted by calculating their gradients with respect to the loss, indicating the direction for minimization (Han et al., 2018). An optimizer used this information to update the W and b values and enhance model performance (Han et al., 2018). This cycle was repeated for a set number of epochs or until the improvements were negligible, at which point the weights were optimized (Han et al., 2018).

This praxis opted for categorical cross entropy (CCE) as the loss function of all three ANNs because it measured the discrepancy between the predicted probabilities and the actual distribution (Kerkhof et al., 2023). This approach was ideal for classifying multiple classes in the CICIoT2023 dataset. This praxis selected Adaptive Moment Estimation (Adam) for the optimizer because it enhanced the model's efficiency by adjusting the learning rate for each parameter to ensure quicker convergence and enhanced performance (Hassan et al., 2023).

3.5.2 Recurrent Neural Network

LNNs are inherently sequential models, three RNNs were developed in this praxis to establish a performance benchmark in sequential NNs with small footprints. Sequential models process data represented as a series of elements, each corresponding to a time step and presented to the network in a specific order (Ostmeyer & Cowell, 2019). For example, in sequence $S = \{x_1, x_2, x_3, \dots, x_n\}$, each element x_i is an individual data point processed by the network at its corresponding time step.

Each RNN developed in this praxis was structured with a consistent three-layer architecture comprising an input, hidden, and output layer. This uniform design facilitated direct comparisons with ANNs and LNNs. The distinguishing factor of these RNNs was the variability in the neuron count within their hidden layers, which impacted the overall number of trainable parameters and, ultimately, the total size of the model, as shown in Table 3-3.

Table 3-3: RNN Configurations and Sizes

Model	Hidden Layer Neurons	Total Parameters	Size
RNN 1	10	1,336	5.22 KB
RNN 2	20	2,866	11.20 KB
RNN 3	30	4,596	17.95 KB

Like ANNs, RNNs process data feedforward through an input layer, one or more hidden layers, and an output layer. However, the key component of RNNs is their memory feature, called a hidden state (Garcia et al., 2021), which provides the network with a memory to remember what it has seen at each time step to influence the current and future states of the network. This approach is foundational to RNNs since it enables them to capture unique temporal dynamics and dependencies within the data (Garcia et al., 2021) by processing the input sequentially at each time step with the hidden state to create a loop (i.e., recurrent connection) that integrates past and present information. This process enables the RNN to make predictions based on the sequence of events or data

points as a whole rather than in isolation, like in ANNs. Equation 8 demonstrates how the RNN's hidden state was updated at each time step:

$$x(t+1) = f(x(t), I(t), t;\ \theta), \tag{8}$$

where

t = input to the RNN at time t,

$x(t)$ = state of the RNN at time t,

$x(t+1)$ = updated state of the RNN at the next time step $t+1$

f = function that determines how RNN is updated, and

θ = parameters of the function f learned during training.

Due to the sequential processing inherent in RNNs, the hyperbolic tangent (tanh) activation function was employed within its neurons, which is standard in RNN development (Shen et al., 2022). This practice was complemented by using the CCE loss function, BPTT, and an Adam optimizer for each RNN. The structured step-by-step computations of RNNs necessitated BPTT to effectively update the network's memory at each timestep (Bellec et al., 2020). The tanh activation function output values between –1 and 1 to crucially preserve and scale the gradients during BPTT and enable efficient learning and model stability.

3.5.3 Liquid Neural Network

The LNN model this praxis developed for IoT-H malware detection and classification comprised CfC networks with neural circuit policy (NCP) wiring. CfCs are essentially faster LTCs, while NCPs are optimized neuron connection configurations specifically tailored to enhance the efficiency of LNNs. This approach to enhancing IoT-

H malware detection and classification has not been explored previously in the existing literature.

3.5.3.1 Closed-Form Continuous-Time Networks

CfC networks are essentially LTCs that do not use numerical ODE solvers, making them nearly 150 times faster than LTCs during training (Hasani et al., 2022). Since LTCs are highly dependent upon the rate at which an ODE solver can solve the ODEs at each neuron, their performance is limited due to the computational demands of solving ODEs (Hasani et al., 2021). CfC networks eliminate this performance constraint by calculating a closed-form solution of the ODEs (Hasani et al., 2022).

A closed-form solution is a straightforward equation that gives an exact answer each time, regardless of the input data's complexity (Borwein & Crandall, 2013). Furthermore, the CfC network accelerates and stabilizes the training process through a shared backbone architecture that branches into three network heads. This shared backbone is a main feature extractor that transforms raw data into a representation that the three network heads use. The model can independently explore temporal and structural dependencies by having separate network heads, which enhances its ability to capture complex data patterns. The CfC network is shown in Equation 9 (Hasani et al., 2022) and Figure 3-2 (Hasani et al., 2022):

$$x(t) = \sigma\left(-f\left(x, I; \theta_f\right) t\right) \odot g\left(x, I; \theta_g\right) + \left[1 - \sigma\left(-f\left(x, I; \theta_f\right)\right] t\right)] \odot h(x, I; \theta_h), \quad (9)$$

where

x = hidden state,

t = time,

I = input sequence,

σ = sigmoid function bounds output between 0 and 1,

$\odot$ = Hadamard product that combines the gated outputs of g and h,

f = modulates the gating mechanism,

g = influence on the state when the gate is open,

h = influence on the state when the gate is closed, and

θ = parameters of NNs.

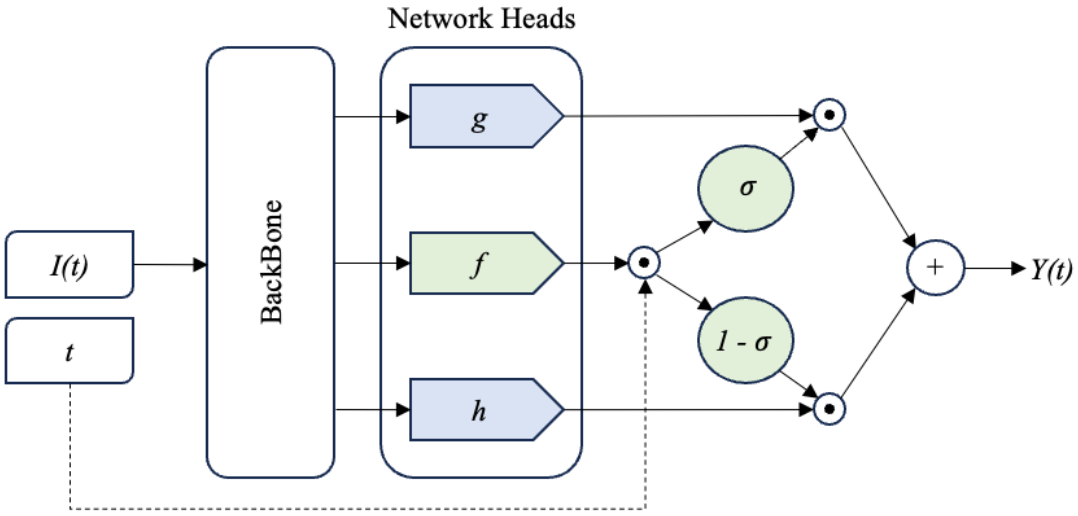

Figure 3-2: CfC Neuron Architecture

3.5.3.2 Neural Circuit Policies

The CfC networks in this praxis were structured with NCP wiring to further optimize performance. NCPs are modeled after the Caenorhabditis elegans (*C. elegans*) nematode's nervous system, known for its simplicity and well-mapped neural connections (Gleeson et al., 2018). Hence, a fast and efficient LNN topology was ideal for deploying throughout IoT-H IDSs.

NCPs are organized into four layers (Lechner et al., 2020). The first layer comprises sensory neurons acting like input neurons to receive data into the model (Lechner et al., 2020). The second layer comprises inter-neurons, which act as a hidden layer to process and learn how the data changes over time (Lechner et al., 2020). The third layer comprises command neurons that generate the output decision, and the fourth layer comprises motor neurons that receive the output from the command neurons to execute its generated decision (Lechner et al., 2020). From the sensory neurons to inter-neurons, NCPs have feedforward connections; from the inter-neurons to command neurons, they have recurrent connections; and from the command neurons to motor neurons, they have feedforward connections (Lechner et al., 2020). Given that the detection and classification of IoT-H malware constitutes multi-class classification, flatten and output layers were added to the NCP in this praxis, like the ones used in ANNs and RNNs, which enabled the NCP-CfC model to categorize the data into one of the numerous potential attack types. An example of the NCP architecture developed in this praxis is illustrated in Figure 3-3 below:

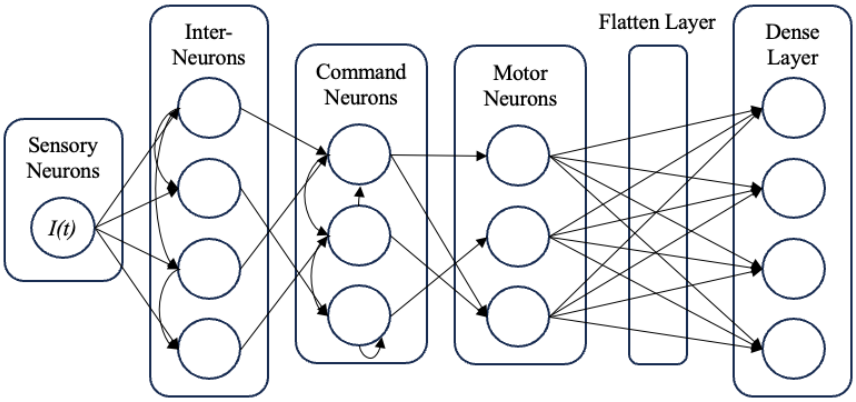

Figure 3-3: Example NCP Architecture with Flatten and Dense Layers

Each neuron is influenced by its connection through an input connection to another neuron (Lechner et al., 2020). For example, the state dynamics $x_i(t)$ of neuron i in an NCP are influenced by its connection through an input connection to another neuron j, as demonstrated in Equation 10 by Lechner et al. (2020):

$$\dot{x}_i = -\left(\frac{1}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(x_j)\right)x_i + \left(\frac{x_{leak_i}}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(x_j)E_{ij}\right), \quad (10)$$

where

$x_i(t)$ = state of neuron i at any given time t,

$\dot{x}_i$ = rate of change of state $\dot{x}_i$ of neuron i over time,

τ_i = time-constant of neuron i,

j = neuron which gives signals or influences to neuron i,

w_{ij} = synaptic weight (connection strength) from neuron i to j,

C_{m_i} = membrane capacitance (potential to change state) of neuron i,

$\sigma_i\left(x_j(t)\right)$ = sigmoid activation function applied to neuron j at time t,

x_{leak_i} = potential of neuron i when not activated, and

E_{ij} = excitatory or inhibitory synapses between neurons i and j.

Given the sequential nature shared with RNNs and LTCs, this praxis utilized the CCE loss function, Adam optimizer, BPTT, and Lecun tanh activation function to fine-tune the NCP-CfC model's weights. The Lecun tanh activation function is the default activation function in LNNs and shares a similar profile with the traditional tanh activation function but extends the boundaries, scaling the range from approximately –1.7 to 1.7. The main aspect of each NCP-CfC was the variability in the neuron count, which impacted the overall number of trainable parameters and the total size of the model to

facilitate direct comparisons between ANNs, RNNs, and LNNs. Table 3-4 displays the configurations and sizes of each developed NN.

Table 3-4: NCP-CfC Configurations and Sizes

		ANN		RNN		NCP-CfC (LNN)	
Model	Neurons	Parameters	Size	Parameters	Size	Parameters	Size
1	10	476	1.86 KB	1,336	5.22 KB	889	3.47 KB
2	20	946	3.70 KB	2,866	11.20 KB	1,625	6.35 KB
3	30	1,416	5.53 KB	4,596	17.95 KB	2,945	11.50 KB

3.6 Neural Network Hyperparameters

In this praxis, the LNN models were developed with a range of hyperparameters influencing their behavior and performance. The hyperparameter values were selected after extensive exploration to evaluate the models' sensitivity to these parameters. The hyperparameters across each model type were normalized to ensure reliable outcomes and facilitate a balanced comparison with ANNs and RNNs. This approach ensured that the comparison across different NNs was fair and consistent. All hyperparameter values tested in each model are illustrated in Table 3-5:

Table 3-5: Hyperparameter Values of Developed ANN, RNN, and LNN Models

	ANN	RNN	NCP-CfC
Hidden Neurons	(10, 20, 30)	(10, 20, 30)	(10, 20, 30)
Hidden Activation	ReLU	Tanh	Lecun Tanh
Output Activation	Softmax	Softmax	Softmax
Optimizer	Adam	Adam	Adam
Loss Function	CCE	CCE	CCE
Learning Rate	(0.01, 0.001, 0.0001)	(0.01, 0.001, 0.0001)	(0.01, 0.001, 0.0001)
Batch Size	(128, 256, 512)	(128, 256, 512)	(128, 256, 512)

3.7 Performance Evaluation Metrics

This praxis calculated standard evaluation metrics for all models developed to evaluate and benchmark LNN performance, including accuracy, precision, and recall. This approach aligned with the prevalent use of these metrics in most related studies from the literature review. This praxis also calculated the weighted-average F1-score, Matthews correlation coefficient (MCC), and Cohen's kappa, where the best possible score for all metrics was 1 to expand the evaluation scope and provide a holistic performance assessment.

Accuracy, demonstrated in Equation 11, quantified the proportion of correct predictions among all classifications (Hicks et al., 2022):

$$Accuracy = \frac{\sum_{c \in C} TP_c + TN_c}{\sum_{c \in C} TP_c + TN_c + FP_c + FN_c}, \tag{11}$$

where

C = set of all classes,

c = class belonging to the set of C,

TP_c = number of true positive predictions for class c,

TN_c = number of false negative predictions for class c,

FP_c = number of false positive predictions for class c, and

FN_c = number of false negative predictions for class c.

Precision, illustrated in Equation 12, assessed the model's accuracy in identifying positive cases without incorrectly marking negative instances as positive (Hicks et al., 2022). This assessment is crucial in IoT-H malware detection, where wrongly classifying benign traffic as malicious can drain company resources and distract from real threats.

$$Precision = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FP_c)}, \quad (12)$$

where

C = set of all classes,

c = class belonging to the set of C,

TP_c = number of true positive predictions for class c, and

FP_c = number of false positive predictions for class c.

Moreover, recall, shown in Equation 13, measured the proportion of actual positives correctly identified by the model, which was critical since overlooking a positive malware instance could lead to severe consequences (Hicks et al., 2022):

$$Recall = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FN_c)}, \quad (13)$$

where

C = set of all classes,

c = class belonging to the set of C,

TP_c = number of true positive predictions for class c, and

FN_c = number of false negative predictions for class c.

The weighted-average F1 score, demonstrated in Equation 14, computed the F1 score for each class and then averaged them (Takahashi et al., 2022) to give more weight to the more prevalent classes and ensure that classes with higher instances significantly influenced the overall score (Takahashi et al., 2022):

$$F1\ Score\ W = \sum_{c=1}^{C} w_c \times F1\ Score_c, \quad (14)$$

where w_c was equal to $\frac{Number\ of\ samples\ in\ class\ c}{Total\ number\ of\ samples}$.

The multi-class MCC, illustrated in Equation 15, considered all classes simultaneously and accounted for the balance between all types of errors in detecting multiple malware types in IoT-H networks. It was computed by evaluating each class against all others in pairwise comparisons (Gorodkin, 2004; Grandini et al., 2020):

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2)(s^2 - \sum_k^K t_k^2)}}, \quad (15)$$

where

c = number of accurately predicted samples,

k = class from 1 to k,

p_k = instances where class k was predicted,

s = number of samples, and

t_k = actual occurrences of class k.

Lastly, Cohen's kappa, shown in Equation 16, measured the agreement between two classifiers while correcting for chance agreement (Yilmaz & Demirhan, 2023):

$$k = \frac{p_o - p_e}{1 - p_e}, \quad (16)$$

where

p_o = empirical probability of agreement, and

p_e = expected agreement when both annotators assign labels randomly

3.8 Summary and Conclusion

This chapter comprehensively outlined the data, methodologies, tools, and procedures employed in developing and evaluating Liquid NeurIoTic, a novel LNN-based IDS for IoT-H networks. This chapter effectively laid the groundwork for the results presented in Chapter 4. It highlighted the extensive preparatory efforts

underpinning the credibility and aims of this praxis while aiming to accept or reject the hypotheses posited.

The CICIoT2023 dataset was presented as the ideal IoT-H malware dataset for this praxis due to its extensive network topology that simulates 33 unique attacks across seven categories. An in-depth examination of the processes involved throughout the EDA process was provided. It included data preparation techniques such as grouping attacks by class, feature importance selection using MI, data splitting, employing SRS to select a subset of the dataset to enhance training speed and efficiency, using RUS to balance the training set, applying the Kolmogorov-Smirnov test to pre-SRS and post-SRS dataset to confirm distribution similarity, feature scaling using *z*-score standardization, and converting the standardized features into sine and cosine waveforms. Lastly, the architecture and hyperparameters of the ANN, RNN, and LNN models developed and tested were presented, as well as the evaluation metrics used to assess each model's performance.

Chapter 4—Results and Discussions

4.1 Introduction

This chapter evaluates the performance of Liquid NeurIoTic, a liquid neural network (LNN)-based intrusion detection system (IDS) for Internet of Things in Healthcare (IoT-H) networks. Key metrics are compared against artificial neural networks (ANNs) and recurrent neural networks (RNNs), which this praxis also developed to establish baseline neural networks (NNs) and sequential NN performance.

The structure of this chapter follows Chapter 3 to provide a clear and comprehensive presentation of the outcomes generated throughout the methodologies adopted in this praxis. Section 4.1 overviews the chapter, while Section 4.2 delves into the findings of the exploratory data analysis (EDA). Section 4.3 unveils the results obtained through the data preprocessing phase, which includes subsections about grouping attacks by class, selecting features based on mutual information (MI) importance scores, splitting the data, sampling the data through stratified random sampling (SRS) and random undersampling (RUS). It verifies that the sampled datasets maintain the same distributions as the original dataset using the Kolmogorov-Smirnov (K-S) test, feature scaling with *z*-score standardization, transforming the standardized features into waveforms and converting the classes into a numerical form using one-hot encoding. Finally, Section 4.4 delves into the hyperparameters and architectures of the developed ANN, RNN, and LNN models, with the results of each model's performance discussed in Section 4.5.

4.2 Exploratory Data Analysis

This praxis completed the EDA process in four phases: 1) identifying data types, 2) uncovering null values, 3) collecting summary statistics, and 4) conducting feature importance analysis to identify the least informative features.

4.2.1 Identify Data Types

The initial EDA phase focused on identifying the CICIoT2023 dataset's features according to data types. This step crucially provided the foundation for the data preprocessing needed for the training, validation, and testing phases of all NNs developed in this praxis. The dataset comprised 46 features as 64-bit floating point numbers (i.e., the float64 data type) and two target variables (i.e., the object data type) comprised of alphanumeric characters (Table 4-1).

Table 4-1: Data Types in the CICIoT2023 Dataset

Feature Name	Data Type	Feature Name	Data Type	Feature Name	Data Type
flow_duration	float64	fin_count	float64	LLC	float64
Header_Length	float64	urg_count	float64	Tot sum	float64
Protocol Type	float64	rst_count	float64	Min	float64
Duration	float64	HTTP	float64	Max	float64
Rate	float64	HTTPS	float64	AVG	float64
Srate	float64	DNS	float64	Std	float64
Drate	float64	Telnet	float64	Tot size	float64
fin_flag_number	float64	SMTP	float64	IAT	float64
syn_flag_number	float64	SSH	float64	Number	float64
rst_flag_number	float64	IRC	float64	Magnitue	float64
psh_flag_number	float64	TCP	float64	Radius	float64
ack_flag_number	float64	UDP	float64	Covariance	float64
ece_flag_number	float64	DHCP	float64	Variance	float64
cwr_flag_number	float64	ARP	float64	Weight	float64
ack_count	float64	ICMP	float64	Label	Object
syn_count	float64	IPv	float64	Class	Object

Since NNs require numerical data, the float64 features were inherently compatible, whereas the object data type variables required conversion into a numerical format for processing, as performed in Section 4.3.9.

4.2.2 Uncovering Null Values

The second phase of the EDA focused on identifying missing or not available (NA) values. NA values can misrepresent the data and affect an LNN's ability to learn the underlying patterns effectively. The dataset was free of NA values, which made sense as each row of the CICIoT2023 dataset represented a unique network traffic record, and each column detailed a specific characteristic of that traffic.

4.2.3 Collecting Summary Statistics

The third phase in the EDA process involved collecting summary statistics about each numerical feature and understanding their scale to determine if standardizing the data was necessary. Table 4-2 shows significant disparities in the minimum, maximum, mean, and standard deviation values of several features. For example, duration and inter-arrival time (IAT) have mean values of 66.35 and 83,182,530.00, respectively, while flow duration and covariance have max values of 394,357.20 and 154,902,200.00, respectively. Hence, standardization across the entire dataset was necessary as these huge value differences could lead to higher-valued features overpowering the NN's performance at each neuron's processing.

Table 4-2: Summary Statistics for the Features of the CICIoT2023 Dataset

Feature Name	count	mean	Std	min	25%	50%	75%	Max
flow_duration	46686579	5.77	285.03	0.00	0.00	0.00	0.11	394357.20
Header_Length	46686579	76705.96	461331.70	0.00	54.00	54.00	280.56	9907148.00
Protocol Type	46686579	9.07	8.95	0.00	6.00	6.00	14.33	47.00
Duration	46686579	66.35	14.02	0.00	64.00	64.00	64.00	255.00
Rate	46686579	9064.06	99562.49	0.00	2.09	15.75	117.38	8388608.00

Srate	46686579	9064.06	99562.49	0.00	2.09	15.75	117.38	8388608.00
Drate	46686579	0.00	0.01	0.00	0.00	0.00	0.00	29.72
fin_flag_number	46686579	0.09	0.28	0.00	0.00	0.00	0.00	1.00
syn_flag_number	46686579	0.21	0.41	0.00	0.00	0.00	0.00	1.00
rst_flag_number	46686579	0.09	0.29	0.00	0.00	0.00	0.00	1.00
psh_flag_number	46686579	0.09	0.28	0.00	0.00	0.00	0.00	1.00
ack_flag_number	46686579	0.12	0.33	0.00	0.00	0.00	0.00	1.00
ece_flag_number	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
cwr_flag_number	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
ack_count	46686579	0.09	0.29	0.00	0.00	0.00	0.00	7.70
syn_count	46686579	0.33	0.66	0.00	0.00	0.00	0.06	12.87
fin_count	46686579	0.10	0.33	0.00	0.00	0.00	0.00	248.32
urg_count	46686579	6.24	71.85	0.00	0.00	0.00	0.00	4401.70
rst_count	46686579	38.47	325.38	0.00	0.00	0.00	0.01	9613.00
HTTP	46686579	0.05	0.21	0.00	0.00	0.00	0.00	1.00
HTTPS	46686579	0.06	0.23	0.00	0.00	0.00	0.00	1.00
DNS	46686579	0.00	0.01	0.00	0.00	0.00	0.00	1.00
Telnet	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
SMTP	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
SSH	46686579	0.00	0.01	0.00	0.00	0.00	0.00	1.00
IRC	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
TCP	46686579	0.57	0.49	0.00	0.00	1.00	1.00	1.00
UDP	46686579	0.21	0.41	0.00	0.00	0.00	0.00	1.00
DHCP	46686579	0.00	0.00	0.00	0.00	0.00	0.00	1.00
ARP	46686579	0.00	0.01	0.00	0.00	0.00	0.00	1.00
ICMP	46686579	0.16	0.37	0.00	0.00	0.00	0.00	1.00
IPv	46686579	1.00	0.01	0.00	1.00	1.00	1.00	1.00
LLC	46686579	1.00	0.01	0.00	1.00	1.00	1.00	1.00
Tot sum	46686579	1308.32	2613.30	42.00	525.00	567.00	567.54	127335.80
Min	46686579	91.61	139.70	42.00	50.00	54.00	54.00	13583.00
Max	46686579	181.96	524.03	42.00	50.00	54.00	55.26	49014.00
AVG	46686579	124.67	240.99	42.00	50.00	54.00	54.05	13583.00
Std	46686579	33.32	160.34	0.00	0.00	0.00	0.37	12385.24
Tot size	46686579	124.69	241.55	42.00	50.00	54.00	54.06	13583.00
IAT	46686579	83182530.00	17047350.00	0.00	83071570.00	83124520.00	83343910.00	167639400.00
Number	46686579	9.50	0.82	1.00	9.50	9.50	9.50	15.00
Magnitue	46686579	13.12	8.63	9.17	10.00	10.39	10.40	164.82
Radius	46686579	47.09	226.77	0.00	0.00	0.00	0.51	17551.27
Covariance	46686579	30724.36	323710.70	0.00	0.00	0.00	1.34	154902200.00
Variance	46686579	0.10	0.23	0.00	0.00	0.00	0.08	1.00
Weight	46686579	141.51	21.07	1.00	141.55	141.55	141.55	244.60

4.2.4 Feature Importance Selection

Feature importance selection was employed to enhance the LNN's efficiency and address overfitting concerns using MI analysis. This step aimed to ultimately enhance computational efficiency, reduce training times, and minimize memory requirements by pinpointing and preserving the most relevant features. MI importance scores were calculated with the results provided in Table 4-3.

Table 4-3: Mutual Information Scores of CICIoT2023 Dataset Features

Feature	MI Score	Feature	MI Score	Feature	MI Score
IAT	0.853298	Magnitue	0.160794	SMTP	0.000000
Tot size	0.590882	syn_count	0.112479	IPv	0.000000
Tot sum	0.571683	Protocol Type	0.109098	ICMP	0.000000
AVG	0.565651	HTTPS	0.054733	DHCP	0.000000
Max	0.562430	ack_count	0.021110	UDP	0.000000
Min	0.419861	ack_flag_number	0.014322	TCP	0.000000
Header_Length	0.380891	fin_count	0.011953	HTTP	0.000000
Srate	0.244379	SSH	0.000408	psh_flag_number	0.000000
Rate	0.244322	DNS	0.000223	rst_flag_number	0.000000
Std	0.226168	ARP	0.000189	syn_flag_number	0.000000
Radius	0.225009	ece_flag_number	0.000060	fin_flag_number	0.000000
Covariance	0.221573	IRC	0.000047	Drate	0.000000
flow_duration	0.202469	cwr_flag_number	0.000019	Duration	0.000000
Variance	0.193522	Telnet	0.000018	Weight	0.000000
rst_count	0.189489	Number	0.000000		
urg_count	0.178557	LLC	0.000000		

Note. Green highlights are above the 0.05 threshold; red highlights are below the threshold.

After examining the MI analysis, 0.05 was determined as the optimal MI cutoff value that balanced model complexity while eliminating noise from less informative features (highlighted in Table 4-3). A threshold of 0.05 is frequently adopted as the optimal MI value to enhance the efficiency and accuracy of ML models. For example, Prasetiyowati et al. (2021) leveraged this threshold to boost speed and prediction accuracy, while Jeuken and Käll (2024) and Takagi (2020) utilized it to optimize feature selection.

With most features scoring well above MI scores of 0.05, setting the threshold at this range ensured the retention of valuable information while avoiding premature exclusion. As a result, the following 20 features of the CICIoT2023 dataset emerged as

the most informative for LNN development: inter-arrival times (IAT), total packet length (Tot size), summation (Tot sum), average (AVG), maximum (Max) and minimum (Min) packet length in the flow, header length, rate of packet transmission, rate of outbound packet transmission (Srate), radius, variance, covariance, flow duration, number of packets with reset flag (rst_count), urgent flag (urg_count), synchronize flag (syn_count), magnitue, protocol type, and HTTPS flag. These led to the commencement of the data preparation phase. The following 26 features were removed from the dataset given the identification of the most informative features: ack_count, ack_flag_value, ARP, cwr_flag_number, DHCP, DNS, Drate, Duration, ece_flag_number, fin_count, fin_flag_number, HTTP, ICMP, IPv, IRC, LLC, number, psh_flag_number, rst_flag_number, SMTP, SSH, syn_flag_number, TCP, Telnet, UDP, and weight.

4.3 Data Preparation

The data preparation phase of this praxis included grouping attacks by class, splitting the data, sampling the data through SRS and RUS, verifying that the pre-sampled datasets maintained the same distributions as the original dataset using the K-S test, feature scaling with *z*-score standardization, the transformation of the standardized features into sine and cosine waveforms, and converting the classes into numerical form through one-hot encoding.

4.3.1 Grouping Attacks by Class

The 33 class subtypes of the CICIoT2023 dataset, shown at the top of Figure 4-1, were grouped into their main class types: DDoS, DoS, Mirai, benign, spoofing, Recon, web, and brute force, shown at the bottom of Figure 4-2. As explained in Section 3.4.1, the 33 attack subtypes were grouped into seven primary categories to enhance the

generalization capabilities of NNs when faced with new and unfamiliar data. Fewer categories resulted in faster training times and reduced computational costs.

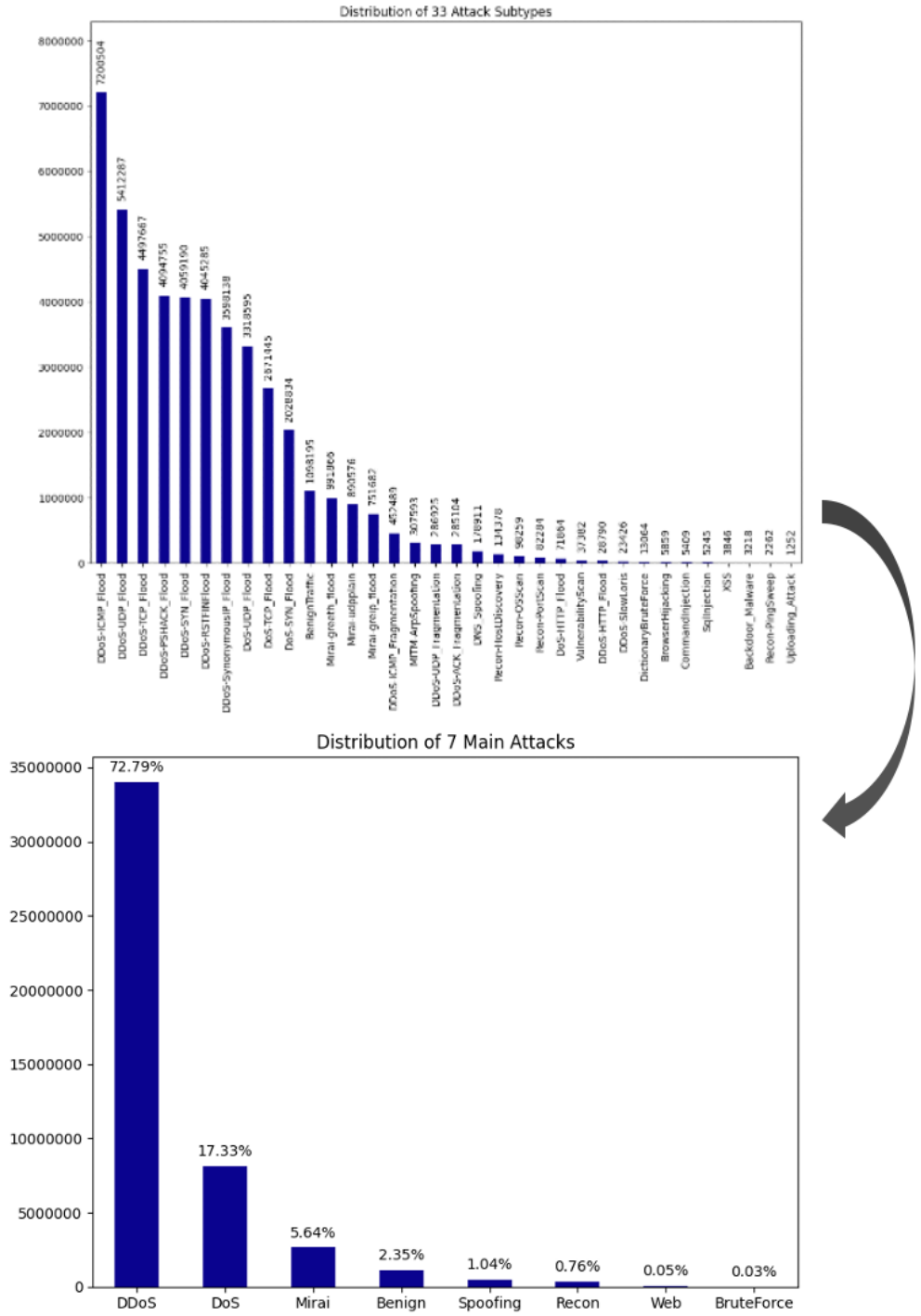

Figure 4-1: Distribution of 33 Subtypes Grouped Into Seven Main Classes

4.3.2 Splitting Data

The dataset was split into three subsets to train the NNs and evaluate performance: 70% for training, 15% for validation, and 15% for testing. The training set contained most of the data since teaching the model to discern underlying patterns and features was crucial. The validation set monitored the NN's learning progress, while the test set provided an evaluation of its final performance after training.

4.3.3 Stratified Random Sampling

Given the dataset's substantial size, SRS was used to select 6% of the original dataset to increase training speed and computational efficiency. The 6% proportion was selected as the minimum necessary to retain sufficient records for effective model training. SRS was chosen to preserve the original dataset's distribution and key population characteristics, as shown in Figures 4-2, 4-3, and 4-4.

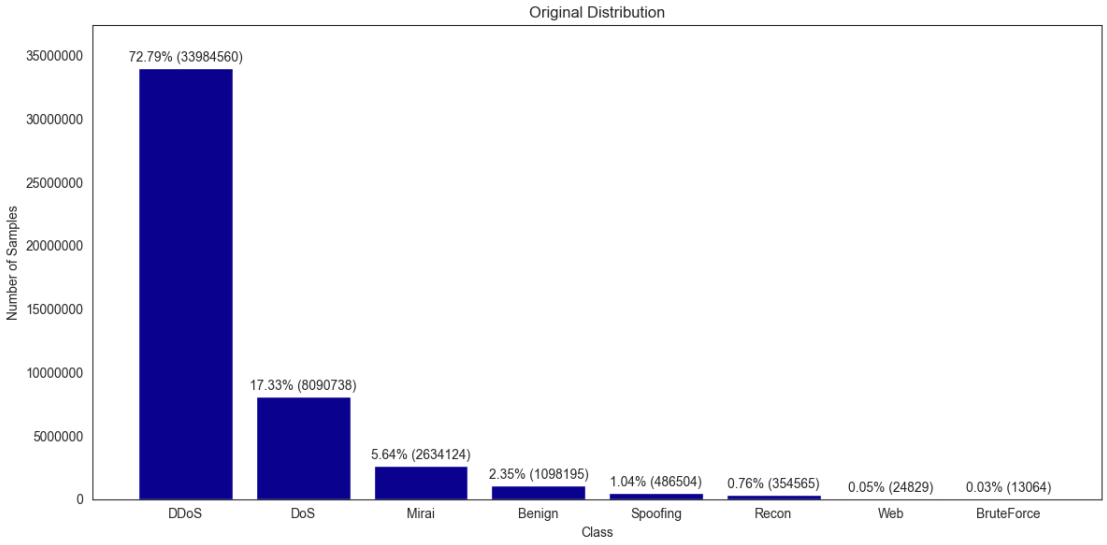

Figure 4-2: Original Distribution of CICIoT2023 Classes

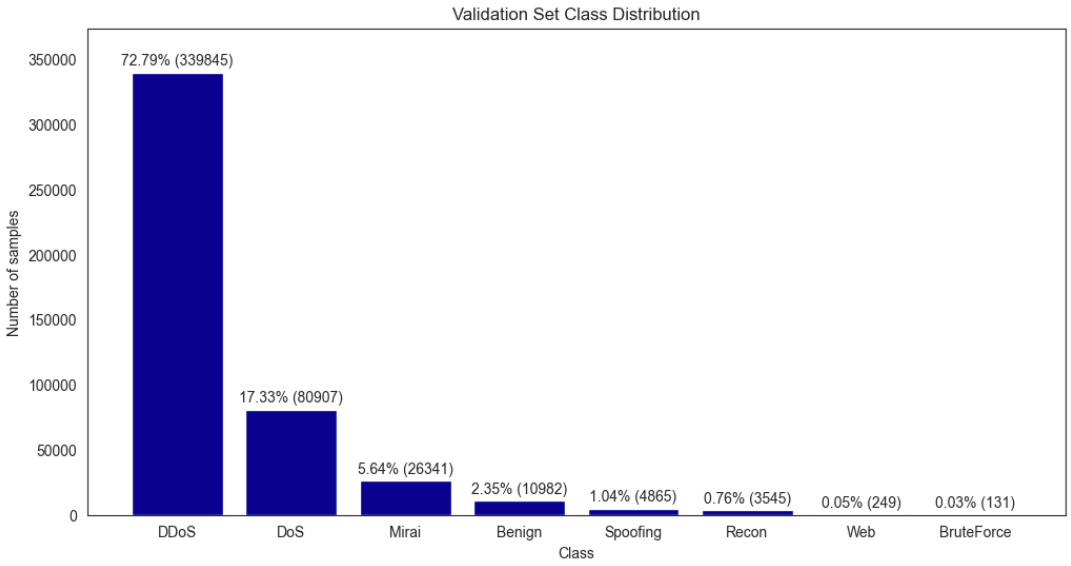

Figure 4-3: Validation Set Distribution After SRS

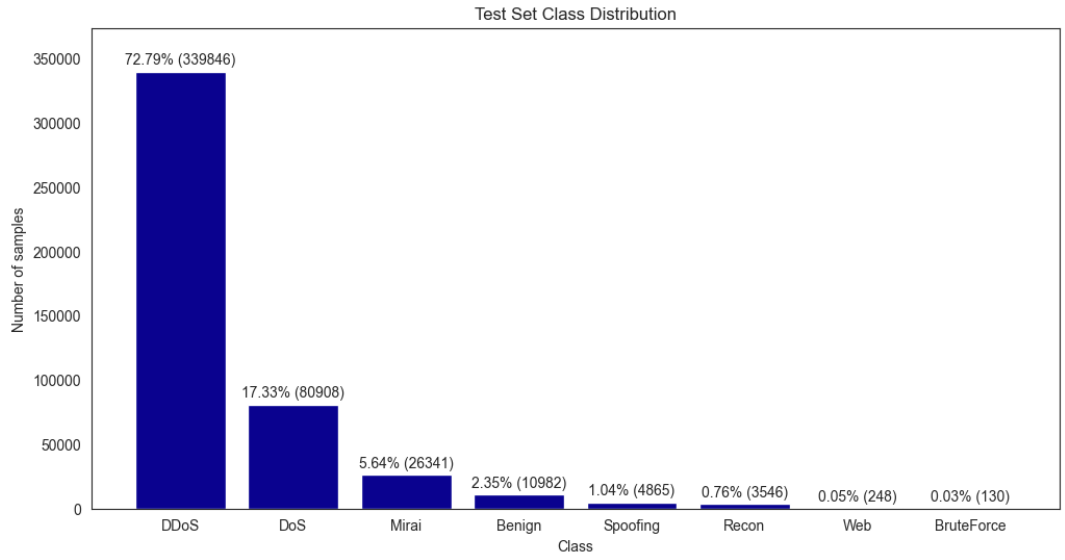

Figure 4-4: Test Set Distribution After SRS

4.3.4 Random Undersampling

RUS was applied to the minority class to balance the classes in the training set and address the dataset's significant imbalance in Section 4.3.1. During this phase, the

web-based and brute force attack classes had exceedingly low sample counts to provide enough samples to effectively balance the training set. As a result, they were deemed insufficient for effective model training and were dropped from the dataset during the splitting, SRS, and RUS processes, as shown in Figures 4-5, 4-6, and 4-7. These findings aligned with Van Der Ploeg et al.'s (2014) findings that robust NN outcomes require up to 50 events per variable. For example, with 20 features, the web-based and brute force classes necessitated a minimum of 1,000 records in the validation and test sets—a criterion these two classes did not meet. Lastly, the post-SRS validation and test sets maintained the same distribution as the original dataset upon visual inspection.

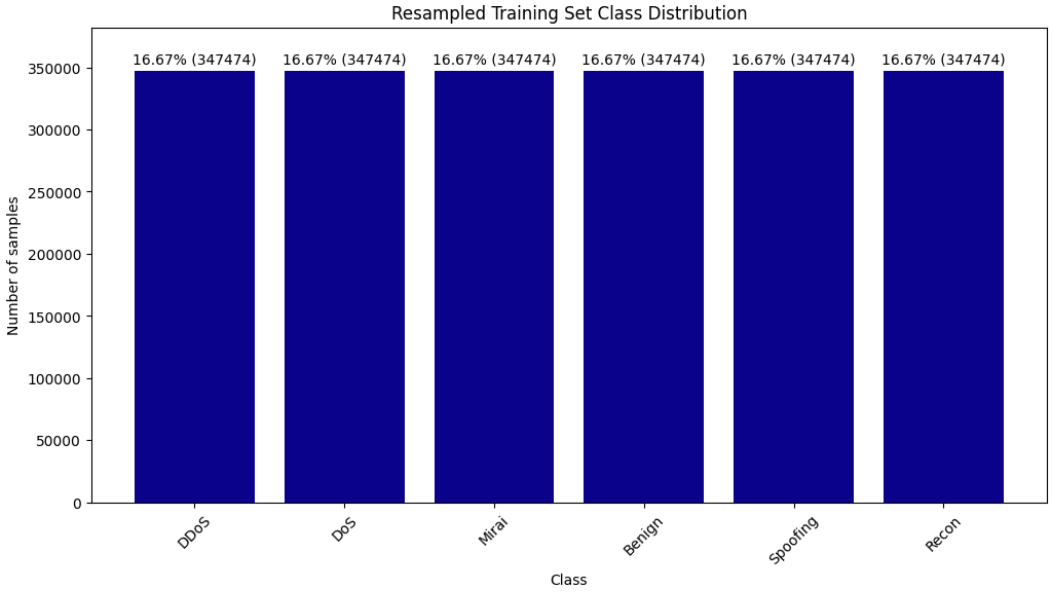

Figure 4-5: Training Set Balanced After Random Under Sampling

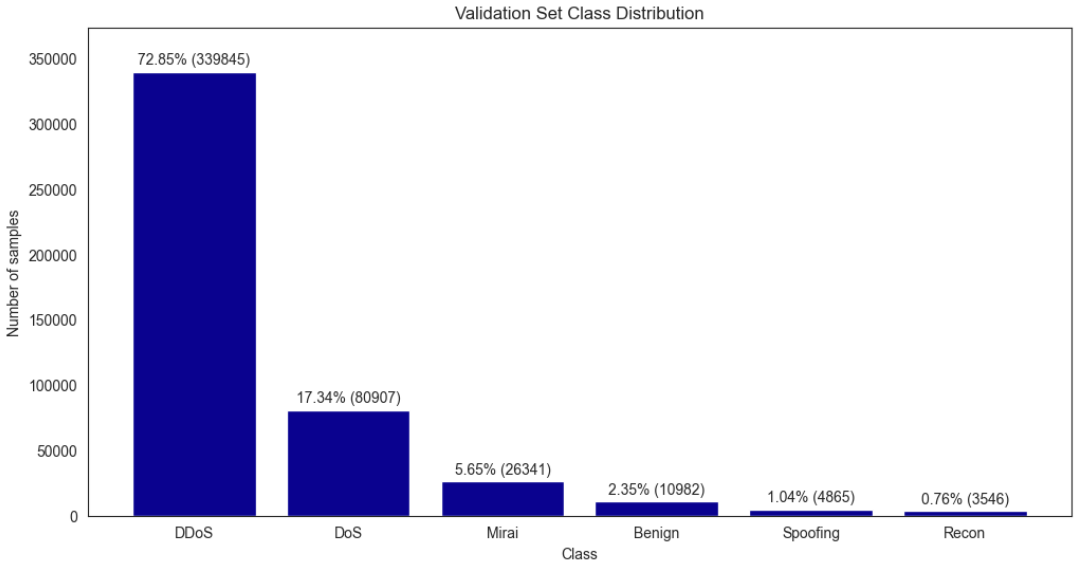

Figure 4-6: Validation Set After Removal of Web and Brute Force Attack Classes

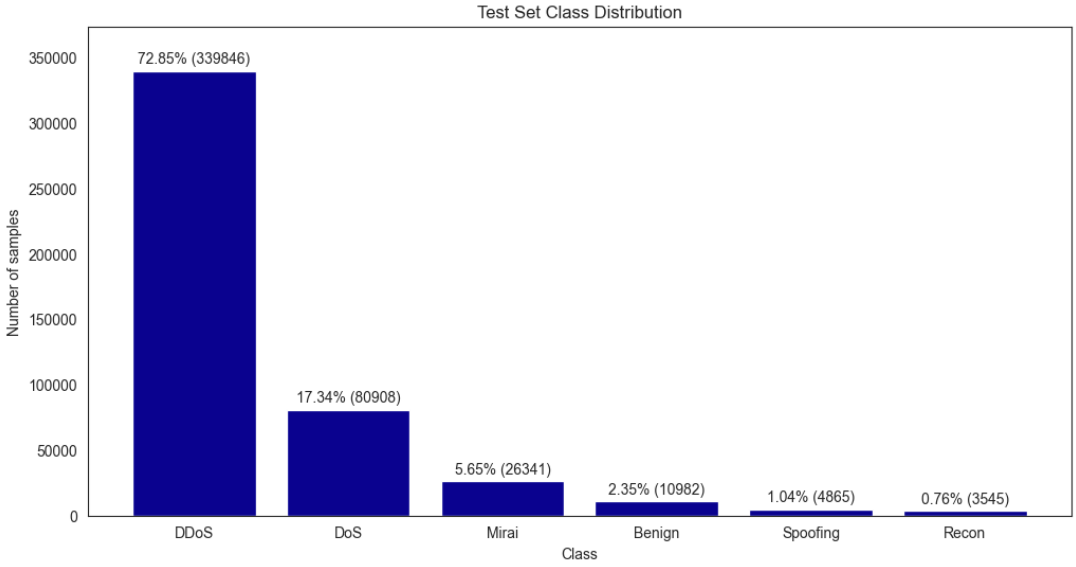

Figure 4-7: Test Set After Removal of Web and Brute Force Attack Classes

4.3.5 Data Distribution Similarity Test Using Kolmogorov-Smirnov

Visual inspection revealed that the post-SRS validation and test sets retained the same distribution as the original dataset. The K-S test was applied to each feature

variable in the pre-SRS and post-SRS datasets to confirm the goodness of fit, as presented in Table 4-4. All K-S values were below 0.00, and all *p*-values were above 0.05. K-S values close to 0, and *p*-values above 0.05 indicate identical distributions (Hassani & Silva, 2015). Hence, the pre-SRS and post-SRS datasets were statistically confirmed to maintain similar distributions, which mitigated sampling bias.

Table 4-4: Kolmogorov-Smirnov Test Results

	Train Set		Validation Set		Test Set	
Feature	K-S Statistic	*P*-Value	K-S Statistic	*P*-Value	K-S Statistic	*P*-Value
flow_duration	0.000012819	1.00	0.000859844	0.88	0.000859844	0.88
Header_Length	0.000015761	1.00	0.001105762	0.62	0.001105762	0.62
Protocol Type	0.000013582	1.00	0.001145894	0.58	0.001145894	0.58
Rate	0.000012943	1.00	0.000893854	0.85	0.000893854	0.85
Srate	0.000012943	1.00	0.000893854	0.85	0.000893854	0.85
syn_count	0.000020045	1.00	0.001146402	0.58	0.001146402	0.58
urg_count	0.000014126	1.00	0.001001725	0.74	0.001001725	0.74
rst_count	0.000018606	1.00	0.000937988	0.81	0.000937988	0.81
HTTPS	0.000000512	1.00	0.000210643	1.00	0.000210643	1.00
Tot sum	0.000024999	1.00	0.001605086	0.18	0.001605086	0.18
Min	0.000019202	1.00	0.001276231	0.44	0.001276231	0.44
Max	0.000022340	1.00	0.00133826	0.38	0.00133826	0.38
AVG	0.000021938	1.00	0.001317788	0.40	0.001317788	0.40
Std	0.000010149	1.00	0.00081156	0.92	0.00081156	0.92
Tot size	0.000022437	1.00	0.001306728	0.41	0.001306728	0.41
IAT	0.000015651	1.00	0.001088946	0.64	0.001088946	0.64
Magnitue	0.000021977	1.00	0.001301217	0.41	0.001301217	0.41
Radius	0.000011291	1.00	0.000818377	0.92	0.000818377	0.92
Covariance	0.000012528	1.00	0.000974692	0.77	0.000974692	0.77
Variance	0.000006726	1.00	0.000723125	0.97	0.000723125	0.97

4.3.6 Feature Scaling and Conversion into Waveforms

During the EDA process of this praxis in Section 4.2.3, huge value disparities in the dataset were uncovered. Thus, all continuous variables were normalized using *z*-score

standardization to equalize the impact of all features during model training by converting each feature's mean to 0 and standard deviation to 1. Then, each z-score's standardized value was augmented by multiplying it by 2π and applying sine and cosine functions to each augmented value. This process yielded two new features for each original feature in the dataset: one representing its sine value and the other representing its cosine value. These features were then stacked together to form two dimensions, as Figure 4-8 depicts, which enabled the input to be represented as time series and enhanced its utility for input into the LNN models. As a result, Hypothesis 1 of this praxis was accepted.

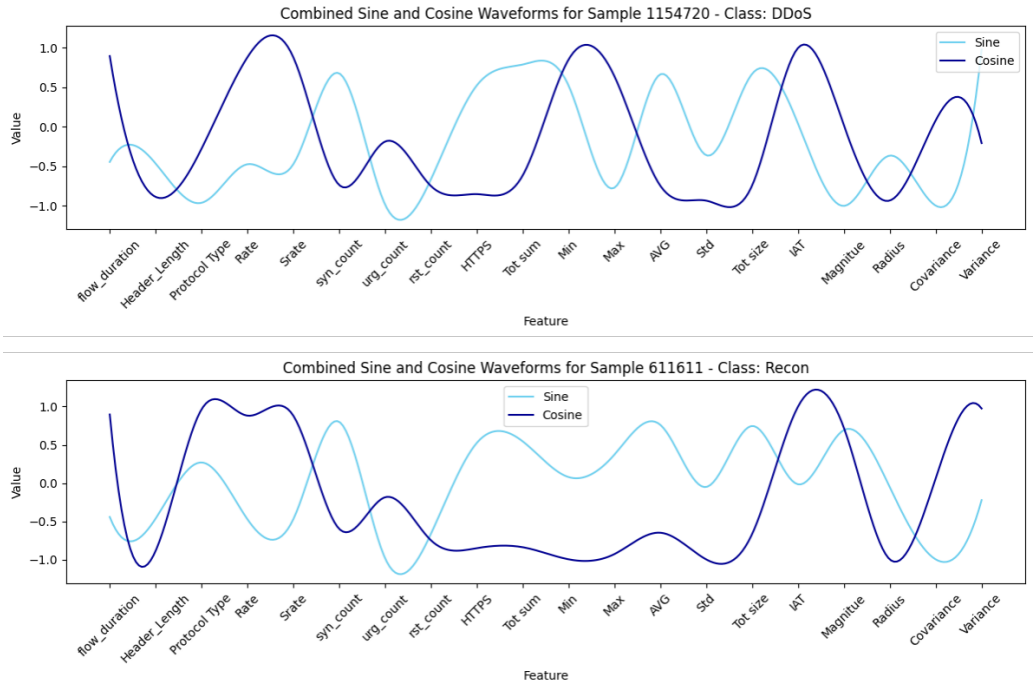

Figure 4-8: DDoS and Recon Attacks Converted Into Stacked Waveforms

4.3.7 One-Hot Encoding of Classes

During the initial EDA phase of this praxis in Section 4.2.1, categorical data, characterized by the object datatype, was identified within the dataset. Given that this

datatype represented the dataset's classes, this praxis employed the one-hot encoding technique to transform them into numerical form, as demonstrated in Table 4-5.

Table 4-5: One-Hot Encoded Representation of All Classes

Class	One-Hot Vector
Benign	[1, 0, 0, 0, 0, 0]
DDoS	[0, 1, 0, 0, 0, 0]
DoS	[0, 0, 1, 0, 0, 0]
Mirai	[0, 0, 0, 1, 0, 0]
Recon	[0, 0, 0, 0, 1, 0]
Spoofing	[0, 0, 0, 0, 0, 1]

4.4 Neural Network Models Development

This section analyzes the optimal architectures and hyperparameters discovered during the development and hyperparameter tuning of all ANNs, RNNs, and neural circuit policy (NCP) closed-form continuous-time (CfC) networks developed and tested, which were the LNNs developed in this praxis.

4.4.1 Optimal Hyperparameters and Architecture

Given the complexity of multi-class classification, a standardized approach in hyperparameter tuning was adopted to ensure fair performance evaluation, which involved using the default activation functions, optimizers, and loss functions for each NN type. Thus, the main hyperparameters tested to find the optimal architectures were the neuron counts, learning rate, and batch size. Furthermore, given the exceptionally compact footprint of LNN models, the exploration of neuron counts for the ANNs and RNNs in this praxis remained as few as possible, adhering to a defined range of 10 to 30 neurons. This approach was designed to keep the NNs as small as possible for

comparison with the NCP-CfCs. Considering the novel development of LNNs for IoT-H malware classification, an extended range of neuron values for the NCP-CfCs was investigated but still adhered to the pre-defined range between 10 to 30 neurons.

A total of 216 NN architecture variations were developed and evaluated in this praxis: 54 variations of ANNs, 54 variations of RNNs, and 108 variations of NCP-CfCs. The optimal hyperparameters for each model are presented in bold in Table 4-6. Example architectures of the optimal ANN, RNN, and NCP-CfC models identified in Table 4-6 are illustrated in Figures 4-9, 4-10, and 4-11, respectively.

Table 4-6: Optimal Hyperparameters for Each Model Developed

	ANN	RNN	NCP-CfC
Hidden Neurons	(10, 20, **30**)	(10, **20**, 30)	(10, **18**, 20, 24, 27, 30)
Hidden Activation	ReLU	Tanh	Lecun Tanh
Output Activation	Softmax	Softmax	Softmax
Optimizer	Adam	Adam	Adam
Loss Function	CCE	CCE	CCE
Learning Rate	(0.01, **0.001**, 0.0001)	(**0.01**, 0.001, 0.0001)	(**0.01**, 0.001, 0.0001)
Batch Size	(**128**, 256, 512)	(**128**, 256, 512)	(128, 256, **512**)
Liquid NeurIoTic Waveforms	Yes	Yes	Yes

Note. The best hyperparameter is in bold.

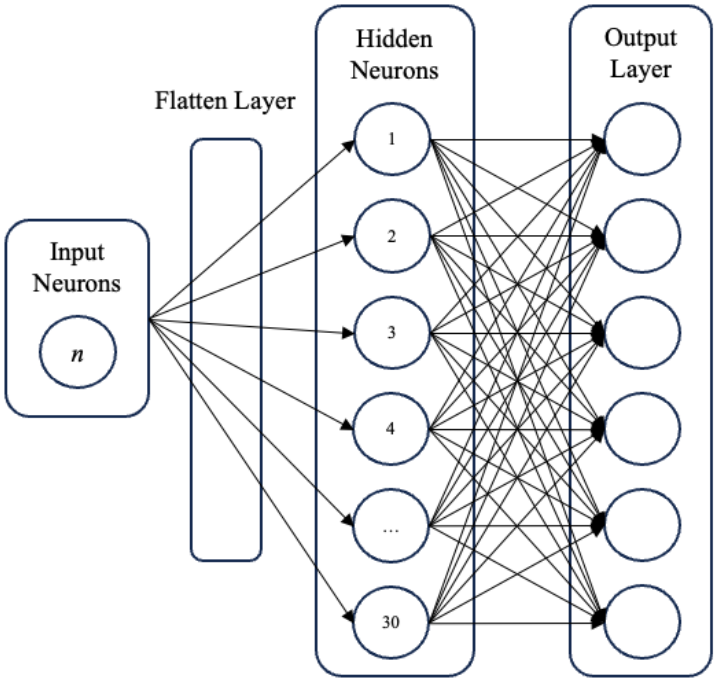

Figure 4-9: Optimal ANN Architecture With 30 Neurons

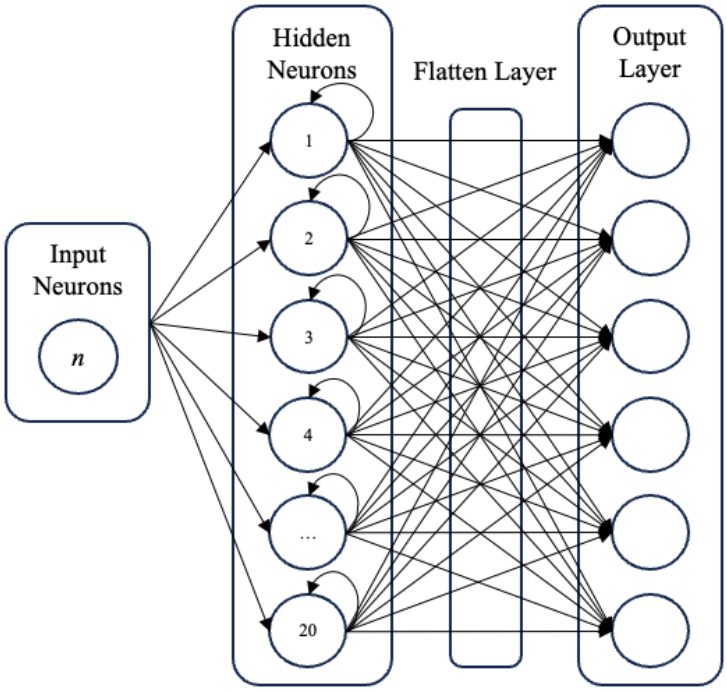

Figure 4-10: Optimal RNN Architecture With 20 Neurons

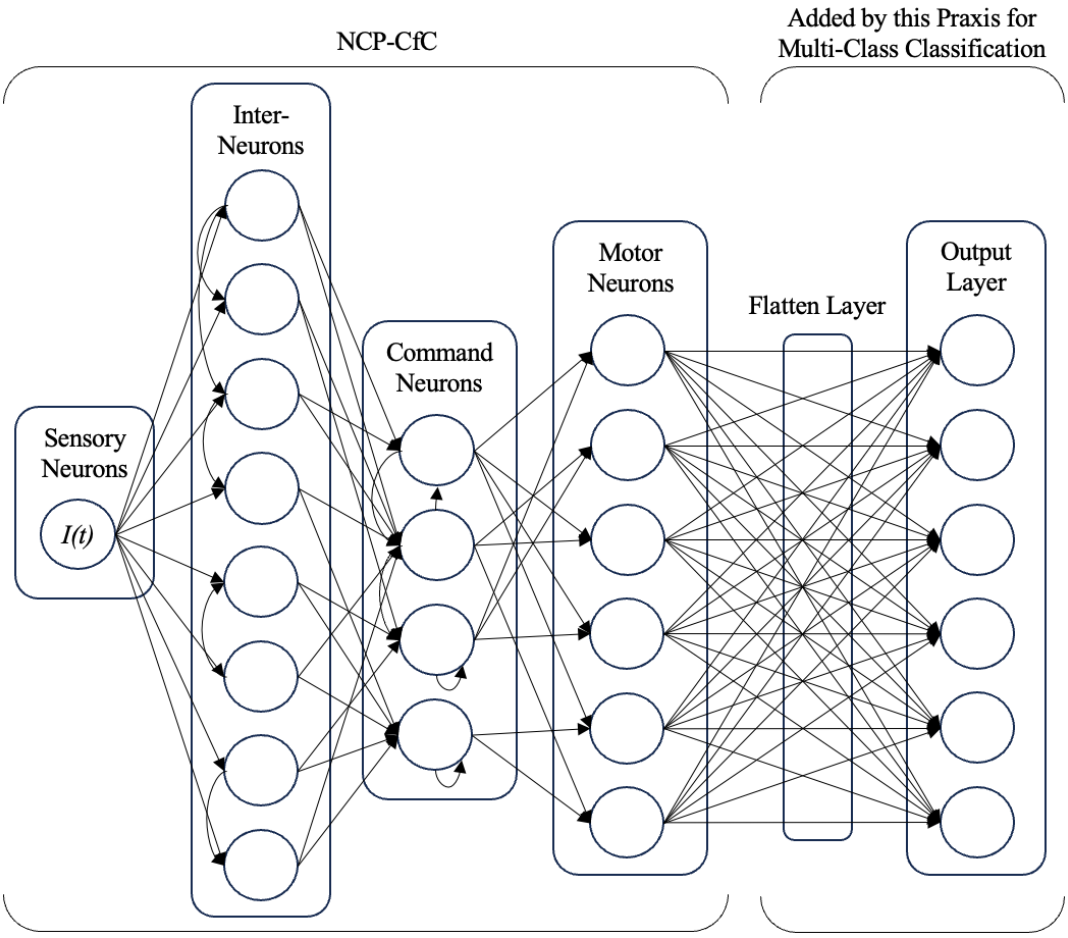

Figure 4-11: Optimal NCP-CfC Architecture With 18 Neurons

Section 4.3.8 described the transformation of features into multidimensional sine and cosine waveforms. As a result, a flatten layer and dense output layer were integrated into the NCP-CfC architecture, a novel approach not found in the existing literature. This additional process was critical for allowing linear vector inputs to be fed into the output layer since they enabled the NCP-CfC to conduct multi-class classification. Similarly, both the ANNs and RNNs traditionally required inputs in a flattened, linear format for processing and were thus given flatten and dense output layers.

4.5 Neural Network Model Results

This section presents the results from testing ANNs, RNNs, and LNNs in classifying IoT-H malware before and after undergoing the Liquid NeurIoTic cyclical waveform conversion. The findings demonstrate the impact of this conversion on each NN model, which underscores the substantial enhancement in the dataset's utility for developing NNs, especially for LNNs. These results were critical for evaluating Hypothesis 1 (i.e., converting IoT-H network traffic features into cyclical waveforms and layering them for input enables time-series representation and enhances their utility). Additionally, the results provide a comprehensive comparison of key performance metrics across all tested NN models to evaluate Hypothesis 2 (i.e., an LNN can be developed to detect and classify attacks in IoT-H networks) and Hypothesis 3 (i.e., an LNN outperforms ANN and RNN and models in IoT-H malware classification).

4.5.1 Artificial Neural Network Results

Optimal results were obtained using 30 neurons across 42 epochs, with a learning rate of 0.001 and a batch size of 128. Table 4-7 details the performance of the top ANN models developed before and after Liquid NeurIoTic's waveform conversion. See Table A1 in Appendix A for a list of the top 25 performing ANNs.

Table 4-7: Top ANNs Before and After Liquid NeurIoTic Waveform Conversion

Model	Neu-rons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Params	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
ANN	30	0.001	128	42	1910.39	6.63	1,416	5.53	0.0281	0.9918	0.9978	0.9985	0.9921	0.9812	0.9812	After
ANN	30	0.01	512	20	54.81	7.12	1,416	5.53	0.0995	0.9865	0.9964	0.9974	0.9871	0.9691	0.9691	Before

The waveform conversion led to a 62.13% improvement in loss and a 0.12% to 2.05% increase in performance in every metric category. These findings suggest that performance enhancements are associated with increased neuron count and the Liquid NeurIoTic waveform conversion process. Hence, adding complexity to the network architecture and the data transformation process significantly boosts an NN's ability to recognize and utilize the intricate patterns and features inherent in the data.

4.5.2 Recurrent Neural Network Results

A notable divergence from the trends seen in ANNs as RNN optimal performance was achieved through a balanced number of neurons—the best-performing model had 20 neurons over 29 epochs, a learning rate of 0.01, and a batch size of 128. This nuanced approach can be attributed to the RNNs' unique memory component, which experiences information loss over time if overloaded with excessive neurons. Furthermore, while RNNs require fewer neurons than ANNs, RNNs possess more than twice as many parameters and are significantly larger than ANNs, which can be attributed to the hidden state of RNNs that adds an extra layer of complexity. Table 4-8 presents the performance outcomes of the best RNN models before and after undergoing the Liquid NeurIoTic waveform conversion process. See Table A2 in Appendix A for the top 25 performing RNNs.

Table 4-8: Top RNNs Before and After Liquid NeurIoTic Waveform Conversion

Model	Neu-rons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Params	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
RNN	20	0.01	128	29	755.40	9.68	2,866	11.2	0.0447	0.9905	0.9955	0.9985	0.9908	0.9781	0.9781	After
RNN	10	0.01	128	17	393.50	9.38	1,336	5.22	0.0360	0.9885	0.9971	0.9954	0.9886	0.9737	0.9738	Before

4.5.3 Liquid Neural Network Results

The optimal performance of the NCP-CfC was achieved after the Liquid NeurIoTic waveform conversion process by applying a balanced number of neurons. Utilizing 18 neurons over 15 epochs, a learning rate of 0.01, and a batch size of 512, this NCP-CfC saw a 42.31% improvement in loss and increased performance in every category. Additionally, it utilized 10% fewer neurons, 57.14% fewer epochs, and 11.23% fewer trainable parameters. Moreover, it was 11.29% smaller in size, which resulted in improvements to training speed by 85.18% compared to the NCP-CfCs applied to the non-waveform data.

These findings suggest that performance enhancements are associated with the waveform conversion process and the LNN architecture employed in Liquid NeurIoTic. They underscore that adding complexity to the data transformation process and utilizing LNNs significantly boosts the ability to recognize and utilize the intricate patterns and features inherent in the data. The performance outcomes of the best NCP-CfCs before and after undergoing the Liquid NeurIoTic waveform conversion process are detailed in Table 4-9. See Table A3 in Appendix A for the top 25 performing NCP-CfCs.

Table 4-9: Top NCP-CfCs Before and After Liquid NeurIoTic Conversion

Model	Neu-rons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Params	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
NCP-CfC	18	0.01	512	15	729.58	20.35	1,550	6.05	0.0210	0.9925	0.9980	0.9978	0.9926	0.9828	0.9829	After
NCP-CfC	20	0.001	128	35	4,921.34	20.05	1,746	6.82	0.0364	0.9880	0.9967	0.9970	0.9884	0.9725	0.9726	Before

Like RNNs in their sequential nature, the neuron count in NCP-CfCs must be carefully balanced to prevent the loss of information over time, as the incorporation of

hidden time-constant states adds a layer of complexity not found in ANNs or RNNs. However, adding more neurons to the NCP-CfC architectures did not result in an exponential growth in parameters and size, like RNNs. For example, an 18-neuron NCP-CfC had 45.92% fewer parameters and 45.98% smaller in size than a 20-neuron RNN, despite that RNN having two more neurons. As LNNs were successfully developed to detect and classify IoT-H malware, Hypothesis 2 was accepted.

4.5.4 Superior Results of Liquid Neural Networks

The top performing ANN, RNN, and LNNs developed in this praxis are presented in Table 4-10. These results showcase the superior efficacy of Liquid NeurIoTic's waveform conversion process and LNNs in IoT-H malware classification. For example, the top three performing models were NCP-CfCs that leveraged the Liquid NeurIoTic waveform conversion process, featuring 18, 24, and 27 neurons, respectively. These outperformed all ANN and RNN models in every performance metric category. Given that the NCP-CfC surpassed the performance metrics of both ANN and RNN models in IoT-H malware classification utilizing Liquid NeurIoTic's waveforms, Hypothesis 1 and Hypothesis 3 were accepted.

Table 4-10: Superior Results of Liquid Neural Networks

Model	Neu-rons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Params	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
NCP-CfC	18	0.01	512	15	729.58	20.35	1,550	6.05	0.0210	0.9925	0.9980	0.9978	0.9926	0.9828	0.9829	Yes
NCP-CfC	24	0.01	256	9	930.40	19.53	2,210	8.63	0.0268	0.9919	0.9974	0.9984	0.9921	0.9815	0.9815	Yes
NCP-CfC	27	0.01	256	12	1027.65	20.71	2,622	10.2	0.0234	0.9919	0.9973	0.9986	0.9922	0.9814	0.9814	Yes
ANN	30	0.001	128	42	1910.39	6.63	1,416	5.53	0.0281	0.9918	0.9978	0.9985	0.9921	0.9812	0.9812	Yes
RNN	20	0.01	128	29	755.40	9.68	2,866	11.2	0.0447	0.9905	0.9955	0.9985	0.9908	0.9781	0.9781	Yes

Notably, the NCP-CfC models demonstrated enhanced efficiency by achieving optimal performance with fewer neurons and epochs relative to the ANNs and RNNs. Despite a slight increase in size and parameter count compared to the best-performing ANN equipped with 30 neurons, the leading NCP-CfC model minimized overall resource usage. For example, the top three NCP-CfC models necessitated 40%, 20%, and 10% fewer neurons, respectively, while reducing the required epochs by 64.29%, 71.43%, and 78.57%, which led to a 61.81% reduction in training time. Therefore, this model can cut training expenses, given the substantial resources and expenses typically required for continuous model training.

Compared to the best-performing RNN model equipped with 20 neurons, the 18-neuron NCP-CfC model boasted 45.92% fewer parameters and a 45.98% reduction in size, which resulted in 3.54% quicker training time. This efficiency was achieved despite the NCP-CfC model utilizing 10% fewer neurons. Furthermore, the NCP-CfC model reached its optimal performance in 48.28% fewer epochs and showcased superior efficiency in loss reduction, which marked an improvement of 53.02%.

The only drawback of the 18-neuron NCP-CfC was the average increase of approximately 110% and 200% in testing times relative to the RNNs and ANNs, respectively. However, access to training resources is often more critical, as the substantial costs associated with training are the primary obstacle to advancing to the testing and deployment stages in NN development. For example, the 2024 Artificial Intelligence Index Report from Stanford University highlights that the cost of artificial intelligence (AI) model training is a significant barrier to developing NNs (Maslej et al., 2024). The cost of training advanced models can soar into millions of dollars, making it

prohibitive for non-industry players like universities to deploy models and contribute to cutting-edge AI technologies (Knight, 2023; Smith, 2023). For example, OpenAI's GPT-3 language model cost at least USD 5 million in GPU expenses for each training iteration, and the development of GPT-4 surpassed USD 100 million in training cost (Knight, 2023; Smith, 2023).

4.5.5 Liquid NeurIoTic Compared to Existing Literature

The superiority of Liquid NeurIoTic using NCP-CfC models in IoT-H malware classification was further demonstrated by comparisons with the performances of healthcare IoT malware classification models found in existing literature, as shown in Table 4-11. Furthermore, this praxis enhanced its evaluation framework by adopting more exhaustive metrics to offer a holistic assessment of model performance to demonstrate Liquid NeurIoTic's efficacy in handling imbalanced classes. This approach contrasted with other studies, which have limited their evaluations to conventional metrics that have not explicitly accounted for class imbalance, such as accuracy, precision, recall, F1-score, and AUC-ROC scores. As a result, this praxis included more robust metrics that accounted for heavily imbalanced classes, such as weighted F1-score, Cohen's kappa, and Matthews correlation coefficient (MCC).

Table 4-11: Superior Results of Liquid NeurIoTic Compared to the Existing Literature

Author and Year	Dataset	Model	Classification Type	Epochs	Neurons	Parameters	Size	Metrics (Best)
Vijayakumar et al. (2023)	ECU-IoHT	DNN	Multi-Class	500+	16	Minimal	Minimal	Accuracy: 99%+, Precision: 99.3%, Recall: 96.8%, F1-Score: 90.3%
Ravi et al. (2023b)	WUSTL EHMS 2020; SDN-IoT	CNN-LSTM	Multi-Class	40	75	Likely Very High	Likely Very High	Accuracy: 99%, Precision: 99%, Recall: 99%, ROC-AUC: 99%

Ravi et al. (2023a)	CDMC-2020-IoMT-Malware	CNN-LSTM	Multi-Class	100	1,792	Likely Very High	Likely Very High	Accuracy: 94-95%
Wozniak et al. (2021)	IoT-23	RNN-LSTM	Multi-Class	Not provided	1,114	Likely Very High	Likely Very High	Accuracy: 99.9957%
Anand et al. (2021)	Malimg	CNN-DMA	Multi-Class	20	896	Likely Very High	Likely Very High	Accuracy: 99%, Precision: 79%–98.9% Recall: 79%–98.9%
Akshay Kumaar et al. (2022)	CICIDS2017, 2018; Bell DNS 2021	ImmuneNet (DL)	Multi-Class	15	1,296	825,137	3,253 KB	Accuracy: 99.19%, Precision: 99.22%, Recall: 99.19%, ROC-AUC: 99.2%
Khan et al. (2023)	ToN-IoT	ANN, DT, RF, k-NN	Multi-Class	NA	NA	Likely Very High	Likely Very High	Accuracy: 91%, F1-score: 90%
Abbas et al. (2024)	CICIoT2023	RNN	Multi-Class	30	82	Likely High	Likely High	Accuracy: 96.5% Precision: 96.52% Recall: 96.52% F1-Score: 95.73%
Neto et al. (2023)	CICIoT2023	LR, ANN, AdaBoost, RF, DNN	Multi-Class	NA	NA	Likely Minimal	Likely Minimal	Accuracy: 99.11% Precision: 70.5% Recall: 91% F1-Score: 71.2%
Jony and Arnob (2024)	CICIoT2023	LSTM	Multi-Class	50	NA	Likely High	Likely High	Accuracy: 98.75% F1-Score: 98.59%
Khan and Alkhathami (2024)	CICIoT2023	RF	Multi-Class	NA	NA	NA	NA	Accuracy: 95.55% Precision: 95.56% Recall: 95.55% F1-Score: 95.52%
Liquid NeurIoTic (This Praxis)	CICIoT2023	NCP-CfC (LNN)	Multi-Class	15	18	1,429	5.58 KB	Accuracy: 99.25%, Precision: 99.80%, Recall: 99.78%, F1-Weighted: 99.26%, MCC: 98.29%, Cohen's Kappa: 98.28

4.6 Summary and Conclusion

This chapter highlighted the exceptional performance of LNNs using NCP-CfC models in the multi-class classification of IoT malware, attributing a significant part of their success to the entire Liquid NeurIoTic architecture, such as the transformation of IoT-H network traffic features into sine and cosine waveforms. This transformation enhanced the utility of the data and allowed for the representation of all inputs as time series. As a result, Hypothesis 1 was accepted, positing that converting IoT-H network traffic features into cyclical waveforms, coupled with their strategic layering, facilitated a time-series representation and enhanced their utility.

Furthermore, this chapter discussed the novel Liquid NeurIoTic architecture developed and employed in this praxis, including the novel LNN models using NCP-CfCs, which surpassed traditional ANNs and RNNs in performance and efficiency. By achieving faster training times, utilizing fewer neurons and epochs with a higher learning rate and batch size, this praxis' NCP-CfC architecture set a new benchmark in the field. This advancement supported the acceptance of Hypothesis 2, which posits the feasibility of developing an LNN capable of detecting and classifying attacks in IoT-H networks. Similarly, as the NCP-CfC architecture outperformed the ANNs and RNNs developed in this praxis and custom NN architectures cited in existing literature on healthcare IoT malware classification, Hypothesis 3 was accepted.

Chapter 5—Conclusions

5.1 Discussion

The primary goal of this praxis was to enhance the cyber resiliency of Internet of Things (IoT) devices in healthcare against cyber threats. As a result, this praxis created a liquid neural network (LNN)-based intrusion detection system (IDS) called Liquid NeurIoTic to effectively identify and classify malware in IoT networks in healthcare (IoT-H) traffic. This goal arose from alarming data indicating that cyberattacks constitute the main cause of data breaches in healthcare since 90% of these organizations using IoT devices have reported at least one security incident (Jiang & Bai, 2019; Williams et al., 2020). Consequently, nearly 50% of IoT devices in healthcare have become vulnerable to cyberattacks that compromise patient privacy and expose them to identity theft, financial losses, and potentially fatal risks.

While the research community has made considerable progress in developing neural network (NN) solutions for IoT-H, scalability and resource management challenges persist. For example, the resource-intensive nature of many NN models renders them costly to train and impractical for use in smaller, resource-limited IoT devices. Moreover, research has shown that NN-based IDSs struggle to keep up with the constantly evolving attack vectors.

Therefore, this praxis developed Liquid NeurIoTic, an LNN-based IDS using an 18-neuron closed-form continuous-time (CfC) network with neural circuit policy (NCP) wiring (NCP-CfC) model designed for IoT-H malware detection and classification. The extremely compact footprint, adaptability, and scalability of Liquid NeurIoTic make it an

attractive choice for multi-class classification in IoT-H networks where data patterns are inconsistent and unpredictable.

5.2 Final Conclusions

This praxis accomplished all research objectives presented in Section 1.5, guided by the three research questions and their corresponding hypotheses presented in Section 1.6. As a result, this praxis demonstrated that all three hypotheses were correct. Hence, they were accepted.

Hypothesis 1 posited that converting IoT-H network traffic features into cyclical waveforms and layering them for input enable time-series representation and amplify their utility. This outcome was achieved through a three-step process: 1) applying *z*-score standardization to the dataset's features, 2) multiplying each standardized value by 2π, and 3) applying sine and cosine functions to each augmented value. As a result, two new features were created for each original feature in the dataset: one representing its sine value and the other representing its cosine value. These features were then stacked together to form two dimensions, which enabled the input to be represented as a time series and enhanced its utility for input into the LNN models. This technique to enhance IoT-H malware traffic data has not been previously accomplished in the existing literature.

Hypothesis 2 posited that an LNN can be developed to detect and classify attacks in IoT-H networks. This outcome was achieved by developing a unique LNN architecture using an 18-neuron NCP-CfC model and enabling it for multi-class classification by incorporating a flatten layer and a dense output layer, which has not been previously accomplished in the existing literature. Moreover, Hypothesis 3 posited that LNNs

surpass artificial neural networks (ANNs) and recurrent neural networks (RNNs) in the performance of classifying IoT-H malware. This outcome was accomplished by observing the results of the unique LNN architecture in Liquid NeurIoTic, which achieved the best performance metric scores out of all models tested. Against the best-performing 30-neuron ANN model, Liquid NeurIoTic's NCP-CfC model required 40% fewer neurons and 64% fewer epochs, resulting in a 61.81% reduced training time. Compared to the best-performing 20-neuron RNN model, it required 10% fewer neurons, 48.28% fewer epochs, and 45.92% fewer parameters, with a 45.98% reduction in size, resulting in 3.54% faster training. Additionally, Liquid NeurIoTic stood out in the broader NN research landscape as it used drastically fewer neurons, epochs, parameters, and total size while surpassing models in the existing literature by comparing performance metrics like accuracy, recall, precision, and F1 score.

Overall, the unique LNN architecture developed in this praxis, called Liquid NeurIoTic, delivered superior performance in IoT-H malware classification compared to comparative ANN and RNN models while outperforming other healthcare IoT NN models developed in existing literature. These results demonstrated the feasibility of using LNNs for IoT-H malware classification to enhance the cyber resilience of IoT-H devices.

5.3 Contributions to Body of Knowledge

This praxis makes three main contributions to the existing body of knowledge in the realm of NN-based IoT security in healthcare:

1. This praxis presents a novel approach that transforms IoT network features into sine and cosine waveforms called Liquid NeurIoTic waveform

conversion. The transformation provides a novel time-series representation of network data, which enhances the utility of these features during the training phase of virtually any NN.

2. This praxis develops an LNN model called an NCP-CfC, which is designed for the multi-class classification of IoT-H malware. This model represents the first of its kind, setting a new benchmark in NN-based IDSs for IoT-H security.
3. This praxis develops Liquid NeurIoTic, an LNN-based IDS that identifies and classifies malware in the traffic of IoT-H.

5.4 Recommendations for Future Research

This praxis established the building blocks necessary to develop and implement LNNs, like NCP-CfCs, within a real-time framework. The remarkable flexibility of LNNs, attributed to their *liquid* ability to adapt to evolving data landscapes, presents a promising avenue for future exploration. Future studies can explore their scalability and effectiveness through real-time analyses in IoT-H networks and other contexts. Such research may provide insights into the dynamic adaptability of LNNs post-training.

Additionally, considering the minimal neuron count required by LNNs, such as NCP-CfCs, there is a compelling case for tailoring an AI explainability tool that provides insights into the decision-making processes of the NCP-CfC's individual neurons. This approach can provide a more profound understanding of the LNN's decision-making process, which may further improve performance and reduce testing times by offering a deeper refinement of the model's operational efficiency.

Lastly, in developing Liquid NeurIoTic, this praxis leveraged the CICIoT2023 dataset, which initially comprised 33 distinct cyberattack types. However, to enhance generalization capabilities and focus, this praxis reclassified these attacks into seven broad categories and excluded the brute force and web-based attacks due to insufficient data representation. This praxis also employed a feature importance selection called MI, which removed 26 of 46 features. Future studies can reintegrate the omitted classes to embrace a holistic examination encompassing all 33 unique attacks and the complete set of 46 features while exploring additional datasets that feature new cyberattack vectors not covered in the CICIoT2023 dataset.

References

Abbas, S., Bouazzi, I., Ojo, S., Al Hejaili, A., Sampedro, G. A., Almadhor, A., & Gregus, M. (2024). Evaluating deep learning variants for cyber-attacks detection and multi-class classification in IoT networks. *PeerJ Computer Science*, *10*, e1793. https://doi.org/10.7717/peerj-cs.1793

Adedeji, K. B., Abu-Mahfouz, A. M., & Kurien, A. M. (2023). DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and Challenges. *Journal of Sensor and Actuator Networks*, *12*(4), 51. https://doi.org/10.3390/jsan12040051

Affia, A. O., Finch, H., Jung, W., Samori, I. A., Potter, L., & Palmer, X.-L. (2023). IoT Health Devices: Exploring Security Risks in the Connected Landscape. *IoT*, *4*(2), 150–182. https://doi.org/10.3390/iot4020009

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), e4150. https://doi.org/10.1002/ett.4150

Akshay Kumaar, M., Samiayya, D., Vincent, P. M. D. R., Srinivasan, K., Chang, C.-Y., & Ganesh, H. (2022). A Hybrid Framework for Intrusion Detection in Healthcare Systems Using Deep Learning. *Frontiers in Public Health*, *9*, 824898. https://doi.org/10.3389/fpubh.2021.824898

Al Kalaa, M. O., Balid, W., Refai, H. H., LaSorte, N. J., Seidman, S. J., Bassen, H. I., Silberberg, J. L., & Witters, D. (2017). Characterizing the 2.4 GHz Spectrum in a Hospital Environment: Modeling and Applicability to Coexistence Testing of Medical Devices. *IEEE Transactions on Electromagnetic Compatibility*, *59*(1), 58–66. https://doi.org/10.1109/TEMC.2016.2602083

Alabsi, B., Anbar, M., & Rihan, S. (2023). Conditional Tabular Generative Adversarial Based Intrusion Detection System for Detecting Ddos and Dos Attacks on the Internet of Things Networks. *Sensors*, *23*(12), 5644. https://doi.org/10.3390/s23125644

Alalhareth, M., & Hong, S.-C. (2023). An Improved Mutual Information Feature Selection Technique for Intrusion Detection Systems in the Internet of Medical Things. *Sensors*, *23*(10), Article 10. https://doi.org/10.3390/s23104971

Alani, M. M., & Damiani, E. (2023). XRecon: An Explainable IoT Reconnaissance Attack Detection System Based on Ensemble Learning. *Sensors*, *23*(11), 5298. https://doi.org/10.3390/s23115298

Al-kahtani, M. S., Khan, F., & Taekeun, W. (2022). Application of Internet of Things and Sensors in Healthcare. *Sensors*, *22*(15), 5738. https://doi.org/10.3390/s22155738

Alotaibi, B., & Elleithy, K. (2016). A New MAC Address Spoofing Detection Technique Based on Random Forests. *Sensors*, *16*(3), 281. https://doi.org/10.3390/s16030281

Al-Sarem, M., Saeed, F., Alkhammash, E. H., & Alghamdi, N. S. (2021). An Aggregated Mutual Information Based Feature Selection with Machine Learning Methods for Enhancing IoT Botnet Attack Detection. *Sensors*, *22*(1), 185. https://doi.org/10.3390/s22010185

Anand, A., Rani, S., Anand, D., Aljahdali, H. M., & Kerr, D. (2021). An Efficient CNN-Based Deep Learning Model to Detect Malware Attacks (CNN-DMA) in 5G-IoT Healthcare Applications. *Sensors*, *21*(19), 6346. https://doi.org/10.3390/s21196346

Andrade, C. (2021). Z Scores, Standard Scores, and Composite Test Scores Explained. *Indian Journal of Psychological Medicine*, *43*(6), 555–557. https://doi.org/10.1177/02537176211046525

Argaw, S. T., Troncoso-Pastoriza, J. R., Lacey, D., Florin, M.-V., Calcavecchia, F., Anderson, D., Burleson, W., Vogel, J.-M., O'Leary, C., Eshaya-Chauvin, B., & Flahault, A. (2020). Cybersecurity of Hospitals: Discussing the challenges and working towards mitigating the risks. *BMC Medical Informatics and Decision Making*, *20*(1), 146. https://doi.org/10.1186/s12911-020-01161-7

Auffret, P. (2010). SinFP, unification of active and passive operating system fingerprinting. *Journal in Computer Virology*, *6*(3), 197–205. https://doi.org/10.1007/s11416-008-0107-z

Begli, M., Derakhshan, F., & Karimipour, H. (2019). A Layered Intrusion Detection System for Critical Infrastructure Using Machine Learning. *2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*, 120–124. https://doi.org/10.1109/SEGE.2019.8859950

Belalis, I., Spathoulas, G., & Anagnostopoulos, I. (2023). Modeling Intruder Reconnaissance Behavior through State Diagrams to Support Defensive Deception. *Journal of Cybersecurity and Privacy*, *3*(2), 275–302. https://doi.org/10.3390/jcp3020015

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, *11*(1), 3625. https://doi.org/10.1038/s41467-020-17236-y

Beshah, Y. K., Abebe, S. L., & Melaku, H. M. (2024). Drift Adaptive Online DDoS Attack Detection Framework for IoT System. *Electronics*, *13*(6), 1004. https://doi.org/10.3390/electronics13061004

Bhushan, B., Kumar, A., Agarwal, A. K., Kumar, A., Bhattacharya, P., & Kumar, A. (2023). Towards a Secure and Sustainable Internet of Medical Things (IoMT): Requirements, Design Challenges, Security Techniques, and Future Trends. *Sustainability*, *15*(7), 6177. https://doi.org/10.3390/su15076177

Bhushan, S., Kumar, A., Lone, S. A., Anwar, S., & Gunaime, N. M. (2023). An Efficient Class of Estimators in Stratified Random Sampling with an Application to Real Data. *Axioms*, *12*(6), 576. https://doi.org/10.3390/axioms12060576

Bonardi, A., James, S., & Davison, A. J. (2020). Learning One-Shot Imitation From Humans Without Humans. *IEEE Robotics and Automation Letters*, *5*(2), 3533–3539. https://doi.org/10.1109/LRA.2020.2977835

Borwein, J. M., & Crandall, R. E. (2013). Closed Forms: What They Are and Why We Care. *Notices of the American Mathematical Society*, *60*(01), 50. https://doi.org/10.1090/noti936

Cartwright, A. J. (2023). The elephant in the room: Cybersecurity in healthcare. *Journal of Clinical Monitoring and Computing*. https://doi.org/10.1007/s10877-023-01013-5

Chaganti, R., Mourade, A., Ravi, V., Vemprala, N., Dua, A., & Bhushan, B. (2022). A Particle Swarm Optimization and Deep Learning Approach for Intrusion Detection System in Internet of Medical Things. *Sustainability*, *14*(19), 12828. https://doi.org/10.3390/su141912828

Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., Amini, A., & Rus, D. (2023). Robust flight navigation out of distribution with liquid neural networks. *Science Robotics*, *8*(77), eadc8892. https://doi.org/10.1126/scirobotics.adc8892

Chen, G., & Liu, Z. (2022). Artificial Neural Network-Based Feed-Forward and Feedback Control Design and Convergence Analysis. *Mathematical Problems in Engineering*, *2022*, 1–10. https://doi.org/10.1155/2022/1238020

Chu, J., Liu, X., Zhang, Z., Zhang, Y., & He, M. (2021). A novel method overcoming overfitting of artificial neural network for accurate prediction: Application on thermophysical property of natural gas. *Case Studies in Thermal Engineering*, *28*, 101406. https://doi.org/10.1016/j.csite.2021.101406

Chu, Y., Fei, J., & Hou, S. (2020). Adaptive Global Sliding-Mode Control for Dynamic Systems Using Double Hidden Layer Recurrent Neural Network Structure. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(4), 1297–1309. https://doi.org/10.1109/TNNLS.2019.2919676

Collier, K. (2022, September 8). *Cyberattacks against U.S. hospitals mean higher mortality rates, study finds.* [News]. https://www.nbcnews.com/tech/security/cyberattacks-us-hospitals-mean-higher-mortality-rates-study-finds-rcna46697

Dablain, D., Jacobson, K. N., Bellinger, C., Roberts, M., & Chawla, N. V. (2023). Understanding CNN fragility when learning with imbalanced data. *Machine Learning*. https://doi.org/10.1007/s10994-023-06326-9

Davis, J. (2023, February 2). *Passion botnet cyberattacks hit healthcare, as actors offer threat as DDoS-as-a-service*.

Dobrojevic, M., Zivkovic, M., Chhabra, A., Sani, N. S., Bacanin, N., & Mohd Amin, M. (2023). Addressing Internet of Things security by enhanced sine cosine metaheuristics tuned hybrid machine learning model and results interpretation based on SHAP approach. *PeerJ Computer Science*, *9*, e1405. https://doi.org/10.7717/peerj-cs.1405

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

Faruqui, N., Yousuf, M. A., Whaiduzzaman, M., Azad, A., Alyami, S. A., Liò, P., Kabir, M. A., & Moni, M. A. (2023). SafetyMed: A Novel IoMT Intrusion Detection System Using CNN-LSTM Hybridization. *Electronics*, *12*(17), 3541. https://doi.org/10.3390/electronics12173541

Fujiwara, K., Huang, Y., Hori, K., Nishioji, K., Kobayashi, M., Kamaguchi, M., & Kano, M. (2020). Over- and Under-sampling Approach for Extremely Imbalanced and Small Minority Data Problem in Health Record Analysis. *Frontiers in Public Health*, *8*, 178. https://doi.org/10.3389/fpubh.2020.00178

Gajjar, P., Saxena, A., Acharya, K., Shah, P., Bhatt, C., & Nguyen, T. T. (2023). Liquidt: Stock market analysis using liquid time-constant neural networks. *International Journal of Information Technology*. https://doi.org/10.1007/s41870-023-01506-1

Garcia, R., Munz, T., & Weiskopf, D. (2021). Visual analytics tool for the interpretation of hidden states in recurrent neural networks. *Visual Computing for Industry, Biomedicine, and Art*, *4*(1), 24. https://doi.org/10.1186/s42492-021-00090-0

Ghubaish, A., Salman, T., Zolanvari, M., Unal, D., Al-Ali, A., & Jain, R. (2021). Recent Advances in the Internet-of-Medical-Things (IoMT) Systems Security. *IEEE Internet of Things Journal*, *8*(11), 8707–8718. https://doi.org/10.1109/JIOT.2020.3045653

Gleeson, P., Lung, D., Grosu, R., Hasani, R., & Larson, S. D. (2018). c302: A multiscale framework for modelling the nervous system of *Caenorhabditis elegans*. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *373*(1758), 20170379. https://doi.org/10.1098/rstb.2017.0379

Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*, *28*(5–6), 367–374. https://doi.org/10.1016/j.compbiolchem.2004.09.006

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: An Overview*. https://doi.org/10.48550/ARXIV.2008.05756

Guo, X., & Gao, X. (2022). A SYN Flood Attack Detection Method Based on Hierarchical Multihead Self-Attention Mechanism. *Security and Communication Networks*, *2022*, 1–13. https://doi.org/10.1155/2022/8515836

Han, J., & Pak, W. (2023). High Performance Network Intrusion Detection System Using Two-Stage LSTM and Incremental Created Hybrid Features. *Electronics*, *12*(4), 956. https://doi.org/10.3390/electronics12040956

Han, S.-H., Kim, K. W., Kim, S., & Youn, Y. C. (2018). Artificial Neural Network: Understanding the Basic Concepts without Mathematics. *Dementia and Neurocognitive Disorders*, *17*(3), 83–89. https://doi.org/10.12779/dnd.2018.17.3.83

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., Teschl, G., & Rus, D. (2022). Closed-form continuous-time neural networks. *Nature Machine Intelligence*, *4*(11), 992–1003. https://doi.org/10.1038/s42256-022-00556-7

Hasani, R., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2021). Liquid Time-constant Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(9), 7657–7666. https://doi.org/10.1609/aaai.v35i9.16936

Hasani, R. M., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2018). *Liquid Time-constant Recurrent Neural Networks as Universal Approximators*. https://doi.org/10.48550/ARXIV.1811.00321

Hassan, E., Shams, M. Y., Hikal, N. A., & Elmougy, S. (2023). The effect of choosing optimizer algorithms to improve computer vision tasks: A comparative study. *Multimedia Tools and Applications*, *82*(11), 16591–16633. https://doi.org/10.1007/s11042-022-13820-0

Hassani, H., & Silva, E. (2015). A Kolmogorov-Smirnov Based Test for Comparing the Predictive Accuracy of Two Sets of Forecasts. *Econometrics*, *3*(3), 590–609. https://doi.org/10.3390/econometrics3030590

He, D., Qiao, Q., Gao, Y., Zheng, J., Chan, S., Li, J., & Guizani, N. (2019). Intrusion Detection Based on Stacked Autoencoder for Connected Healthcare Systems. *IEEE Network*, *33*(6), 64–69. https://doi.org/10.1109/MNET.001.1900105

Hernández, D. G., & Samengo, I. (2019). Estimating the Mutual Information between Two Discrete, Asymmetric Variables with Limited Samples. *Entropy*, *21*(6), 623. https://doi.org/10.3390/e21060623

Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. *Scientific Reports*, *12*(1), 5979. https://doi.org/10.1038/s41598-022-09954-8

Hnamte, V., & Hussain, J. (2023). Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach. *Telematics and Informatics Reports*, *11*, 100077. https://doi.org/10.1016/j.teler.2023.100077

Huraj, L., Šimon, M., & Horák, T. (2020). Resistance of IoT Sensors against DDoS Attack in Smart Home Environment. *Sensors*, *20*(18), 5298. https://doi.org/10.3390/s20185298

Ibarra-Esquer, J., González-Navarro, F., Flores-Rios, B., Burtseva, L., & Astorga-Vargas, M. (2017). Tracking the Evolution of the Internet of Things Concept Across Different Application Domains. *Sensors*, *17*(6), 1379. https://doi.org/10.3390/s17061379

Ibrahim, R. F., Abu Al-Haija, Q., & Ahmad, A. (2022). DDoS Attack Prevention for Internet of Thing Devices Using Ethereum Blockchain Technology. *Sensors*, *22*(18), 6806. https://doi.org/10.3390/s22186806

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., & Graepel, T. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, *364*(6443), 859–865. https://doi.org/10.1126/science.aau6249

Javaid, M., & Khan, I. H. (2021). Internet of Things (IoT) enabled healthcare helps to take the challenges of COVID-19 Pandemic. *Journal of Oral Biology and Craniofacial Research*, *11*(2), 209–214. https://doi.org/10.1016/j.jobcr.2021.01.015

Jebb, A. T., Parrigon, S., & Woo, S. E. (2017). Exploratory data analysis as a foundation of inductive research. *Human Resource Management Review*, *27*(2), 265–276. https://doi.org/10.1016/j.hrmr.2016.08.003

Jeuken, G. S., & Käll, L. (2024). Pathway analysis through mutual information. *Bioinformatics*, *40*(1), btad776. https://doi.org/10.1093/bioinformatics/btad776

Jiang, J. (Xuefeng), & Bai, G. (2019). Evaluation of Causes of Protected Health Information Breaches. *JAMA Internal Medicine*, *179*(2), 265. https://doi.org/10.1001/jamainternmed.2018.5295

Jony, A. I., & Arnob, A. K. B. (2024). A long short-term memory based approach for detecting cyber attacks in IoT using CIC-IoT2023 dataset. *Journal of Edge Computing*. https://doi.org/10.55056/jec.648

Kelly, J. T., Campbell, K. L., Gong, E., & Scuffham, P. (2020). The Internet of Things: Impact and Implications for Health Care Delivery. *Journal of Medical Internet Research*, *22*(11), e20135. https://doi.org/10.2196/20135

Kerkhof, M., Wu, L., Perin, G., & Picek, S. (2023). No (good) loss no gain: Systematic evaluation of loss functions in deep learning-based side-channel analysis. *Journal of Cryptographic Engineering*, *13*(3), 311–324. https://doi.org/10.1007/s13389-023-00320-6

Khan, F., Al-Atawi, A. A., Alomari, A., Alsirhani, A., Alshahrani, M. M., Khan, J., & Lee, Y. (2022). Development of a Model for Spoofing Attacks in Internet of Things. *Mathematics*, *10*(19), 3686. https://doi.org/10.3390/math10193686

Khan, F., Jan, M. A., Alturki, R., Alshehri, M. D., Shah, S. T., & Rehman, A. U. (2023). A Secure Ensemble Learning-Based Fog-Cloud Approach for Cyberattack Detection in IoMT. *IEEE Transactions on Industrial Informatics*, *19*(10), 10125–10132. https://doi.org/10.1109/TII.2022.3231424

Khan, M. M., & Alkhathami, M. (2024). Anomaly detection in IoT-based healthcare: Machine learning for enhanced security. *Scientific Reports*, *14*(1), 5872. https://doi.org/10.1038/s41598-024-56126-x

Knight, W. (2023, April 17). OpenAI's CEO Says the Age of Giant AI Models Is Already Over. *Wired*. https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/

Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and Other Botnets. *Computer*, *50*(7), 80–84. https://doi.org/10.1109/MC.2017.201

Kumar, P., Gupta, G. P., & Tripathi, R. (2021). An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks. *Computer Communications*, *166*, 110–124. https://doi.org/10.1016/j.comcom.2020.12.003

Kumari, K., & Mrunalini, M. (2022). Detecting Denial of Service attacks using machine learning algorithms. *Journal of Big Data*, *9*(1), 56. https://doi.org/10.1186/s40537-022-00616-0

Laštovička, M., Husák, M., Velan, P., Jirsík, T., & Čeleda, P. (2023). Passive operating system fingerprinting revisited: Evaluation and current challenges. *Computer Networks*, *229*, 109782. https://doi.org/10.1016/j.comnet.2023.109782

Lechner, M., Hasani, R., Amini, A., Henzinger, T. A., Rus, D., & Grosu, R. (2020). Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, *2*(10), 642–652. https://doi.org/10.1038/s42256-020-00237-3

Lechner, M., Hasani, R., Zimmer, M., Henzinger, T. A., & Grosu, R. (2019). Designing Worm-inspired Neural Networks for Interpretable Robotic Control. *2019 International Conference on Robotics and Automation (ICRA)*, 87–94. https://doi.org/10.1109/ICRA.2019.8793840

Liu, V., Musen, M. A., & Chou, T. (2015). Data Breaches of Protected Health Information in the United States. *JAMA*, *313*(14), 1471. https://doi.org/10.1001/jama.2015.2252

Loy-Benitez, J., Tariq, S., Nguyen, H. T., Safder, U., Nam, K., & Yoo, C. (2022). Neural circuit policies-based temporal flexible soft-sensor modeling of subway PM2.5 with applications on indoor air quality management. *Building and Environment*, *207*, 108537. https://doi.org/10.1016/j.buildenv.2021.108537

Lu, M.-C., Huang, Q.-X., Chiu, M.-Y., Tsai, Y.-C., & Sun, H.-M. (2022). PSPS: A Step toward Tamper Resistance against Physical Computer Intrusion. *Sensors*, *22*(5), 1882. https://doi.org/10.3390/s22051882

Macedo, F., Valadas, R., Carrasquinha, E., Oliveira, M. R., & Pacheco, A. (2022). Feature selection using Decomposed Mutual Information Maximization. *Neurocomputing*, *513*, 215–232. https://doi.org/10.1016/j.neucom.2022.09.101

Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R. P., & Ni, W. (2019). Anatomy of Threats to the Internet of Things. *IEEE Communications Surveys & Tutorials*, *21*(2), 1636–1675. https://doi.org/10.1109/COMST.2018.2874978

Malasri, K. & Lan Wang. (2009). Securing wireless implantable devices for healthcare: Ideas and challenges. *IEEE Communications Magazine*, *47*(7), 74–80. https://doi.org/10.1109/MCOM.2009.5183475

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access*, *9*, 22351–22370. https://doi.org/10.1109/ACCESS.2021.3056614

Maslej, N., Fattorini, L., Perrault, R., Parli, V., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., & Clark, J. (2024). *The AI Index 2024 Annual Report*. Stanford University. https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf

Mazurczyk, W., & Caviglione, L. (2021). Cyber reconnaissance techniques. *Communications of the ACM*, *64*(3), 86–95. https://doi.org/10.1145/3418293

Mejía-Granda, C. M., Fernández-Alemán, J. L., Carrillo-de-Gea, J. M., & García-Berná, J. A. (2023). Security vulnerabilities in healthcare: An analysis of medical devices and software. *Medical & Biological Engineering & Computing*. https://doi.org/10.1007/s11517-023-02912-0

Meneghello, F., Calore, M., Zucchetto, D., Polese, M., & Zanella, A. (2019). IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal*, *6*(5), 8182–8201. https://doi.org/10.1109/JIOT.2019.2935189

Metongnon, L., & Sadre, R. (2018). Beyond Telnet: Prevalence of IoT Protocols in Telescope and Honeypot Measurements. *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*, 21–26. https://doi.org/10.1145/3229598.3229604

Miao, H., Wu, H., & Xue, H. (2014). Generalized Ordinary Differential Equation Models. *Journal of the American Statistical Association*, *109*(508), 1672–1682. https://doi.org/10.1080/01621459.2014.957287

Mohapatro, M., & Snigdh, I. (2021). An Experimental Study of Distributed Denial of Service and Sink Hole Attacks on IoT based Healthcare Applications. *Wireless Personal Communications*, *121*(1), 707–724. https://doi.org/10.1007/s11277-021-08657-z

Moon, S. (2021). ReLU Network with Bounded Width Is a Universal Approximator in View of an Approximate Identity. *Applied Sciences*, *11*(1), 427. https://doi.org/10.3390/app11010427

Namasudra, S., Sharma, P., Crespo, R. G., & Shanmuganathan, V. (2023). Blockchain-Based Medical Certificate Generation and Verification for IoT-Based Healthcare Systems. *IEEE Consumer Electronics Magazine*, *12*(2), 83–93. https://doi.org/10.1109/MCE.2021.3140048

Natarajan, R., Lokesh, G. H., Flammini, F., Premkumar, A., Venkatesan, V. K., & Gupta, S. K. (2023). A Novel Framework on Security and Energy Enhancement Based on Internet of Medical Things for Healthcare 5.0. *Infrastructures*, *8*(2), 22. https://doi.org/10.3390/infrastructures8020022

Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, *1*(3), 133–143. https://doi.org/10.1038/s42256-019-0025-4

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors*, *23*(13), 5941. https://doi.org/10.3390/s23135941

Newaz, A. I., Sikder, A. K., Rahman, M. A., & Uluagac, A. S. (2019). HealthGuard: A Machine Learning-Based Security Framework for Smart Healthcare Systems. *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 389–396. https://doi.org/10.1109/SNAMS.2019.8931716

Nguyen, T. D., Shih, M.-H., Srivastava, D., Tirthapura, S., & Xu, B. (2021). Stratified random sampling from streaming and stored data. *Distributed and Parallel Databases*, *39*(3), 665–710. https://doi.org/10.1007/s10619-020-07315-w

Nielsen, M. H., Yeh, C.-Y., Shen, M., & Médard, M. (2023). Blockage Prediction in Directional mmWave Links Using Liquid Time Constant Network. *2023 48th International Conference on Infrared, Millimeter, and Terahertz Waves (IRMMW-THz)*, 1–2. https://doi.org/10.1109/IRMMW-THz57677.2023.10299092

Nigrin, D. J. (2014). When “Hacktivists” Target Your Hospital. *New England Journal of Medicine*, *371*(5), 393–395. https://doi.org/10.1056/NEJMp1407326

Ostmeyer, J., & Cowell, L. (2019). Machine learning on sequential data using a recurrent weighted average. *Neurocomputing*, *331*, 281–288. https://doi.org/10.1016/j.neucom.2018.11.066

Pal Singh, A., & Deep Singh, M. (2014). Analysis of Host-Based and Network-Based Intrusion Detection System. *International Journal of Computer Network and Information Security*, *6*(8), 41–47. https://doi.org/10.5815/ijcnis.2014.08.06

Parhi, R., & Nowak, R. D. (2020). The Role of Neural Network Activation Functions. *IEEE Signal Processing Letters*, *27*, 1779–1783. https://doi.org/10.1109/LSP.2020.3027517

Park, D., Kim, S., Kwon, H., Shin, D., & Shin, D. (2021). Host-Based Intrusion Detection Model Using Siamese Network. *IEEE Access*, *9*, 76614–76623. https://doi.org/10.1109/ACCESS.2021.3082160

Petrosyan, A. (2023). *Annual number of Internet of Things (IoT) malware attacks worldwide from 2018 to 2022*. Statista. https://www.statista.com/statistics/1377569/worldwide-annual-internet-of-things-attacks/

Prasetiyowati, M. I., Maulidevi, N. U., & Surendro, K. (2021). Determining threshold value on information gain feature selection to increase speed and prediction accuracy of random forest. *Journal of Big Data*, *8*(1), 84. https://doi.org/10.1186/s40537-021-00472-4

Qiu, W., Zhang, W., Wang, G., Guo, Z., Zhao, J., & Ma, K. (2023). Combined wind speed forecasting model based on secondary decomposition and quantile regression closed-form continuous-time neural network. *International Journal of Green Energy*, 1–22. https://doi.org/10.1080/15435075.2023.2269443

Quan, H. D., Khai, H. D., & Huynh, H. T. (2022). *ODE Based Neural Network and Random Forest in the Quality Assessment of Hand Hygiene Process* (1.0) [Computer software]. Code Ocean. https://doi.org/10.24433/CO.6786224.V1

Raghupathi, W., Raghupathi, V., & Saharia, A. (2023). Analyzing Health Data Breaches: A Visual Analytics Approach. *AppliedMath*, *3*(1), 175–199. https://doi.org/10.3390/appliedmath3010011

Rahmani, A. M., Bayramov, S., & Kiani Kalejahi, B. (2022). Internet of Things Applications: Opportunities and Threats. *Wireless Personal Communications*, *122*(1), 451–476. https://doi.org/10.1007/s11277-021-08907-0

Ramesh, P., Bhaskari, D. L., & -, CH. S. (2010). A Comprehensive Analysis of Spoofing. *International Journal of Advanced Computer Science and Applications*, *1*(6). https://doi.org/10.14569/IJACSA.2010.010623

Rathee, G., Maheswar, R., Sehar, S., & Bavirisetti, D. P. (2023). Towards reliable IoT communication and robust security: Investigating trusted schemes in the internet of medical things using blockchain. *Scientific Reports*, *13*(1), 20671. https://doi.org/10.1038/s41598-023-47989-7

Ravi, V., Pham, T. D., & Alazab, M. (2023a). Attention-Based Multidimensional Deep Learning Approach for Cross-Architecture IoMT Malware Detection and Classification in Healthcare Cyber-Physical Systems. *IEEE Transactions on Computational Social Systems*, *10*(4), 1597–1606. https://doi.org/10.1109/TCSS.2022.3198123

Ravi, V., Pham, T. D., & Alazab, M. (2023b). Deep Learning-Based Network Intrusion Detection System for Internet of Medical Things. *IEEE Internet of Things Magazine*, *6*(2), 50–54. https://doi.org/10.1109/IOTM.001.2300021

Razzaq, W., & Hongwei, M. (2023). Neural Circuit Policies Imposing Visual Perceptual Autonomy. *Neural Processing Letters*, *55*(7), 9101–9116. https://doi.org/10.1007/s11063-023-11194-4

Rejeb, A., Rejeb, K., Treiblmaier, H., Appolloni, A., Alghamdi, S., Alhasawi, Y., & Iranmanesh, M. (2023). The Internet of Things (IoT) in healthcare: Taking stock and moving forward. *Internet of Things*, *22*, 100721. https://doi.org/10.1016/j.iot.2023.100721

Rezk, N. M., Purnaprajna, M., Nordstrom, T., & Ul-Abdin, Z. (2020). Recurrent Neural Networks: An Embedded Computing Perspective. *IEEE Access*, *8*, 57967–57996. https://doi.org/10.1109/ACCESS.2020.2982416

Rihan, S. D. A., Anbar, M., & Alabsi, B. A. (2023). Approach for Detecting Attacks on IoT Networks Based on Ensemble Feature Selection and Deep Learning Models. *Sensors*, *23*(17), 7342. https://doi.org/10.3390/s23177342

Riskhan, B., Safuan, H. A. J., Hussain, K., Elnour, A. A. H., Abdelmaboud, A., Khan, F., & Kundi, M. (2023). An Adaptive Distributed Denial of Service Attack Prevention Technique in a Distributed Environment. *Sensors*, *23*(14), 6574. https://doi.org/10.3390/s23146574

Rohner, N. (2023, February 14). *Russian Hacktivist Group KillNet Hits U.S. Hospitals with DDoS Attacks*. https://blogs.blackberry.com/en/2023/02/killnet-hits-us-hospitals-with-ddos-attacks

Sadaiyandi, J., Arumugam, P., Sangaiah, A. K., & Zhang, C. (2023). Stratified Sampling-Based Deep Learning Approach to Increase Prediction Accuracy of Unbalanced Dataset. *Electronics*, *12*(21), 4423. https://doi.org/10.3390/electronics12214423

Sadhu, P. K., Yanambaka, V. P., Abdelgawad, A., & Yelamarthi, K. (2022). Prospect of Internet of Medical Things: A Review on Security Requirements and Solutions. *Sensors*, *22*(15), 5517. https://doi.org/10.3390/s22155517

Seh, A. H., Zarour, M., Alenezi, M., Sarkar, A. K., Agrawal, A., Kumar, R., & Ahmad Khan, R. (2020). Healthcare Data Breaches: Insights and Implications. *Healthcare*, *8*(2), Article 2. https://doi.org/10.3390/healthcare8020133

Seok, J., & Seon Kang, Y. (2015). Mutual Information between Discrete Variables with Many Categories using Recursive Adaptive Partitioning. *Scientific Reports*, *5*(1), 10981. https://doi.org/10.1038/srep10981

Shamsi, Z., Nandwani, A., Leonard, D., & Loguinov, D. (2016). Hershel: Single-Packet OS Fingerprinting. *IEEE/ACM Transactions on Networking*, *24*(4), 2196–2209. https://doi.org/10.1109/TNET.2015.2447492

Shang, H., & Hu, B.-J. (2022). CALNet: LiDAR-Camera Online Calibration With Channel Attention and Liquid Time-Constant Network. *2022 26th International Conference on Pattern Recognition (ICPR)*, 5147–5154. https://doi.org/10.1109/ICPR56361.2022.9956145

Shao, Z., Yuan, S., & Wang, Y. (2021). Adaptive online learning for IoT botnet detection. *Information Sciences*, *574*, 84–95. https://doi.org/10.1016/j.ins.2021.05.076

Shen, S.-L., Zhang, N., Zhou, A., & Yin, Z.-Y. (2022). Enhancement of neural networks with an alternative activation function tanhLU. *Expert Systems with Applications*, *199*, 117181. https://doi.org/10.1016/j.eswa.2022.117181

Smith, C. (2023, September 8). What Large Models Cost You – There Is No Free AI Lunch. *Forbes*. https://www.forbes.com/sites/craigsmith/2023/09/08/what-large-models-cost-you--there-is-no-free-ai-lunch/?sh=6a2ea47b4af7

Somasundaram, R., & Thirugnanam, M. (2021). Review of security challenges in healthcare internet of things. *Wireless Networks*, *27*(8), 5503–5509. https://doi.org/10.1007/s11276-020-02340-0

Spathoulas, G., Giachoudis, N., Damiris, G.-P., & Theodoridis, G. (2019). Collaborative Blockchain-Based Detection of Distributed Denial of Service Attacks Based on Internet of Things Botnets. *Future Internet*, *11*(11), 226. https://doi.org/10.3390/fi11110226

Statista Market Insights. (2023, September). *Internet of Things—Worldwide*. https://www.statista.com/outlook/tmo/internet-of-things/worldwide#revenue

Takagi, K. (2020). Principles of Mutual Information Maximization and Energy Minimization Affect the Activation Patterns of Large Scale Networks in the Brain. *Frontiers in Computational Neuroscience*, *13*, 86. https://doi.org/10.3389/fncom.2019.00086

Takahashi, K., Yamamoto, K., Kuchiba, A., & Koyama, T. (2022). Confidence interval for micro-averaged F1 and macro-averaged F1 scores. *Applied Intelligence*, *52*(5), 4961–4972. https://doi.org/10.1007/s10489-021-02635-5

Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things: A Comprehensive Review. *Sensors*, *23*(8), 4117. https://doi.org/10.3390/s23084117

Tripathi, S., Gupta, B., Almomani, A., Mishra, A., & Veluru, S. (2013). Hadoop Based Defense Solution to Handle Distributed Denial of Service (DDoS) Attacks. *Journal of Information Security*, *04*(03), 150–164. https://doi.org/10.4236/jis.2013.43018

Tweneboah-Koduah, S., Skouby, K. E., & Tadayoni, R. (2017). Cyber Security Threats to IoT Applications and Service Domains. *Wireless Personal Communications*, *95*(1), 169–185. https://doi.org/10.1007/s11277-017-4434-6

Tylkin, P., Wang, T.-H., Palko, K., Allen, R., Siu, H. C., Wrafter, D., Seyde, T., Amini, A., & Rus, D. (2022). Interpretable Autonomous Flight Via Compact Visualizable Neural Circuit Policies. *IEEE Robotics and Automation Letters*, *7*(2), 3265–3272. https://doi.org/10.1109/LRA.2022.3146555

Ul, I., Bin, M., Asif, M., & Ullah, R. (2018). DoS/DDoS Detection for E-Healthcare in Internet of Things. *International Journal of Advanced Computer Science and Applications*, *9*(1). https://doi.org/10.14569/IJACSA.2018.090140

Ullah, S., Ahmad, J., Khan, M. A., Alkhammash, E. H., Hadjouni, M., Ghadi, Y. Y., Saeed, F., & Pitropakis, N. (2022). A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering. *Sensors*, *22*(10), 3607. https://doi.org/10.3390/s22103607

Vailshery, L. (2023, July 27). *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030*. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

Vanin, P., Newe, T., Dhirani, L. L., O'Connell, E., O'Shea, D., Lee, B., & Rao, M. (2022). A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning. *Applied Sciences*, *12*(22), 11752. https://doi.org/10.3390/app122211752

Victor, P., Lashkari, A. H., Lu, R., Sasi, T., Xiong, P., & Iqbal, S. (2023). IoT malware: An attribute-based taxonomy, detection mechanisms and challenges. *Peer-to-Peer Networking and Applications*, *16*(3), 1380–1431. https://doi.org/10.1007/s12083-023-01478-w

Vijayakumar, K. P., Pradeep, K., Balasundaram, A., & Prusty, M. R. (2023). Enhanced Cyber Attack Detection Process for Internet of Health Things (IoHT) Devices Using Deep Neural Network. *Processes*, *11*(4), 1072. https://doi.org/10.3390/pr11041072

Wang, A., Chang, W., Chen, S., & Mohaisen, A. (2018). A Data-Driven Study of DDoS Attacks and Their Dynamics. *IEEE Transactions on Dependable and Secure Computing*, 1–1. https://doi.org/10.1109/TDSC.2018.2808344

Wang, H., Zhao, S., Dong, Q., Cui, Y., Chen, Y., Han, J., Xie, L., & Liu, T. (2019). Recognizing Brain States Using Deep Sparse Recurrent Neural Network. *IEEE Transactions on Medical Imaging*, *38*(4), 1058–1068. https://doi.org/10.1109/TMI.2018.2877576

Wang, J., Wang, L., & Wang, R. (2023). A Method of DDoS Attack Detection and Mitigation for the Comprehensive Coordinated Protection of SDN Controllers. *Entropy*, *25*(8), 1210. https://doi.org/10.3390/e25081210

Wazzan, M., Algazzawi, D., Bamasaq, O., Albeshri, A., & Cheng, L. (2021). Internet of Things Botnet Detection Approaches: Analysis and Recommendations for Future Research. *Applied Sciences*, *11*(12), 5713. https://doi.org/10.3390/app11125713

Wei, G.-W. (2019). Protein structure prediction beyond AlphaFold. *Nature Machine Intelligence*, *1*(8), 336–337. https://doi.org/10.1038/s42256-019-0086-4

Weinger, B., Kim, J., Sim, A., Nakashima, M., Moustafa, N., & Wu, K. J. (2022). Enhancing IoT anomaly detection performance for federated learning. *Digital Communications and Networks*, *8*(3), 314–323. https://doi.org/10.1016/j.dcan.2022.02.007

Williams, C. M., Chaturvedi, R., & Chakravarthy, K. (2020). Cybersecurity Risks in a Pandemic. *Journal of Medical Internet Research*, *22*(9), e23692. https://doi.org/10.2196/23692

Wozniak, M., Silka, J., Wieczorek, M., & Alrashoud, M. (2021). Recurrent Neural Network Model for IoT and Networking Malware Threat Detection. *IEEE Transactions on Industrial Informatics*, *17*(8), 5583–5594. https://doi.org/10.1109/TII.2020.3021689

Xiong, Y., Dong, S., Liu, R., Shi, F., & Jing, X. (2023). IoT network traffic classification: A deep learning method with Fourier transform-assisted hyperparameter optimization. *Frontiers in Physics*, *11*, 1273862. https://doi.org/10.3389/fphy.2023.1273862

Yaacoub, J.-P. A., Noura, M., Noura, H. N., Salman, O., Yaacoub, E., Couturier, R., & Chehab, A. (2020). Securing internet of medical things systems: Limitations, issues and recommendations. *Future Generation Computer Systems*, *105*, 581–606. https://doi.org/10.1016/j.future.2019.12.028

Yilmaz, A. E., & Demirhan, H. (2023). Weighted kappa measures for ordinal multi-class classification performance. *Applied Soft Computing*, *134*, 110020. https://doi.org/10.1016/j.asoc.2023.110020

Yin, H., Zhou, Y., Cao, L., & Xu, Y. (2021). Channel Prediction with Liquid Time-Constant Networks: An Online and Adaptive Approach. *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 1–6. https://doi.org/10.1109/VTC2021-Fall52928.2021.9625323

Young, A. L., Van Den Boom, W., Schroeder, R. A., Krishnamoorthy, V., Raghunathan, K., Wu, H.-T., & Dunson, D. B. (2023). Mutual information: Measuring nonlinear dependence in longitudinal epidemiological data. *PLOS ONE*, *18*(4), e0284904. https://doi.org/10.1371/journal.pone.0284904

Yu, E. (2023, April 19). *These medical IoT devices carry the biggest security risks* [Tech News]. https://www.zdnet.com/article/these-medical-iot-devices-carry-biggest-security-risks/

Zhang, T., Xu, C., Shen, J., Kuang, X., & Grieco, L. A. (2023). How to Disturb Network Reconnaissance: A Moving Target Defense Approach Based on Deep Reinforcement Learning. *IEEE Transactions on Information Forensics and Security*, *18*, 5735–5748. https://doi.org/10.1109/TIFS.2023.3314219

Zhang, X., Upton, O., Beebe, N. L., & Choo, K.-K. R. (2020). IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers. *Forensic Science International: Digital Investigation*, *32*, 300926. https://doi.org/10.1016/j.fsidi.2020.300926

Appendix A

Table A1: Top 25 ANN Results

Model	Neurons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Param	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
ANN	30	0.001	128	42	1,910.39	6.63	1,416	5.53	0.0281	0.9918	0.9978	0.9985	0.9921	0.9812	0.9812	Yes
ANN	30	0.001	512	50	501.43	6.91	1,416	5.53	0.0357	0.9915	0.9977	0.9985	0.9919	0.9806	0.9806	Yes
ANN	30	0.01	256	12	55.03	6.77	1,416	5.53	0.0413	0.9913	0.9977	0.9980	0.9916	0.9802	0.9802	Yes
ANN	20	0.001	512	50	572.19	6.81	946	3.7	0.0366	0.9907	0.9972	0.9987	0.9911	0.9787	0.9787	Yes
ANN	30	0.001	256	31	145.80	6.85	1,416	5.53	0.0403	0.9906	0.9974	0.9984	0.9909	0.9784	0.9784	Yes
ANN	20	0.01	128	11	91.30	7.05	946	3.7	0.0486	0.9904	0.9967	0.9985	0.9909	0.9780	0.9781	Yes
ANN	20	0.001	128	50	462.24	6.75	946	3.7	0.0344	0.9903	0.9967	0.9987	0.9908	0.9778	0.9779	Yes
ANN	20	0.01	512	25	65.06	6.73	946	3.7	0.0344	0.9902	0.9961	0.9980	0.9908	0.9775	0.9775	Yes
ANN	20	0.001	256	36	158.94	6.41	946	3.7	0.0572	0.9894	0.9970	0.9983	0.9898	0.9757	0.9757	Yes
ANN	30	0.01	128	27	1,212.97	6.86	1,416	5.53	0.0511	0.9890	0.9970	0.9968	0.9893	0.9747	0.9748	Yes
ANN	10	0.01	256	19	78.49	6.44	476	1.86	0.0464	0.9882	0.9964	0.9968	0.9886	0.9730	0.9730	Yes
ANN	10	0.01	128	16	122.13	6.33	476	1.86	0.0592	0.9880	0.9968	0.9973	0.9886	0.9726	0.9726	Yes
ANN	20	0.01	256	13	245.30	6.71	946	3.7	0.0569	0.9877	0.9959	0.9971	0.9884	0.9719	0.9719	Yes
ANN	10	0.01	512	30	68.00	6.60	476	1.86	0.0813	0.9875	0.9974	0.9953	0.9876	0.9713	0.9714	Yes
ANN	10	0.001	128	40	306.91	6.51	476	1.86	0.0492	0.9870	0.9959	0.9966	0.9875	0.9703	0.9703	Yes
ANN	30	0.01	512	20	54.81	7.12	1,416	5.53	0.0995	0.9865	0.9964	0.9974	0.9871	0.9691	0.9691	No
ANN	10	0.001	256	50	210.01	6.93	476	1.86	0.0802	0.9860	0.9966	0.9980	0.9868	0.9681	0.9682	Yes
ANN	20	0.001	128	36	309.44	7.02	946	3.7	0.1071	0.9858	0.9964	0.9973	0.9864	0.9675	0.9675	No
ANN	10	0.01	512	32	69.74	6.99	476	1.86	0.1518	0.9852	0.9969	0.9961	0.9858	0.9662	0.9663	No
ANN	30	0.0001	128	50	805.74	6.95	1,416	5.53	0.1627	0.9850	0.9962	0.9975	0.9857	0.9659	0.9660	Yes
ANN	20	0.0001	128	50	856.23	6.64	946	3.7	0.1493	0.9845	0.9961	0.9969	0.9852	0.9647	0.9648	Yes
ANN	20	0.01	512	31	80.64	7.19	946	3.7	0.1108	0.9836	0.9950	0.9980	0.9846	0.9624	0.9625	No
ANN	10	0.001	512	20	50.79	7.09	476	1.86	0.1774	0.9835	0.9957	0.9965	0.9840	0.9623	0.9624	Yes
ANN	20	0.01	256	28	125.87	7.04	946	3.7	0.0742	0.9827	0.9966	0.9969	0.9832	0.9607	0.9608	No
ANN	10	0.0001	128	50	401.73	6.85	476	1.86	0.2404	0.9815	0.9947	0.9962	0.9821	0.9578	0.9579	Yes

Table A2: Top 25 RNN Results

Model	Neurons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Param	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
RNN	20	0.01	128	29	755.40	9.68	2,866	11.2	0.0447	0.9905	0.9955	0.9985	0.9908	0.9781	0.9781	Yes
RNN	30	0.01	256	21	459.84	10.28	4,596	18	0.4401	0.9898	0.9989	0.9967	0.9901	0.9766	0.9766	Yes
RNN	10	0.01	128	17	393.50	9.38	1,336	5.22	0.0360	0.9885	0.9971	0.9954	0.9886	0.9737	0.9738	No
RNN	10	0.01	128	26	606.37	9.99	1,336	5.22	0.0443	0.9875	0.9945	0.9973	0.9880	0.9714	0.9715	Yes
RNN	30	0.01	256	19	444.56	10.62	4,596	18	0.0580	0.9874	0.9975	0.9957	0.9876	0.9711	0.9712	No
RNN	10	0.01	256	22	306.56	10.00	1,336	5.22	0.0436	0.9873	0.9961	0.9967	0.9876	0.9709	0.9709	No
RNN	20	0.01	128	15	441.32	10.23	2,866	11.2	0.0457	0.9872	0.9958	0.9968	0.9878	0.9707	0.9708	No
RNN	20	0.001	128	50	1429.08	10.56	2,866	11.2	0.1597	0.9864	0.9959	0.9975	0.9871	0.9690	0.9691	No
RNN	30	0.001	256	15	329.74	10.42	4,596	18	0.1556	0.9854	0.9960	0.9968	0.9859	0.9667	0.9667	Yes
RNN	20	0.01	512	23	302.94	10.34	2,866	11.2	0.0626	0.9852	0.9960	0.9967	0.9857	0.9663	0.9663	No
RNN	30	0.01	128	9	317.21	10.67	4,596	18	0.0528	0.9852	0.9958	0.9953	0.9857	0.9662	0.9663	Yes
RNN	20	0.01	256	28	503.03	9.79	2,866	11.2	0.0517	0.9851	0.9960	0.9956	0.9854	0.9658	0.9658	Yes
RNN	30	0.001	512	34	614.49	10.82	4,596	18	0.0508	0.9847	0.9961	0.9968	0.9852	0.9650	0.9651	Yes
RNN	10	0.01	256	20	266.23	9.44	1,336	5.22	0.2554	0.9847	0.9956	0.9964	0.9853	0.9650	0.9650	Yes
RNN	20	0.001	256	40	695.53	9.85	2,866	11.2	0.0483	0.9847	0.9950	0.9973	0.9853	0.9650	0.9650	Yes
RNN	30	0.001	128	14	484.09	10.32	4,596	18	0.0463	0.9847	0.9959	0.9966	0.9853	0.9649	0.9650	Yes
RNN	20	0.01	256	14	256.54	10.15	2,866	11.2	0.4489	0.9841	0.9953	0.9971	0.9849	0.9636	0.9637	No
RNN	30	0.0001	128	50	1771.58	9.79	4,596	18	0.0463	0.9838	0.9954	0.9972	0.9844	0.9630	0.9631	Yes
RNN	10	0.01	512	17	153.38	9.76	1,336	5.22	0.0587	0.9836	0.9950	0.9966	0.9840	0.9624	0.9624	Yes
RNN	20	0.001	128	16	419.94	9.86	2,866	11.2	0.0497	0.9835	0.9951	0.9971	0.9840	0.9622	0.9623	Yes
RNN	30	0.0001	256	50	1077.69	9.92	4,596	18	0.0521	0.9832	0.9956	0.9970	0.9839	0.9617	0.9617	Yes
RNN	10	0.001	128	28	652.45	9.71	1,336	5.22	0.0501	0.9826	0.9954	0.9961	0.9832	0.9603	0.9604	Yes
RNN	20	0.0001	128	50	1,362.37	10.13	2,866	11.2	0.0554	0.9819	0.9955	0.9968	0.9826	0.9587	0.9588	Yes
RNN	20	0.001	512	22	272.64	9.90	2,866	11.2	0.0789	0.9817	0.9945	0.9973	0.9824	0.9582	0.9582	Yes
RNN	20	0.0001	256	50	855.99	9.97	2,866	11.2	0.0550	0.9815	0.9950	0.9970	0.9823	0.9578	0.9579	Yes

Table A3: Top 25 NCP-CfC Results

Model	Neurons	LR	Batch Size	Epoch	Train Time (Sec)	Test Time (Sec)	Param	Size (KB)	Loss	Acc.	Prec.	Recall	F1-Weighted	Cohen's Kappa	MCC	Liquid NeurIoTic Waveform
NCP-CfC	18	0.01	512	15	729.58	20.35	1,550	6.05	0.0210	0.9925	0.9980	0.9978	0.9926	0.9828	0.9829	Yes
NCP-CfC	24	0.01	256	9	930.40	19.53	2,210	8.63	0.0268	0.9919	0.9974	0.9984	0.9921	0.9815	0.9815	Yes
NCP-CfC	27	0.01	256	12	1027.65	20.71	2,622	10.2	0.0234	0.9919	0.9973	0.9986	0.9922	0.9814	0.9814	Yes
NCP-CfC	18	0.01	256	20	1537.86	20.00	1,550	6.05	0.0365	0.9882	0.9966	0.9958	0.9884	0.9729	0.9729	Yes
NCP-CfC	20	0.01	256	15	1240.89	20.86	1,746	6.82	0.0354	0.9881	0.9957	0.9968	0.9885	0.9727	0.9727	Yes
NCP-CfC	20	0.001	128	35	4921.34	20.05	1,746	6.82	0.0364	0.9880	0.9967	0.9970	0.9884	0.9725	0.9726	No
NCP-CfC	24	0.001	128	23	3627.28	19.16	2,210	8.63	0.0311	0.9880	0.9953	0.9973	0.9884	0.9725	0.9725	Yes
NCP-CfC	18	0.01	128	20	2685.28	20.78	1,550	6.05	0.0384	0.9879	0.9951	0.9969	0.9882	0.9722	0.9722	Yes
NCP-CfC	30	0.001	128	16	2506.02	22.53	3,090	12.1	0.0380	0.9864	0.9962	0.9969	0.9869	0.9690	0.9690	Yes
NCP-CfC	24	0.001	256	36	3089.81	20.88	2,210	8.63	0.0379	0.9862	0.9965	0.9966	0.9866	0.9684	0.9685	Yes
NCP-CfC	20	0.001	512	42	2164.88	20.27	1,746	6.82	0.0694	0.9859	0.9960	0.9967	0.9863	0.9677	0.9677	Yes
NCP-CfC	18	0.001	128	30	4005.20	20.37	1,550	6.05	0.0399	0.9855	0.9957	0.9968	0.9860	0.9668	0.9669	Yes
NCP-CfC	27	0.001	128	15	2251.96	20.66	2,622	10.2	0.0412	0.9855	0.9955	0.9972	0.9860	0.9668	0.9668	Yes
NCP-CfC	30	0.001	512	43	2802.01	22.46	3,090	12.1	0.0402	0.9855	0.9958	0.9974	0.9861	0.9668	0.9668	Yes
NCP-CfC	24	0.001	512	42	2449.32	19.01	2,210	8.63	0.0398	0.9853	0.9958	0.9973	0.9858	0.9664	0.9665	Yes
NCP-CfC	20	0.001	128	19	2498.92	20.01	1,746	6.82	0.0442	0.9852	0.9958	0.9969	0.9857	0.9661	0.9662	Yes
NCP-CfC	10	0.01	512	17	751.64	19.88	1,246	4.87	0.0641	0.9848	0.9947	0.9967	0.9854	0.9653	0.9653	Yes
NCP-CfC	27	0.001	256	15	1304.47	20.78	2,622	10.2	0.0489	0.9844	0.9952	0.9973	0.9851	0.9644	0.9644	Yes
NCP-CfC	20	0.001	256	26	2061.56	20.65	1,746	6.82	0.0477	0.9844	0.9960	0.9967	0.9850	0.9642	0.9643	Yes
NCP-CfC	30	0.001	256	20	1889.20	22.31	3,090	12.1	0.1682	0.9842	0.9959	0.9970	0.9849	0.9640	0.9641	Yes
NCP-CfC	18	0.001	256	34	2569.37	20.37	1,550	6.05	0.0418	0.9842	0.9956	0.9969	0.9847	0.9640	0.9640	Yes
NCP-CfC	18	0.001	128	21	2804.28	19.64	1,550	1,550	0.0632	0.9838	0.9959	0.9972	0.9843	0.9630	0.9631	No
NCP-CfC	10	0.01	256	31	2154.16	19.80	1,246	4.87	0.0575	0.9834	0.9959	0.9961	0.9840	0.9622	0.9622	Yes
NCP-CfC	30	0.0001	256	50	4864.84	23.35	3,090	12.1	0.0472	0.9833	0.9954	0.9965	0.9839	0.9618	0.9619	Yes
NCP-CfC	30	0.0001	128	46	7426.83	23.26	3,090	12.1	0.0470	0.9832	0.9955	0.9969	0.9839	0.9618	0.9619	Yes