Spring 3-20-2024

Divergent Detection: A Comprehensive Study of DGAS Using FNNS for Original, Noise-Modified, and Linear Recursive Sequences(LRS)

Anthony Rizi

DIVERGENT DETECTION: A COMPREHENSIVE STUDY OF DGAS USING FNNS FOR ORIGINAL, NOISE-MODIFIED, AND LINEAR RECURSIVE SEQUENCES (LRS)

A dissertation submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Cyber Operations

March 20, 2024

By

Anthony Guilliamo Rizi

Dissertation Committee:

Dr. Varghese Vaidyan

Dr. Tom Halverson

Dr. Mark Spanier

Dr. Cory Singleton

DAKOTA STATE UNIVERSITY

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Philosophy degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Anthony Rizi

Dissertation Title:
DIVERGENT DETECTION: A COMPREHENSIVE STUDY OF DGAS USING FNNS FOR ORIGINAL, NOISE-MODIFIED, AND LINEAR RECURSIVE SEQUENCES(LRS)

Graduate Office Verification: DocuSigned by: Abby Chowning F44C8D9E621C417... Date: 04/01/2024

Dissertation Chair/Co-Chair: DocuSigned by: Varghese Vaidyan 7D0D713978DE43F... Date: 04/01/2024
Print Name: Varghese Vaidyan

Dissertation Chair/Co-Chair: _____ Date: _____
Print Name: _____

Committee Member: DocuSigned by: Tom Halverson DE50EF3DC4B5472... Date: 04/01/2024
Print Name: Tom Halverson

Committee Member: DocuSigned by: Mark Spanier 55905BBEC9A7414... Date: 04/01/2024
Print Name: Mark Spanier

Committee Member: DocuSigned by: Dr. Cory Singleton 7A0C1A989DC9432... Date: 04/04/2024
Print Name: Dr. Cory Singleton

Committee Member: _____ Date: _____
Print Name: _____

Submit Form Through Docusign Only
or to Office of Graduate Studies
Dakota State University

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to everyone who has contributed to the completion of this dissertation. Your support and encouragement have been invaluable throughout this academic journey. First and foremost, I extend my deepest appreciation to my chair, Dr. Vaidyan, for his guidance, expertise, and unwavering commitment. His mentorship has been instrumental in shaping the direction of this research, and for Dr. Hamm for initiating and inspiring the exploration of DGA research. I am thankful for the support of the entire DSU faculty, and I especially appreciate my academic committee members, Dr. Haverson and Dr. Spanier, and Dr. Singleton. Their insightful feedback and constructive criticism have significantly enhanced the quality of this work. I want to express my gratitude to my supervisors and colleagues for their encouragement. The camaraderie and support from each of you have not only made this academic pursuit intellectually stimulating but also genuinely enjoyable. I'm deeply grateful to the Amateur Radio Digital Communications Scholar selection committee for the invaluable award. The financial support greatly aided in completing my coursework and dissertation, inspiring my commitment to advancing in cyber, digital communications, and amateur radio. Thank you for this meaningful opportunity. I'm grateful for the friends I made during my first DSU residency for their excellent support and camaraderie as fellow classmates in the program. I appreciate the unwavering support from friends and family throughout my academic journey. Your love and encouragement have been a constant source of strength. A special thanks to my dearest for the sacrifices made over the years. Despite missed weekends, late-night meetings, and early morning submissions, I eagerly anticipate creating countless memories together in the years ahead.

ABSTRACT

This comprehensive research endeavors to address a pressing issue within the realm of cybersecurity—the challenge posed by malicious activities utilizing Domain Generation Algorithms (DGAs). These algorithms, numbering at least 84 traditional malware families as of late 2023, dynamically generate domain names to facilitate nefarious operations while evading conventional detection mechanisms. The study generated over 159,750 domains and studied more than 1.27 million data points. Existing studies have predominantly focused on surface-level aspects of DGAs, including domain lengths, alphanumeric values, and top-level domains (TLDs). In response to this challenge, the research question at the core of this study aims to investigate whether sophisticated classifiers can effectively detect and classify DGA-enabled malware by discerning variations in DGAs, including original DGAs, those modified with injected noise, and a novel approach of modification through Linear Recursive Sequences (LRS).

The chosen methodology for this research adopts a quantitative design, utilizing Python programming and a suite of libraries for efficient data manipulation, machine learning, and visualization. The focal point of the methodology involves training a Feed-forward Neural Network (FNN) using a meticulously curated dataset comprising both original DGAs and their modified counterparts. To facilitate effective classification, the dataset undergoes a detailed segmentation into categories. The FNN architecture, with specific hyperparameters, employs the Adam optimizer, sigmoid activation functions, and three dense layers. Eight features, including Damerau Levenshtein Distance and String Entropy, and others to contribute to the FNN's understanding of the input data.

The research explores the intricacies of neural network comprehension, dataset classification, and feature identification, overcoming these challenges through extensive multi-classification learning processes. The training configuration involves a Learning Rate of

0.0001, 50 epochs, a batch size of 32, and a 80/20% validation split. Rigorous feature selection and engineering, model selection, and hyperparameter adjustment are integral to the methodology. The study reviews five primary DGA datasets Banjori, Dnschanger, Dyre, Gameover, Murofetweekly, presenting detailed insights into their characteristics. The analysis reveals the challenges posed by DGA families with insufficient sample sizes, necessitating a strategic selection process. The FNN's performance is explicitly evaluated on its ability to classify instances into original DGAs, Noise-Modified DGAs, and LRS-Modified DGAs.

In conclusion, this research contributes significantly to cybersecurity by offering a sophisticated approach to DGA detection. The methodology's robustness is examined through potential challenges, and recommendations for addressing these challenges are provided. This research demonstrates the effectiveness of the FNN in identifying an average of 99.5 percent of noise and LRS DGA modifications. This significant contribution enhances cybersecurity by introducing a sophisticated approach to DGA detection. It underscores the significance of staying abreast of evolving cyber threats and emphasizes the need for proactive cybersecurity measures in the face of continually adapting tactics employed by malicious actors.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another. I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Anthony Guilliamo Rizi

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background of the Problem

In the shadows of the cyber realm, a growing menace lurks in the form Domain Generating Algorithms(DGA)s or automated domain names that are used to support malware command and control connections. As of mid-2021, Ayub A. et al. [1] documented the existence of at least 84 traditional malware families. Existing studies on DGA strings often provide only superficial insights, focusing on general characteristics such as domain lengths, alphanumeric values, and potential Top Level Domains (TLDs). The majority of DGAs analyzed in these studies have been sourced and submitted for research through the DGArchive website.

In 2022, Z. Mu [2] conducted research involving DGAs and employed N-Grams for classification. Mu reported an impressive 95% accuracy in classifying known DGA domains and an 88.5% accuracy in detecting previously unknown domains. His approach involved leveraging numerical and vowel patterns, examining the frequency of repetition or non-repetition of these values. It is crucial to note that Mu's work primarily focuses on botnets, which entail millions of potential DGA domains. The applicability of his research may be limited if DGA domains are used specifically against certain applications.

J. Ahmed et al. [3] carried out research in 2022, focusing on automatically detecting DGA-enabled malware through Software Defined Networking (SDN) and Traffic Behav-

ioral Modeling. Their study spanned 75 days on a school campus, analyzing DNS queries to identify DGA-related domain names. Out of 2.4 billion DNS queries within the campus network, only 589K were found to be related to DGAs. The researchers cross-referenced these potential DGA domains with DGArchive. The research presented the top 10 most used DGA-related domain names, all belonging to four families. Notably, the top two families had 22k to 530k hits, while the remaining families exhibited significantly fewer interactions, averaging only a couple of hundred during the observation period. The research highlights the potential challenge of intermittently active DGAs that may go unnoticed in conventional analyses. In this evolving narrative of threats and innovations, the quest for adaptive detection strategies becomes ever more crucial. However, what if DGAs were modified just enough to evade detection within DGArchive?

1.2 A Brief History of DGAs

1.2.1 Introduction

Domain Generating Algorithms (DGAs) emerged in response to the escalating sophistication of cyber threats, particularly in the realm of malware [4]. These algorithms dynamically generate domain names that malicious software uses to establish communication with command and control servers, evading traditional security measures.

The dissertation covers the domain of Feedforward Neural Networks (FNNs) and their applications, specifically addressing the cybersecurity challenges posed by obfuscated Domain Generating Algorithms (DGAs). FNNs, with their layered structure encompassing input, hidden, and output layers, provide a foundation for learning hierarchical representations. The interconnected neurons, activation functions like sigmoid or ReLU, and the forward propagation principle contribute to their versatility across diverse applications, including pattern recognition, image processing, and natural language understanding. These networks, as universal approximators, play a fundamental role in machine learning

tasks, serving as foundational models for more advanced architectures.

The cybersecurity landscape faces a formidable challenge with the emergence of sophisticated obfuscation techniques within DGAs. The focus of the research is on the utilization of noise and linear recursive sequences in DGAs, introducing complexity that demands advanced analytical approaches. The problem at hand involves identifying and understanding obfuscated DGAs, with a specific emphasis on employing FNNs to discern the underlying linear recursive patterns. The research aims to contribute to cybersecurity practices by providing a robust framework for identifying and analyzing DGAs leveraging linear recursive sequences, enhancing the capabilities of cybersecurity systems to detect and counteract evolving threats.

1.2.2 Early Years

The concept of DGAs dates back to the mid-2000s when security researchers and malware analysts observed a shift in malware tactics. Examples include the Conficker worm (2008), which utilized DGAs to update itself and connect to its command and control infrastructure [5].

The early years of Domain Generating Algorithms (DGAs) marked a significant evolution in cyber threat tactics, showcasing the adaptability of malicious actors. Originating in the mid-2000s, DGAs emerged as a response to traditional static blacklists used for blocking known malicious domains. Cybercriminals sought ways to evade detection and maintain control over their malicious infrastructure. DGAs were ingeniously designed algorithms that dynamically generated a large number of seemingly random domain names. This dynamic generation allowed malware to establish communication with command and control servers without relying on fixed, easily blockable domain names. The early DGAs often utilized pseudo-random techniques, making it challenging for security solutions to predict or block their activities effectively. This shift in strategy marked a cat-and-mouse game between cybersecurity professionals and attackers, as each side continuously adapted

their methods. The early years of DGAs laid the groundwork for more sophisticated and complex algorithms, contributing to the ongoing challenges faced in modern cybersecurity landscapes.

1.2.3 Advancements and Variations

As security measures evolved, so did DGAs. Malware authors introduced more sophisticated algorithms and variations to increase the resilience of their malicious infrastructure. Notable examples include seed-based algorithms, which are a predetermined seed that initiates the generation of seemingly random domains [6]. Pseudo-Random Number Generators (PRNGs) are an alternative common approach, via seeds to create sequences of domain names that appear random, making it difficult for security systems to predict and block them effectively [4]. Additionally, time-based variations introduces time-based elements, leveraging current date or time information in the generation process [7]. Understanding these methods is critical for creating robust cyber defense strategies..

1.2.4 Cat-and-Mouse Game

The battle between cybersecurity professionals and malware authors became a cat-and-mouse game. While security experts developed tools to detect and block DGAs, attackers continually refined their techniques to stay ahead. Some of these tools. Cisco Umbrella, formerly known as OpenDNS, is a cloud-delivered security service that includes DNS-layer security capable of identifying and blocking malicious domains generated by DGAs [8]. FireEye DNS Analytics offers a solution focused on analyzing DNS traffic patterns to detect and mitigate threats associated with DGAs [9]. Security Information and Event Management (SIEM) solutions, such as Splunk and LogRhythm, can be configured to monitor DNS logs and identify anomalous patterns indicative of DGA activity [10], [11]. Additionally, machine learning-based solutions leverage advanced algorithms to analyze DNS traffic and detect deviations associated with DGAs [12]. Employing a combination

of these tools, organizations can bolster their defense against the evolving threat posed by DGA-related activities.

1.2.5 Machine Learning and Behavioral Analysis

In recent years, machine learning and behavioral analysis integration has become pivotal in combating DGAs [13]. Security solutions leverage these technologies to identify patterns and anomalies in domain name generation, enhancing the ability to detect and prevent malicious activities. Machine learning algorithms, including support vector machines (SVM) and deep neural networks, undergo training using extensive datasets incorporating benign and malicious domain names [14]. This training process allows the models to grasp intricate patterns and distinctive characteristics associated with Domain Generation Algorithms (DGAs). Through the analysis of features such as domain length, character frequency, and temporal patterns, these machine learning models can proficiently discern between legitimate and malicious domain names [15].

Behavioral analysis complements machine learning by focusing on the runtime behavior of DGAs within a network. This approach involves monitoring the interactions and communications of dynamically generated domains to identify deviations from normal network behavior. Techniques like anomaly detection and clustering are applied to distinguish malicious activities from legitimate network traffic [16].

By combining machine learning and behavioral analysis, security systems can utilize the adaptability of machine learning models and the contextual insights from behavioral analysis, providing a multi-faceted defense against DGAs. This approach allows security systems to stay ahead of evolving DGA techniques and enhance overall cyber defense resilience.

1.2.6 Current Landscape

In today's cybersecurity landscape, DGAs continue to pose a significant threat. Malware campaigns frequently utilize sophisticated DGAs to establish resilient command and control infrastructures [6].

For example:

- Elevated Sophistication: Malicious actors have heightened the sophistication of DGAs, making it increasingly challenging for conventional security measures to anticipate and counteract their impact effectively.
- **Robust Command and Control Networks:** The integration of advanced DGAs contributes to creating highly robust command and control networks. This adaptability enables cyber adversaries to control compromised systems while evading detection.

Cybersecurity strategies emphasize ongoing vigilance and adaptability to counter the evolving tactics employed by malicious actors using DGAs.

Computer Worm: Conficker, or can be called Downadup or Kido windows based malware. Initially identified in 2008, continues to pose a significant threat in the cybersecurity landscape. This malicious software has demonstrated remarkable persistence over the years. Its multifaceted capabilities include serving as a tool for propagation, forming botnets, executing Command and Control operations through the use of DGAs, delivering payloads, engaging in data theft and espionage activities, and facilitating the dissemination of scareware. The versatility of Conficker makes it a formidable cybersecurity concern, as it can be utilized for a range of malicious activities, from compromising systems and networks to unauthorized data access and even the dissemination of deceptive scare tactics. The ongoing prevalence of Conficker highlights the challenges faced by the cybersecurity community in combating and mitigating evolving threats in the digital realm.

Banking Trojans: Emotet Trojan, a highly sophisticated malware strain, showcases a modular architecture that allows for adaptability and continuous updates to its malicious functionalities. Propagating primarily through deceptive email attachments or links, Emotet employs social engineering tactics to deceive users. Once activated, it delivers additional payloads, stealing sensitive information like credentials and exhibiting lateral movement within networks. It is notorious for its polymorphic nature or technique used to constantly change the code or appearance of the malicious software. Polymorphic malware alters its code without changing its underlying functionality, making it challenging for traditional antivirus solutions that rely on static signatures to detect and block malicious programs. Emotet constantly changes its code to evade traditional detection methods. Utilizing command and control servers via DGAs establishes remote communication, enabling cybercriminals to control and update the malware and gather stolen data. Emotet's evolving tactics make it a formidable threat in the cybersecurity landscape, demanding vigilance and proactive defense measures.

GameOverZeus(GOZ), identified in 2014, is a formidable threat that evolved from a variant of the notorious Zeus banking trojan. Exhibiting a range of sophisticated characteristics, GOZ is adept at spreading through various channels, employing tactics such as malicious email attachments and exploit kits. It establishes a resilient botnet infrastructure, allowing cybercriminals centralized control over compromised systems. GOZ is equipped with keylogging capabilities, enabling the theft of sensitive credentials crucial for online financial transactions. Its use of man-in-the-browser attacks further amplifies its threat, manipulating online banking sessions to facilitate financial fraud. Demonstrating persistence and resilience within infected systems, GOZ employs peer-to-peer communications, reducing dependency on a central server and making it challenging to detect and combat. The dynamic and multifaceted nature of GOZ underscores the importance of robust cybersecurity measures to protect against evolving banking trojans.

1.2.7 DGA Obfuscation

In the cybersecurity landscape, malicious entities' adoption of domain-generating algorithms (DGAs) has emerged as a widespread strategy to establish covert communication with command and control servers. Cyber adversaries deploy DGA obfuscation techniques to elude detection, injecting intricacies into the domain generation process. Seed-based obfuscation, a prominent approach, employs predefined values or seeds to initiate the DGA algorithm, introducing variability and impeding pattern recognition. Complementing this, integrating Pseudo-Random Number Generators (PRNGs) is commonplace, utilizing initial seeds to generate domains with an appearance of randomness, thus adding an extra layer of unpredictability to the malicious infrastructure. Conversely, time-based obfuscation introduces temporal elements, such as the current date, into the domain generation process, resulting in dynamically changing domains. These obfuscation methodologies pose substantial challenges for cybersecurity defenses, necessitating the development of adaptive detection mechanisms. Advanced machine learning algorithms and behavioral analysis play pivotal roles in discerning patterns amidst the intricacies of obfuscation. The persistent evolution of DGA obfuscation tactics underscores the ongoing need for research to proactively address and counteract these evasion techniques.

1.2.8 LRS Concepts

[17] [18] explains Linear recursive sequences are mathematical sequences defined by a linear recurrence relation. The general form of a linear recursive sequence of order k is given by:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k} + b$$

Here, a_n represents the n-th term in the sequence, $c_1, c_2, \ldots, c_k$ are constant coefficients, and b is a constant term. Once the initial terms, $a_0, a_1, \ldots, a_{k-1}$, are specified, the

sequence is completely determined listed in table below.

Key Characteristics:

1. *Linearity:* Linear recursive sequences exhibit linearity because a linear combination of the preceding terms forms each term.

2. *Homogeneous and Inhomogeneous Sequences:* If $b = 0$, the sequence is homogeneous; otherwise, it is inhomogeneous.

3. *Order of the Sequence:* The sequence order is determined by the highest subscript in the recurrence relation, in this case, k.

4. *Initial Conditions:* To uniquely determine the sequence, initial conditions (sequence values for the first k terms) must be provided.

Applications:

Linear recursive sequences find applications in various fields, including computer science, signal processing, and cryptography. In computer science, they are used to model and analyze algorithms. In signal processing, linear recursive sequences are employed in the design of digital filters. Cryptographic algorithms may use linear recursive sequences for generating pseudorandom numbers.

Understanding linear recursive sequences is foundational for solving difference equations and recurrence relations, providing valuable insights into the behavior of sequences over time.

1.3 Difficulties in protecting the cyber domain

Protecting cyber domains against Domain Generating Algorithms (DGAs) poses several challenges for cybersecurity professionals. DGAs are dynamic techniques used by malware to generate domain names dynamically, making it difficult to predict and counteract malicious activities. The following challenges highlight the complexities in defending against DGAs:

1. **Dynamic and Evolving Nature [19]:** DGAs continually evolve to counter existing detection mechanisms. The dynamic nature requires adaptive and real-time detection strategies to keep pace with the changing tactics of malicious actors.

2. **Large Number of Potential Domains [4]:** DGAs can generate a vast number of potential domains, creating a significant search space for security systems. Analyzing and predicting the multitude of potential domains poses a challenge for traditional security solutions.

3. **Algorithm Variability [6]:** Malicious actors employ various obfuscation techniques in DGAs, such as seed-based, time-based, or pseudo-random generation. Each variation introduces unique challenges in understanding the underlying algorithm and developing effective countermeasures.

4. **Fast Flux Networks** Some malware using DGAs utilize fast flux techniques, rapidly changing associated IP addresses. This dynamic nature complicates tracking and blacklisting efforts as the malicious infrastructure constantly shifts.

5. **False Positives [4]:** Overly aggressive detection methods may lead to false positives, misclassifying legitimate domains as malicious. Achieving a balance between accurate detection and minimizing false positives is a constant challenge.

6. **Encryption and Tunnelling [20]:** Malware employing DGAs often uses encrypted communication channels and tunneling techniques. This additional layer of obfuscation makes it challenging to inspect and identify malicious traffic, especially when the communication is encrypted end-to-end.

7. **Resource-Intensive Analysis [4]:** Analyzing a large volume of DNS traffic for potential DGA-generated domains can be resource-intensive. Effective detection mechanisms must be implemented without adversely affecting network performance.

Addressing these challenges requires a comprehensive approach, combining heuristic analysis, machine learning algorithms, and behavioral analysis. Cybersecurity professionals must stay abreast of evolving DGA techniques and continually update defense strategies to mitigate the risks posed by these dynamic and sophisticated threats.

1.4 Machine Learning

Machine Learning (ML) is a pivotal domain within the broader landscape of artificial intelligence, focusing on empowering computers to learn from data and make informed decisions without explicit programming. At its core, machine learning aims to develop algorithms and models capable of recognizing patterns, learning from experiences, and generalizing knowledge to handle novel, unseen data. The essence of machine learning lies in its departure from traditional rule-based programming, shifting towards an approach where algorithms learn and evolve based on the inherent patterns within the data they encounter. Key concepts are listed in Table 1.1.

Key Concepts	Description
Learning from Data	Unlike traditional programming paradigms, machine learning leverages data as the primary source of knowledge. Algorithms are trained on datasets, enabling them to discern patterns and extrapolate insights from the information they are exposed to [21].
Types of Learning	Machine learning encompasses various learning paradigms, including supervised learning, unsupervised learning, and reinforcement learning. Each type addresses specific scenarios, from labeled dataset training to learning through interaction with an environment [22].
Feature Extraction	ML models operate on features, representing the relevant characteristics or variables of the data. Feature extraction involves the selection and transformation of raw data to create a format suitable for learning [23].
Model Training and Evaluation	The training phase involves adjusting the internal parameters of the algorithm to minimize errors and enhance performance. Evaluation on separate datasets gauges the model's ability to generalize to new, unseen data [24].
Applications	Machine learning finds applications across diverse domains, including image and speech recognition, natural language processing, recommendation systems, healthcare diagnostics, and finance for fraud detection [25].

Table 1.1: Key Concepts in Machine Learning

Despite its successes, machine learning encounters challenges such as model interpretability, bias, and ethical considerations. Ongoing research aims to address these issues, enhancing transparency, minimizing bias, and ensuring responsible and fair deployment of machine learning systems.

In conclusion, machine learning emerges as a powerful tool for extracting insights and making predictions from complex datasets. As technology advances, machine learning is poised to play a central role in shaping the future of intelligent systems and applications.

1.4.1 FNN and its Features

A Feedforward Neural Network (FNN) represents a foundational architecture in artificial neural networks, a subset of machine learning inspired by the workings of the human

brain. Often called multilayer perceptrons, FNNs are characterized by layered nodes or neurons without cyclic connections. The architecture typically includes an input layer, one or more hidden layers, and an output layer [26].

Key features define the nature of Feedforward Neural Networks:

The layered structure of FNNs encompasses input, hidden, and output layers, allowing for the learning of intricate hierarchical representations. Neurons within the network are interconnected with weights, and each neuron is associated with a bias term, providing flexibility in decision-making. Activation functions, such as sigmoid or rectified linear unit (ReLU), introduce non-linearity to the neurons, enabling the modeling of complex relationships within the data [24].

The information processing in FNNs follows the principle of forward propagation. During training, the backpropagation algorithm is employed, adjusting weights and biases iteratively to minimize the difference between the predicted and actual outputs [26].

Demonstrating their versatility across diverse applications, one notable characteristic of FNNs is their role as universal approximators. These networks can approximate any continuous function with sufficient neurons and proper training.

In practice, Feedforward Neural Networks find applications in various domains, including pattern recognition, image and speech processing, natural language understanding, and regression tasks. While serving as foundational models, FNNs have paved the way for more advanced architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

The simplicity, interpretability, and effectiveness of Feedforward Neural Networks contribute to their widespread use in machine learning, making them a fundamental tool for various tasks.

1.5 Statement of the Problem

In the landscape of cybersecurity, the emergence of sophisticated obfuscation techniques within Domain Generating Algorithms (DGAs) poses a formidable challenge for threat detection and mitigation. Particularly, the utilization of linear recursive sequences in DGAs introduces a layer of complexity that demands advanced analytical approaches. The problem at hand involves the effective identification and understanding of these obfuscated DGAs, with a specific focus on employing Feedforward Neural Networks (FNNs) to discern the underlying linear recursive patterns.

This research aims to contribute to the advancement of cybersecurity practices by providing a robust framework for the identification and analysis of DGAs that leverage linear recursive sequences. The utilization of FNNs is envisioned as a powerful tool to uncover hidden patterns and relationships within the obfuscated domain generation process, thereby enhancing the capabilities of cybersecurity systems to detect and counteract evolving threats.

The outcomes of this study are anticipated to offer valuable insights into the application of machine learning, specifically FNNs, in the context of cybersecurity, contributing to the development of more adaptive and resilient defense mechanisms against obfuscated DGAs.

1.6 Objectives of the Project

The research aims to achieve two main objectives. Firstly, it seeks to develop and assess the effectiveness of Feedforward Neural Networks (FNNs) in the identification and interpretation of obfuscated algorithms. Specifically, the focus is on detecting and decoding algorithms that are obfuscated through techniques such as introducing noise to strings with character modification and utilizing Linear Recursive Sequences. The choice of these

obfuscation methods is influenced by their distinctive mathematical properties, which pose challenges for conventional detection approaches. Consequently, the research emphasizes the exploration and application of advanced machine learning techniques, particularly FNNs, to address these challenges.

Secondly, the research explores the characteristics of algorithms based on Linear Recursive Sequences used in Domain Generation Algorithms (DGAs). DGAs are often employed in malicious activities, and understanding the new novel properties of introducing algorithms based on linear recursive sequences can contribute to enhanced detection and mitigation strategies. This aspect of the research underscores the importance of investigating the intricacies of linear recursive sequence-based DGAs, shedding light on their patterns and behaviors to bolster the development of more effective cybersecurity measures.

1.7 Dissertation Organization

This dissertation is organized in the following format: Chapter 1 serves as the introduction, providing an overview of Feedforward Neural Networks (FNNs) and their applications, subsequently addressing the cybersecurity challenges posed by obfuscated Domain Generating Algorithms (DGAs). Chapter 2, Literature Review, conducts a thorough literature review, exploring the applications and recent advancements in FNNs, alongside an examination of the Damerau-Levenshtein Distance(DLD) and FNN noise training applications. Chapter 3 Methodologies: elaborates on the identified problem within cybersecurity, specifically focusing on the obfuscated DGAs and the potential role of FNNs in addressing this challenge. Chapter 4 Experiments outlines the dual objectives of the research: evaluating FNN effectiveness in deciphering obfuscated algorithms and understanding the characteristics of Noise and Linear Recursive Sequences in DGAs. Chapter 5 conclusion offers additional background information to enhance contextual understand-

ing. Chapter 6 details the overall organization of the dissertation. Subsequent chapters (7-10) cover the methodology, results and analysis, discussion, and conclusion, providing a comprehensive exploration of the research findings. The final chapter, Chapter 11, suggests potential avenues for future research.

Chapter 2

Literature Review

2.1 Feedforward Neural Networks

Feedforward neural networks have found applications in various fields, including computer vision, natural language processing, and pattern recognition [27]. A feedforward neural network is a type of artificial neural network characterized by its unidirectional flow of information. In this architecture, data travels sequentially from the input layer through any intermediate hidden layers, if present, and finally reaches the output layer. Unlike recurrent neural networks, feedforward networks lack cycles or loops in their connections. Each layer of nodes serves as a function that transforms the input data into a more abstract representation at each subsequent layer [28].

Recent research in the field of neural networks has witnessed significant progress in understanding and optimizing feedforward architectures. Studies have explored diverse aspects, from the network's structure to its activation functions. For instance, researchers have explored the design of novel activation functions to introduce non-linearity, which is crucial for the network to capture complex relationships in data [29]. Additionally, advancements have been made in the learning and training process, with techniques like backpropagation being refined to efficiently adjust the weights assigned to each connection during training [30].

2.2 Damerau-Levenshtein Distance (DLD)

[31] explores the application of the DLD algorithm for word correction, with a specific focus on mitigating extended processing times, especially concerning large dictionaries. To address this challenge, the study introduces a novel method of expediting the DLD algorithm by distributing the dictionary based on the number of characters, resulting in a substantial 29.04-second improvement in processing time. Their research provides an in-depth exploration of the DLD algorithm's operations, including substitution, insertion, deletion, and transposition, emphasizing its significance in spell-checking. They discuss the challenges associated with processing time, referencing related research that optimized the algorithm for improved efficiency. Demonstrating the proposed dictionary distribution method's effectiveness, their study utilizes two stories from a website, showcasing reduced processing time without compromising accuracy. Their research methodology encompasses data selection, the application of the DLD algorithm, and the implementation of dictionary distribution. Crucial components include word suggestions and accuracy assessment, mainly focusing on words with an edit distance of 1 to 3 characters.

Adapting the Methodology for Identifying DGAs with FNNs: Although the primary focus of [31] research is word correction using DLD, its methodology and findings offer a valuable foundation for identifying Domain Generating Algorithms (DGAs) with FNNs. DGAs often involve generating random and obfuscated domain names, posing challenges akin to correcting misspelled words. Leveraging the DLD principles for measuring word differences, FNNs can be trained to discern patterns within DGAs. The innovative distributed dictionary approach, designed to enhance processing time, can be transposed to create a diverse dataset for training FNNs. This allows FNNs to learn the structural similarities and variations within DGAs, including those modified with noise, Linear Recursive Sequence (LRS) or other obfuscation techniques. The research's focus on accuracy and efficiency aligns seamlessly with the requirements of effectively identifying DGAs

in real-world cybersecurity scenarios. The success of the DLD algorithm in addressing language-related challenges provides a solid framework for adapting similar techniques to the intricate patterns exhibited by DGAs, contributing significantly to enhanced threat detection and network security.

2.3 FNN and Noise Training

Applications of feedforward neural networks span a wide range of fields, including computer vision, natural language processing, and pattern recognition[27]. Notable breakthroughs in these areas have been enabled by the adaptability and versatility of feedforward architectures. Furthermore, the continuous evolution of this neural network paradigm has led to the development of more efficient training algorithms and the exploration of new architectures such as convolutional and recurrent neural networks [32].

2.4 Domain Generating Algorithms

This section provides a comprehensive overview of research studies in the cybersecurity domain, focusing on the use of Domain Generation Algorithms (DGAs) and advanced techniques employed by malicious actors. Building upon previous investigations, the discussion encompasses tactics observed in cyber attacks, the analysis of DGAs, and advancements in cyberattack detection using deep learning. Each study addresses specific aspects of cybersecurity, contributing valuable insights to the field. The aim is to explore the intricacies of DGAs, uncover potential challenges posed by randomized techniques, and enhance cyberattack detection methodologies through cutting-edge approaches. The following sections look into the key findings and methodologies of these studies, shedding light on their contributions and potential areas for further exploration.

2.4.1 Tactics of Domain Generating Algorithms

In [33], the authors examined the sophisticated techniques employed by hackers to take control of victimized machines. They highlighted the use of Domain Generation Algorithms (DGAs) in Command and Control operations and drive-by download attacks. These activities involve coercing victims into accessing malicious websites hosting Browser Exploit Packs (BEPs) that exploit vulnerabilities in web browsers or third-party plugins. The authors introduced a classification system covering binary-based DGAs for Command and Control functions, and scripted-based DGAs for infection proliferation. While the Binary-Based DGA section briefly touched on generating pseudorandom domain names based on a predetermined seed value, the authors did not delve into specific seed values or DGA creation details. This new paper aims to provide additional insights by exploring the use of DGAs and Linear Feedback Shift Registers (LRSs), opening up new avenues for research in this domain.

2.4.2 Comprehensive Analysis of DGAs: Addressing Traditional and Randomized Techniques

In [1], authors discussed the use of DGAs by botnets for facilitating Command and Control communications in cyber attacks. They highlighted how hackers utilize algorithms to generate numerous download sites, enabling victim hosts to download malware and upload information. They also explore various modifications that attackers can make to their Tactics, Techniques, and Procedures (TTPs) involving DGAs. The paper addressed different types of DGAs, including traditional and dictionary-based DGAs, which pose challenges to defensive detection methods. However, they do not explicitly discuss randomized DGAs that could mimic random web addresses generated by search engines like Google, making detection even more difficult. The authors also mention DGArchive, a malicious domain dataset containing 84 malware families at the time of their report,

which they utilize for their research. In their approach, the authors discuss botnets' use of DGAs for C2 communications and explore attackers' modifications to their tactics. However, they focus on traditional and dictionary-based DGAs and overlook the challenges posed by randomized DGAs, which mimic search engine-generated web addresses [31]. The paper lacks an in-depth analysis of their methodologies' practical implications and limitations and mainly relies on the DGArchive dataset. In contrast, this new work offers a more comprehensive examination of DGAs, including randomized techniques and practical implications.

In [1], the authors examined botnets' deployment of Domain Generation Algorithms (DGAs) for Command and Control (C2) in cyber attacks. They highlighted the use of algorithms by hackers to create numerous download sites, facilitating malware downloads and information uploads by victim hosts. The paper explored attacker modifications to Tactics, Techniques, and Procedures (TTPs) related to DGAs, addressing traditional and dictionary-based DGAs that challenge defensive detection methods. However, it did not explicitly cover randomized DGAs, which can mimic web addresses generated by search engines like Google, adding to the difficulty of detection. The authors referenced DGArchive, a dataset with 84 malware families used in their research. In their approach [18], the authors focused on botnets' use of DGAs for C2 communications and modifications to tactics. However, they emphasized traditional and dictionary-based DGAs, overlooking challenges posed by randomized DGAs mimicking search engine-generated addresses. The paper lacked a thorough analysis of the practical implications and limitations of their methodologies, relying primarily on the DGArchive dataset. In contrast, this new work provides a more comprehensive examination of DGAs, including randomized techniques, and explores practical implications.

2.5 Advancing Cyberattack Detection with Deep Learning: A Comprehensive Study

The work by [34] centers on bolstering cyberattack detection through deep learning techniques. The study delves into the application of deep neural networks (DNNs) in cybersecurity investigations, encompassing domains like domain generation algorithms (DGAs), intrusion detection, spam and phishing detection, and secure shell (SSH) traffic analysis. The authors present a comprehensive framework, ScaleMalNet, for identifying malware in executables and categorizing malicious traffic. They emphasize the significance of accurately pinpointing harmful URLs using DNNs, highlighting the elimination of the need for traditional blacklisting and signature-matching methods. While the research underscores the potential of DNNs in tackling a variety of cybersecurity challenges, it could further benefit from integrating strategies like obfuscation with linear recursive sequences in DGAs to enhance the overall effectiveness of domain name detection, thus fortifying resilience against evolving cyber threats.

2.6 Linear Recursive Sequences

2.6.1 Limited Precision Deep Learning Architectures

In [35], researchers explored ransomware identification using limited precision deep learning architectures implemented on a Field-Programmable Gate Array (FPGA). They chose a Deep Belief Network (DBN) for training on a dataset containing ransomware samples. This training approach involved an initial unsupervised phase followed by supervised logistic regression with soft-max activation. The DBN used in this study is a generative probabilistic model that incorporates layers of Restricted Boltzmann Machines (RBMs), with the highest layer containing an undirected RBM. For most operations, an 8-bit Lin-

ear Feedback Shift Register (LFSR) was employed to generate uncorrelated bitstreams. The study showed a significant 91% overall detection rate, with a rapid detection time of 0.006ms using their FPGA setup. Unlike non-linear sequences, which lack easily recognizable patterns and reduce the predictability of botnet clients, Linear Feedback Shift Registers (LFSRs) exhibit distinct mathematical relationships among their elements, making them more predictable. Integrating LFSRs through overlapping and XORing the system results would enhance obfuscation, as any Domain Generation Algorithm (DGA) used would only interpret human-readable characters. It's important to note that in real-world scenarios involving malware-infected servers and clients, replication would still be necessary to establish proper connections.

In their recent publication, "Novel Deep Learning Approach for Detecting Domain Generation Algorithms", [36], the authors introduced a novel approach utilizing deep learning to instantly detect randomly generated domain names and DNS homograph attacks. This technique eliminates the necessity for reverse engineering or the inspection of nonexistent domains. Through evaluation, its efficacy in recognizing DNS homograph attacks and Domain Generation Algorithms (DGAs) was showcased, boasting an impressive accuracy of 0.99. Notably, the method demonstrated resilience against common evasive cyberattacks and surpassed the performance of well-known deep-learning architectures.

Through the new incorporation of XOR binary combinations involving both DGAs and Linear Recursive Sequences (LRSs) during the training phase, this approach facilitated the generation of adversarial examples that replicated intricate attack strategies. By subjecting the models to training with these examples, they were empowered to adeptly counter both white-box and black-box attacks, significantly reinforcing their aptitude to withstand efforts aimed at evasion and tampering. Diverging from the methodology detailed in reference [19], this approach integrates Linear Recursive Sequences (LRS) with DGAs, resulting in more realistic adversarial instances and amplifying the model's capability to navigate through an array of detection techniques. Furthermore, a comprehensive

comparative analysis was conducted, highlighting the superiority of the LRS-DGA hybrids. This outcome ultimately heightens obfuscation levels and reduces the occurrence of real-time cyber threat detection alerts.

This new research is focused on DLD and NDL and (NDL, string entropy, string compression, and similarities. Levenshtein Distance and DLD both quantify string dissimilarity through minimum edit operations, but they differ in handling transpositions. While Levenshtein Distance considers insertions, deletions, and substitutions, DLD extends this to include transpositions, allowing the swapping of adjacent characters. DLD is advantageous when transpositions are common, as in spell checking, while Levenshtein is more straightforward and more suitable for scenarios where transpositions are rare or less critical. The choice depends on specific application requirements and the variations expected in the compared strings. DLD and NDL, along with additional features, provided improved results of 100.

The study, "Comparative Analysis of DGA Detection Methods: Machine Learning vs. Deep Learning," presented in [37] delved into the realm of automatically detecting Domain Generation Algorithms (DGAs) through the lens of machine learning. The authors tackled two key challenges in this domain. First, they addressed the issue of comparing DGA detection methods, a challenge exacerbated by the scarcity of independent benchmarks. The second challenge involved the adaptability of cybercriminals to circumvent these detection techniques. The researchers juxtaposed two distinct approaches for DGAs. The first method involved classical machine learning employing manually crafted features, while the second method leveraged the power of deep learning neural networks. The findings revealed that adversaries exhibited a faster adaptation rate against the manually engineered features approach, succeeding approximately 57 percent of the time. However, their effectiveness diminished when confronted with deep learning detection systems, achieving an impressive accuracy of 78.9 percent.

Yang et al. [38] introduced in, "Enhancing DGA Detection with Heterogeneous Deep

Neural Networks and Obfuscation Strategies", an approach using a deep learning-based framework called Heterogeneous Deep Neural Network (HDNN) for detecting malicious domain names generated by stealthy domain generation algorithms (SDGAs), focusing on 20 DGA families. However, the study did not explore the potential of modifying the DGAs to test if the resulting generated domains could still be classified within the original families. To address this, they introduced a strategy involving obfuscation, combining a linear recursive sequence with a domain-generating algorithm. This approach shows promise in overcoming challenges related to adversarial attacks, exploring various feature extraction methods, creating diverse datasets, and conducting a comprehensive comparison with existing deep learning-based schemes for DGA detection.

The study, "Securing Cyber-Physical Systems: Deep Learning-Based Detection of Domain Generating Algorithms", in [39] proposes a deep learning-based domain name detection system to address security concerns related to identifying Domain Generating Algorithms (DGAs) in Cyber-Physical Systems (CPS). This system is designed to combat adversarial sample attacks and improve the identification of malicious domain names generated by DGAs. The authors address the complex challenges associated with detecting DGAs, including the evasive techniques attackers use to evade detection. They also highlight the adaptive and dynamic nature of DGAs, which continuously evolve to bypass conventional detection methods. However, the study does not explore the integration of DGAs with Linear Recursive Sequences (LRSs).

In [40], the authors of "Detection of DGA-Generated Domain Names with TF-IDF " tackle the crucial challenge of detecting domain name generation algorithms (DGAs) utilized by botnets to evade detection. They employ machine learning and deep learning techniques to bolster cybersecurity against these threats. Notably, they introduce the innovative use of TF-IDF (term frequency–inverse document frequency) for identifying DGA-generated domain names. The paper also conducts a thorough review of recent research efforts in this field, showcasing diverse methodologies, datasets, and models.

However, they acknowledge challenges such as dataset variability, model complexity, and feature engineering. They suggest the potential incorporation of linear combinations, though not explicitly XORed LRSs, to enhance feature selection and unveil hidden patterns in domain name generation. Overall, the paper enriches the landscape of DGA detection by combining new approaches with valuable insights from existing research.

The paper, "Enhancing DGA-Driven Botnet Detection," by [41] focuses on detecting DGA-driven botnets using Machine Learning (ML) and Deep Learning (DL) techniques. These botnets utilize dynamic domain generation algorithms (DGAs), making them challenging to detect. The study outlines a comprehensive methodology involving data preprocessing, model training with Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs). and performance assessment using metrics like precision, recall, accuracy, and F1-score. and both ML (e.g., Logistic Regression, Naive Bayes, Random Forest). They did not attempt to use FNNs. One potential enhancement could involve integrating Linear Recursive Modified DGAs, which would add an extra layer of complexity to the domain generation process, heightening the difficulty of botnet detection. This addition would lead to a more robust evaluation of detection methods against a wider range of threats, enhancing applicability to real-world scenarios where attackers continuously innovate their techniques. Moreover, discussing the challenges and benefits of incorporating such advanced algorithms would give readers a deeper understanding of the complexities involved in DGA-driven botnet detection.

The study, "Real-Time Detection of Dictionary DGA Network Traffic Using Deep Learning", presented in [42] identified a gap in effective comparisons with established deep learning frameworks. The authors asserted that their novel hybrid neural network architecture showcases consistent performance when pitted against contemporary cutting-edge deep learning architectures. Nonetheless, the study lacks a direct and effective comparative analysis vis-à-vis these existing architectures. Such a comparative examination with

other deep learning methodologies would substantiate the proposed architecture's superiority and highlight its strengths and limitations. By implementing the DGA/LRS obfuscation technique and conducting meticulous comparative analysis, the suggested hybrid neural network architecture could be directly juxtaposed against prevalent deep learning architectures. This assessment could encompass both unmodified and obfuscated domain datasets to gauge how adeptly the architecture handles dictionary-based DGAs in the context of obfuscation. This new holistic evaluation would offer a more all-encompassing grasp of the architecture's performance and its distinctive attributes when contrasted with other advanced methodologies.

2.6.2 General LFSRs Creation

The study by [27] introduces an approach for generating linear feedback shift register (LFSR) sequences to achieve maximal equidistribution. Enhancements to the algorithm were made feasible by combining the generator's output with other generators. This research also identifies a novel variant of the polynomial Linear Congruential Generator (LCGs) using modulo-2 arithmetic and input additions for desired equidistribution characteristics. This paper's approach differs significantly, presenting a technique that incorporates Linear Feedback Shift Registers (LFSRs) into Domain Generation Algorithms (DGAs) for increased obfuscation. Unlike the focus of the method proposed in this reference, which centers on maximal equidistribution through LFSR sequences by overlapping and XORing specific domain names in binary, this approach concentrates on enhancing the complexity and resilience of obscured domain names.

The study, "Enhanced DGA Detection through Integration of DGA and Linear Feedback Shift Register", in [43] compared deep learning techniques with feature-based approaches like Random Forest (RF) and Multilayer Perceptron (MLP) models. It's important to note that the assessment may not cover the full range of possible feature-based methodologies. Combining a Domain Generation Algorithm (DGA) with a Linear

Feedback Shift Register (LRS), the introduced detection system in the research incorporates a broader set of innovative features, effectively capturing the hidden patterns and unique attributes associated with DGAs. This enhancement enables a more thorough evaluation and comparison of feature-based approaches, potentially improving detection performance.

2.6.3 Critical Analysis of Xorshift Generators

In [44], the authors examined both practical and theoretical aspects of three xorshift generators introduced by [45][46]. Their investigation revealed that while these generators were fast, they suffered from unreliability. The Marsaglia generators were part of a lineage of generators based on a linear recurrence modulo 2 framework, which included SRG, Mersene twister, and other variants. The paper introduced efficient Random Number Generators (RNGs) based on xorshift operations, exhibiting commendable randomness attributes and long periods achieved through specific parameter choices [a, b, c]. However, the paper lacked a comprehensive comparative analysis against alternative RNG methodologies, potentially overlooking recent advancements in the field. Additionally, it did not explore the practical implications of the presented RNGs or address the limitations of specific applications.

2.6.4 Overcoming Data Limitations in DNS Security through Synthetic DGAs and LRS Integration

In [47], the study tackled the challenge of limited public availability of extensive DNS data logs, hindering effective detection strategy development. Legal, privacy, and bureaucratic concerns complicate data sharing across organizations. To address this scarcity, the study proposed using a combination of Domain Generation Algorithms (DGAs) and Linear Feedback Shift Registers (LRSs) to generate large-scale, authentic DNS data logs.

This approach overcomes data scarcity, allowing the sharing of synthetic datasets among organizations for comparative evaluations and the development of more effective detection methods. The upcoming research will focus on known DGA families, utilizing LRSs XORing with a known DGA family for training to identify potential DGAs.

The collective body of related articles lacks comprehensive data pertaining to the creation and analysis of Domain Generating Algorithms (DGAs) when integrated with Linear Recursive Sequences (LRSs). This work is positioned to bridge this void by offering a thorough exploration into the fusion of DGAs with LRSs, thereby expanding avenues for researchers to analyze and detect such combinations. This research addresses a critical deficiency within the current literature. Moreover, the cited related works exhibit notable gaps in terms of providing essential evidence for the development and analysis of DGAs in tandem with LRSs, along with potential detection methodologies. These gaps significantly hinder a comprehensive understanding of the synergistic application of DGAs and LRSs. The omission of discussions surrounding the creation and analysis of DGAs paired with LRSs limits the exploration of the intricacies inherent in this combined strategy. As a result, these articles miss the chance to explore effective strategies for identifying and mitigating these hybrid techniques. In light of these considerations, further research and analysis are imperative to grasp the full scope of integrating DGAs with LRSs. The existing data deficit, especially in terms of the development and analysis of such approaches and the formulation of robust detection techniques, must be resolved. By closing this knowledge gap, this comprehension of the potential cyber threats stemming from these hybrid methodologies will be enhanced, leading to more robust defense strategies against evolving cyber-attacks.

Chapter 3

Research Methodology

3.1 Overview

In the realm of cybersecurity, combating malicious activities such as Domain Generation Algorithms (DGAs) has emerged as a pivotal challenge. DGAs are frequently employed by cybercriminals to dynamically generate domain names for their malicious operations, evading traditional detection mechanisms. The research proposes a sophisticated approach to address this challenge by developing robust classifiers capable of discerning between different variations of DGAs. Three distinct classes are defined: original DGAs, those modified through character substitution to introduce noise, and those modified through Linear Recursive Sequences (LRS). A Feedforward Neural Network (FNN) is employed as the classifier, utilizing a training dataset comprising original DGAs and modified versions. The training process enhances the classifier's resilience to evolving cyber threats, crucial in the face of continuously adapting tactics by malicious actors. The methodology follows a quantitative design, utilizing Python programming and a variety of libraries for efficient data manipulation, machine learning, and visualization. The FNN architecture, with specific hyperparameters, is designed to effectively process and classify DGA data instances, demonstrating a commitment to proactive cybersecurity.

3.2 Challenges

This section navigates through various challenges encountered during the research journey, shedding light on the knowledge gap, research scope, feature selection, and other associated issues. The initial challenge emerges from a gap in understanding neural networks and machine learning intricacies, necessitating extensive self-directed learning to overcome this hurdle. The research scope, as the second challenge, involves transitioning from a macro-level perspective to a micro-level approach for accurate classification of Domain Generation Algorithms (DGAs). Additionally, challenges in feature selection arise, encompassing optimal feature identification, determining effective optimizers, learning rates, and other critical parameters for the Feedforward Neural Network (FNN). Furthermore, various issues related to methodology, such as Python version compatibility, dataset bias, and dependencies on machine learning libraries, are highlighted. The need for addressing these challenges is emphasized to enhance the overall robustness, reliability, and sustainability of the research methodology.

3.2.1 Knowledge Gap

The initial exploration into neural networks and machine learning revealed a notable knowledge gap, particularly at the macro-level of analysis. This macro-level approach, while informative, resulted in unexpected misclassifications, signaling the need for a shift towards a more granular, micro-level approach. The encountered challenges encompassed several facets, including hurdles in feature identification, complexities in selecting optimal optimizers and learning rates, and addressing issues related to dataset imbalance. These intricacies highlighted the importance of a nuanced understanding at the micro-level to effectively navigate and address the specific challenges posed by neural networks and machine learning methodologies.

3.2.2 Research Scope

The second challenge arose from initially adopting a macro-level perspective when analyzing the problem rather than a micro-level approach. Misclassifying the 80-plus datasets as a single class and creating two separate classes for noise-obfuscated and LRS-obfuscated DGAs resulted in inaccurate classifications. The correction involved recognizing that each DGA family needed its own classes, paired with noise and LRS-related classes for each family. Another hurdle was identifying DGA families with an insufficient number of samples per family, causing a risk of overfitting and false positives. Eventually, the solution involved selecting DGA families with more substantial sample sizes, ranging from 300 to 500 or more per family.

3.2.3 Feature Selection

Beyond the initial obstacle, the research encountered additional challenges, notably in the realm of feature identification and the optimization of the neural network model. A critical aspect involved the meticulous process of selecting optimal features for the Feedforward Neural Network (FNN). The identified features played a pivotal role in the network's ability to discern patterns and make informed decisions. Among the diverse set of features considered, the research explores the intricacies of DLD, NDL, entropy, compression, basic similarity, Smith-Waterman similarity, Euclidean distance, and Jaccard similarity.

The challenge extended to determining which combination of these features would best serve the research objectives, necessitating a thorough evaluation of their individual strengths and weaknesses. The significance of each feature in capturing the distinctive characteristics of the obfuscated Domain Generating Algorithms (DGAs) had to be carefully assessed.

Another set of challenges revolved around the fine-tuning of the neural network architecture. This encompassed the identification of the most effective optimizer and learning

rate, crucial factors influencing the training process and overall performance of the FNN. The determination of suitable numerical values for the dense layers, involving considerations of depth and width, added complexity to the optimization process.

Decisions regarding the number of neurons to be employed in each layer, the selection of the sigmoid activation function as the most appropriate for the research context, establishing an ideal batch size for efficient training, and defining the appropriate length of epochs further contributed to the intricate optimization challenges.

Navigating through these challenges required a delicate balance between theoretical understanding and empirical experimentation. The research sought to strike an optimal configuration that maximized the FNN's ability to unravel the hidden patterns within the obfuscated DGAs. This involved an iterative process of experimentation, evaluation, and refinement to converge on a neural network architecture and feature set that demonstrated optimal performance in addressing the complexities of the research problem.

3.2.4 Other issues

The methodology employed in the research project faces several potential challenges and suggests solutions to enhance its robustness. Firstly, the reliance on Python versions 2 and 3 for programming and the "fastDamerauLevenshtein" library might introduce compatibility issues in the future. It is advisable to consider using the latest version of Python and ensure that libraries are up-to-date to mitigate this challenge. Additionally, the choice of datasets, such as Banjori, Dnschanger, Dyre, Gameover, Murofetweekly, poses a risk of bias, and incorporating more diverse datasets could improve the model's generalization. Furthermore, the machine learning process heavily depends on libraries like pandas, numpy, scikit-learn, and keras. Ensuring version compatibility and handling potential updates or deprecations in these libraries is crucial to maintaining the script's functionality over time. The model's performance evaluation and visualization components should be enhanced by providing more precise labels and explanations, contributing to better

interpretability. Moreover, considering the complexity of the Feedforward Neural Network (FNN) architecture and the potential challenges in hyperparameter tuning, model training, and assessment, the script could benefit from breaking down into smaller functions or classes for improved organization and maintainability. Additionally, robust user input validation, detailed training output information, and implementation of save/load model functionalities would contribute to a more user-friendly and error-resilient script. Addressing these challenges would contribute to the research methodology's reliability, scalability, and sustainability.

3.3 Innovations

This section addresses the cybersecurity challenge of combating Domain Generation Algorithms (DGAs). It emphasizes creating classifiers to distinguish between different DGA variations: non-modified, character-substituted noise, and Linear Recursive Sequences (LRS). The training dataset includes original DGAs as a baseline and modified DGAs simulating real-world scenarios. The chosen classifier, the Feedforward Neural Network (FNN), excels at discerning complex patterns. Exposure to diverse DGAs enhances the classifier's ability to recognize subtle variations, making it more resilient to evolving cyber threats. The integration of character substitution and LRS modifications in the training dataset reflects real-world challenges, highlighting the importance of a versatile classifier for proactive cybersecurity.

3.3.1 Classification Framework

To effectively address this challenge, a sophisticated approach involves the creation of classifiers capable of distinguishing between different variations of DGAs. Three distinct classes are defined for this purpose: the first class represents non-modified DGAs, the second involves modifications through character substitution to introduce noise, and the

last employs Linear Recursive Sequences (LRS). This nuanced classification enables the training of classifiers to recognize intricate patterns and alterations introduced by cyber adversaries, aiming to obfuscate their malicious intent.

3.3.2 Training Dataset Composition and Structure

The complexity of the research extended to the composition and structuring of the training dataset for the classifiers. The dataset, a critical component in the development and evaluation of the classifiers, was meticulously designed to encompass three distinct categories, each contributing to a comprehensive understanding of the Feedforward Neural Network's (FNN) capabilities.

The first category within the training dataset comprised "original DGAs," acting as the baseline for the classifiers. Original DGAs represented the typical patterns associated with malicious domain generation, encapsulating the conventional characteristics that cybersecurity professionals often encounter. This category laid the foundation for the classifiers to learn and recognize the standard features indicative of malicious domain-generating algorithms.

In contrast, the training dataset included two modified categories, introducing nuances that mirrored real-world scenarios where cyber attackers employ sophisticated techniques to mask their activities. The second category involved DGAs with character substitution, simulating scenarios where attackers manipulate domain names by substituting characters. This modification aimed to challenge the classifiers with variations that emulate the obfuscation techniques commonly employed by cyber adversaries.

The third category in the training dataset consisted of DGAs with Linear Recursive Sequence (LRS) modifications. This category emulated situations where attackers introduce linear recursive sequences to obfuscate the domain generation process. The addition of this category provided the classifiers with exposure to more intricate patterns, mirroring the sophisticated methods utilized by cyber attackers to evade detection.

The deliberate structuring of the training dataset with these three categories aimed to expose the classifiers to a diverse set of patterns and challenges. The classifiers were thus trained to discern not only the standard characteristics of malicious DGAs but also to adapt and identify variations introduced through character substitution and LRS modifications. This multifaceted training approach was integral to ensuring the robustness and adaptability of the classifiers in addressing the intricacies of obfuscated domain-generating algorithms encountered in real-world cybersecurity scenarios.

3.3.3 Training the FNN for DGA Classification

The training process focuses on the Feedforward Neural Network (FNN) as the chosen classifier, known for its ability to discern intricate patterns within data, specifically in distinguishing between benign and malicious domain generation strategies in the cybersecurity domain. The FNN undergoes rigorous training with a diverse dataset, comprising original DGAs representing typical patterns associated with malicious domain generation and modified versions introducing subtle variations.

Challenges in the training process include determining optimal FNN configuration parameters, such as selecting the most effective optimizer and learning rate, defining numerical values for dense layers, choosing the number of neurons, selecting the appropriate activation function (sigmoid), establishing batch size, and determining epoch length. Balancing the FNN's generalization capabilities while avoiding overfitting or underfitting is crucial.

3.3.4 Enhancing Resilience

The integration of character substitution and LRS modifications into the training dataset enhances the classifier's simulation of evolving cyber threats. Given the continuous adaptation of malicious actors' tactics, a versatile classifier becomes indispensable for proactive cybersecurity measures.

3.4 Quantitative Methodology

This section outlines a quantitative approach for countering Domain Generation Algorithms (DGAs). Utilizing Python versions 2 and 3, the study employs the "fastDamerauLevenshtein" library and constructs an effective Feedforward Neural Network (FNN) architecture with key Python libraries. Five DGA datasets are utilized, categorized into Original DGA, Noise-Modified DGA, and LRS-Modified DGA classes. The section succinctly details the FNN's architecture, training configuration, input features, and methodology, addressing challenges in hyperparameter optimization and model assessment. Suggestions for improvement encompass organization, user input handling, comments, visualization, save/load functionality, training output, and consistency. Overall, the methodology provides a robust framework for the research on DGA countermeasures.

3.4.1 Quantitative Research Design and DGA Dataset Classification

This research was conducted with a quantitative design that focused on specific variables, analysis, and rigid constraints; it was well-structured and designed to ensure its overall validity and reliability could be replicated. A quantitative research model was followed to demonstrate how research could be conducted and how data could be collected, analyzed, and detected. Regular measurements throughout the experiment were fundamental, connecting research observations and the formalization of the research purpose.

Five distinct DGA datasets were used and reviewed; they are Banjori, Dnschanger, Dyre, Gameover, Murofetweekly, and three specific classes were used within this research, remaining 79 datasets can be viewed within Appendix A. All presented unique challenges and opportunities for classification. The first dataset comprises originally unaltered DGA domain names, serving as a foundational baseline for analysis. This dataset provides essential insights into the characteristics and patterns exhibited by legitimate DGA do-

main names. This research covers naming conventions, length distribution, and syntactic structures. This initial classification aids in the creation of a reference point against which anomalous or malicious domain activities can be identified.

The second dataset introduces injecting generated noise into known DGAs, incrementing a minimum of 10 percent of characters within each of the domain name strings. This deliberate manipulation seeks to emulate the adaptive nature of cyber threats, where attackers constantly refine their tactics to evade detection. This research focuses on identifying and quantifying the impact of noise injection on existing classification models.

The third dataset introduces a layer of novel complexity by incorporating linear LRS-modified domain-generated algorithms (DGAs). These algorithms dynamically alter domain names in a systematic manner, posing a significant challenge to traditional classification methods. Researchers must decipher these modified domains' underlying sequences and patterns to distinguish them from legitimate counterparts effectively. The exploration of linear recursive sequences becomes instrumental in uncovering the intricacies of this dataset, ultimately enhancing the accuracy of classification models against evolving cyber threats.

3.4.1.1 Programming Environment and Machine Learning Libraries

This experiment primarily used Python languages versions 2 and 3 for programming. Python was chosen because it also had a library called "fastDamerauLevenshtein," which is a string metric for measuring the edit distance between two sequences of strings. Additionally, data was collected by obtaining results from submissions to DGArchive, most likely through submissions from a predetermined Application Programmable Interface (API) application to DGAarchive.

The Python libraries used in the FNN plays crucial roles in different aspects of the machine learning process. The pandas library, referred to as pd, facilitates efficient data manipulation with its DataFrame structure, enabling tasks like reading, filtering, and

grouping structured data such as CSV files. numpy (as np) serves as a fundamental library for numerical computing, offering support for large, multi-dimensional arrays and mathematical operations on them.

The scikit-learn library, denoted as sklearn, is a comprehensive machine learning toolkit utilized for diverse tasks like model selection, preprocessing, and classification. Notably, the LabelEncoder from scikit-learn converts categorical labels into numerical format. The keras library provides tools for building neural networks, with the Sequential model allowing the creation of a linear stack of layers.

The FNN architecture uses keras.layers.Dense for fully connected layers, and the keras.layers.Dropout layer is applied for regularization. The optimization of the model is handled by keras.optimizers.Adam. Callbacks, such as keras.callbacks.EarlyStopping and ModelCheckpoint, functions were used for stopping training when a metric stops improving and saving the model checkpoints during training.

For visualization, the script utilizes matplotlib.pyplot (as plt) to create various plots and charts. seaborn (imported as sns) builds on Matplotlib, providing a high-level interface for statistical data visualization. The sklearn.metrics module supplies metrics for evaluating the model's performance, while the PCA module reduces dimensionality.

3.4.1.2 Computational Infrastructure and Hardware Configuration

Primary computations for the study were completed with a home computer, DELL Inspiron Configure To Order (CTO) Base Laptop, which held 64GB of Random Access Memory (RAM) and 2TB of Solid State Drive (SSD) space. Two additional External SanDisk SSDs, 2TB and 4TB, could be used for additional storage if required or as backups. The laptop also utilized VMWare Workstation 16 Pro, allowing different operating systems to run simultaneously.

3.5 FNN Architecture and Training Configuration

This section provides an overview of the Feedforward Neural Network (FNN) architecture designed for countering Domain Generation Algorithms (DGAs). Following a multi-class classification strategy, the FNN utilizes the Adam optimizer, sigmoid activation function, and two dense layers. The training process involves epochs, a small batch size, and a validation split. With eight input features, the FNN structure comprises input, two dense layers, and an output layer with softmax activation. The training configuration features the Adam optimizer, learning rate, epochs, sigmoid activation, batch size, and validation split. These features contribute to the FNN's comprehension of DGA data, classifying instances across datasets for robust cybersecurity. The input data comprises eight features: DLD, NDL, Entropy, Compression, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity.

3.5.1 General Training Configuration

Optimizer: The model utilizes the Adam optimizer, an adaptive learning rate optimization algorithm.

Learning Rate: The learning rate for Adam is set to 0.0001, controlling the step size during optimization. Maintaining a lower learning rate ensures more stable convergence.

Epochs: The model undergoes 50 epochs of training, which signifies the number of times the entire dataset is processed by the neural network.

Activation Methods: Sigmoid activation is employed as the activation method. Sigmoid is chosen for its suitability as a multiclass classifier, following a one-vs-all (OvA) or one-vs-rest (OvR) strategy. This entails training individual binary classifiers for each class, where each class is considered positive while the others are treated as negative. Sigmoid activation functions are well-suited for this approach. Softmax activation is used for multi-class classification tasks. It transforms the model's raw output into a probabil-

ity distribution, making it easier to interpret and evaluate the FNN's performance using appropriate loss functions.

Batch Size: Training is conducted with a batch size of 32, where each batch contributes to updating the neural network's weights.

Validation Split: 80% of data used for training and 20% of the data is reserved for validation, enabling the model to assess its performance on unseen data during training. This practice aids in preventing overfitting and provides insights into generalization capabilities.

3.5.2 FNN Model Configuration

Input Layer: The neural network starts with an input layer that accommodates the eight features mentioned above.

First Dense Layer (256 Neurons): The first dense layer uses sigmoid for activation, contains 256 neurons, and applies the sigmoid activation function. This layer is a powerful feature extractor, capturing complex patterns within the input data.

Second Dense Layer (128 Neurons): The second dense layer uses a sigmoid for activation, consists of 128 neurons, and uses the sigmoid activation function. This layer further refines the learned features from the previous layer, enabling the network to understand intricate relationships within the data.

Third Dense Layer: Uses softmax activation for multi-class classification tasks. It transforms the model's raw output into a probability distribution, making it easier to interpret and evaluate the FNN's performance using appropriate loss function.

Output Layer: The output layer employs the softmax activation function, producing a probability distribution across the three classes (Original DGA, Noise-Modified DGA, LRS-Modified DGA). The network is trained to classify input instances into one of these three categories.

3.5.3 Input Features for Neural Network

The eight features, including Class, DLD, NDL, String Entropy, String Compression, Standard String Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity collectively contribute to the neural network's understanding of the input data. These features are crucial for characterizing and distinguishing the different types of DGA data (Original, Noise-Modified, LRS-Modified).

3.5.4 DGA Dataset Selection and Preprocessing

This research extensively examines five primary DGA datasets Banjori, Dnschanger, Dyre, Gamechanger, and Murofetweekly—carefully chosen from a pool of 84 datasets available in DGArchive. The detailed results of the remaining datasets will be documented in Table 4.17 and 4.18. The dataset selection process involved narrowing down the datasets to 71 as 13 lacked a sufficient number of domain names. The selection criteria considered both the length of strings and the overall dataset size. Each selected dataset underwent thorough modification and segmentation, resulting in three distinct categories of classes: Original DGA data, Noise-Modified DGA data, and LRS-Modified DGA data. The primary objective of this multi-class classification task is to effectively train the multi-layer neural network to adeptly categorize instances into one of these three classes, leveraging the unique features provided by each dataset the features provided.

3.5.5 Summary

The FNN is structured to effectively process and classify DGA data instances using the Adam optimizer, sigmoid activation functions, and two dense layers. The choice of features and the multi-class nature of the task contribute to the network's ability to discern patterns and make informed predictions for the different types of DGA data.

3.6 Machine Learning Methodology

3.6.1 Feature Selection for FNN Training

The selection of features for training a Feedforward Neural Network (FNN) with domain generating algorithms (DGAs) is crucial for robust defense against LRS obfuscated DGAs through domain-specific noise and deep learning analysis. The "Class" feature serves as the target variable, providing distinctive labels for the various DGA types (Original, Noise-Modified, LRS-Modified), enabling supervised learning and aiding the neural network in pattern recognition. The DLD, quantifying the number of edit operations needed to transform strings, provides crucial insights into string similarity and structural disparities. This information is vital for identifying unique patterns associated with different Domain Generating Algorithms (DGAs). Normalizing DL Distance ensures a consistent scale, preventing biases and enhancing the neural network's ability to discern patterns independently of string lengths. String Entropy, representing unpredictability, and String Compression, reflecting compressibility, capture essential aspects of DGA patterns, assisting the neural network in distinguishing between random and structured strings. Standard String Similarity and Smith-Waterman Similarity provide fundamental metrics for understanding baseline string relationships and fine-grained local alignments. With its numerical representation, Euclidean Distance captures geometric relationships between vectors, contributing to the neural network's understanding of numerical patterns in DGA data. Jaccard Similarity, measuring set similarity, helps analyze specific character sets or sequences, offering insights into the distinctive character combinations typical of DGAs. These features collectively empower the neural network to comprehend and differentiate between the various DGA types, forming a robust defense against LRS obfuscated DGAs through domain-specific noise and deep learning analysis.

3.6.2 Optimizing FNN Weight Parameters

Navigating the intricacies of optimizing weight parameters in an FNN comes with various challenges, especially considering the specific architecture and training parameters outlined. One significant challenge involves selecting an appropriate learning rate for the Adam optimizer, which critically influences the convergence speed and stability during training. Additionally, striking the right balance between model complexity and generalization poses a challenge, given the risk of overfitting with the specified architecture of 256 and 128 neurons in the two dense layers and a Softmax dense layer for processing output in chartable format. Techniques like dropout or weight regularization may be necessary to mitigate this risk. The characteristics of the dataset, such as imbalanced classes or noisy features, can significantly impact the model's performance, requiring preprocessing steps like normalization and feature engineering. The validation split of 20% is a common choice but may need adjustment based on the dataset's size and complexity. Hyperparameter tuning, including the number of epochs and batch size, demands careful consideration to avoid underfitting or overfitting. Feature scaling is crucial for the Adam optimizer's efficiency, and the interpretation of results can be challenging with a complex architecture, necessitating techniques like feature importance analysis or more info SHAP values. SHAP (SHapley Additive exPlanations) values are a method used in machine learning to explain the output of a model for a specific instance or prediction[48]. They provide a way to allocate the contribution of each feature to the final prediction. Moreover, the computational demands of training a deep neural network, coupled with the need for domain knowledge for effective decision-making, add further layers of complexity to the optimization process. Regular validation and monitoring are paramount for iteratively refining the model's performance in the face of these challenges.

3.6.3 Model Assessment During Research

Comments and suggestions for improvement: Graphical Visual Representation: Labeling of the Title and other items needed to be increased to improve readability. Organization: The model could be broken down into separate Python files to ease editing and improve readability and maintainability. User Input Handling: Improved input validations and possible default inputs could be added to reduce errors. Validation checks may be added to check for valid integers or floats. Visualization: Have included various visualization functions, which are great for understanding model performance. Make sure that the visualizations are correctly labeled and easy to interpret. Save and Load Model: implement functionality to save and load models. Ensure that the paths the user provides are correct and that the models are saved and loaded successfully. Training Output: Print more information during training, such as the current epoch, loss, and accuracy. This can help users monitor the training progress more effectively. Consistency: Ensure consistency in variable naming conventions and coding style throughout the script. Error Handling: Implement robust error handling to gracefully handle potential issues, such as incorrect file paths or invalid user inputs.

Chapter 4

Experiments

4.1 Results

This section, will cover outcomes of the artifact and content analysis, explore the results generated by machine learning processes, and assess the performance of the multi-layer FNN model.

4.1.1 Artifact and Context Results

4.1.1.1 Datasets

Initially, a vast pool of 84 Domain Generation Algorithms (DGAs) was identified from the DGArchive repository due to its extensive collection. Subsequently, the selection was refined to approximately 72 DGAs for the specific focus of this dissertation. The study centers on a detailed examination of five distinct DGAs, while the outcomes for the remaining DGAs are systematically presented in an appendix for comprehensive review.

To facilitate classification and analysis, three classes were defined for each DGA:

Original DGAs: This class constitutes the baseline dataset, representing unaltered DGAs that capture the typical patterns associated with malicious domain generation.

Noise-Modified DGAs: This dataset introduces injected noise to emulate adaptive cyber threats. The noise is generated by randomly performing a 10% one-up character modification. For example, in the domain "abc123.com," the character 'c' is modified to 'd,' resulting in "abd123.com."

LRS-Modified DGAs: This dataset incorporates modifications through Linear Recursive Sequences (LRS), presenting challenges to conventional classification methods. The LRS modification involves creating an extended sequence of binary ones and zeros. At a specific point in the sequence, the binary is XOR-added to the original DGA's binary, and the resulting human-readable characters are utilized as the domain name. This deliberate complexity adds a layer of intricacy for more nuanced analysis. iginal DGA and the resulting human readable characters were used as the domain name.

4.1.1.2 FNN Architecture

The model structure employed comprises a Feedforward Neural Network (FNN) characterized by the Adam optimizer, sigmoid activation functions, and two dense layers containing 256 and 128 neurons, respectively. Using the Adam optimizer enhances the model's efficiency in optimizing and updating its parameters during the training process. Sigmoid activation functions are applied to introduce non-linearity, crucial for capturing complex patterns within the data. The two dense layers, comprising 256 and 128 neurons, contribute to the model's capacity to learn hierarchical representations of the input features.

Several key parameters are defined to guide the learning process regarding the training configuration. These include the learning rate, which dictates the step size during optimization, the number of epochs representing the complete iteration through the dataset, the activation methods applied to introduce non-linearity, the batch size determining the number of samples processed in each iteration, and the validation split specifying the proportion of the dataset reserved for validation during training. The careful consideration and tuning of these parameters are pivotal in optimizing the model's performance and ensuring effective learning from the provided data.

4.1.1.3 Machine Learning Methodology

Feature Selection: Eight features, including DLD, NDL, Entropy, Compression, and others. Referenced in Chapter (3.6.1)

Engineering Challenges: Considerations for learning rate, model complexity, class imbalance, and feature scaling. Referenced in Chapter (3.6.2)

4.1.1.4 Challenges Faced

The initial exploration into neural networks and machine learning revealed a notable knowledge gap, particularly at the macro-level of analysis. This macro-level approach, while informative, resulted in unexpected misclassifications, signaling the need for a shift towards a more granular, micro-level approach. The encountered challenges encompassed several facets, including hurdles in feature identification, complexities in selecting optimal optimizers and learning rates, and addressing issues related to dataset imbalance. These intricacies highlighted the importance of a nuanced understanding at the micro-level to effectively navigate and address the specific challenges posed by neural networks and machine learning methodologies.

4.1.1.5 Innovations Introduced

The project involves the intricate task of developing classifiers with the capability to discern variations in Domain Generating Algorithms (DGAs). This necessitates the creation of a comprehensive training dataset that incorporates not only the original DGAs but also versions that have been modified through the introduction of noise and Linear Recursive Sequences (LRS). Adding these modified DGAs to the dataset is crucial for ensuring that the classifiers are robust and adaptable to diverse obfuscation techniques employed by cyber threats. A key component of the project is the utilization of Feedforward Neural Networks (FNN) as a powerful tool to enhance the resilience of the classifiers to evolving cyber threats. The FNN is a sophisticated model capable of learning intricate patterns

within the modified DGAs, thereby enabling a more effective identification and analysis of the obfuscated algorithms. This strategic use of FNN contributes to the project's overarching goal of developing advanced classifiers that can withstand the dynamic nature of cyber threats and provide robust defense mechanisms in cybersecurity.

4.1.1.6 Methodology Enhancements

Addressing potential challenges like Python version compatibility and diverse dataset incorporation. Recommendations for script improvement, including clearer labels, user-friendly features, and enhanced performance evaluation.

4.1.1.7 Model Assessment Suggestions

Organizational improvements for code readability and maintainability. Enhanced user input handling, validation, and default inputs. Inclusion of detailed comments, additional visualizations, and save/load model functionality. Robust error handling for potential issues.

4.1.2 Machine Learning Results

Preliminary findings and examination of datasets, including Banjori, Dnschanger, Dyre, Gameover, and Murotfetweekly, are detailed in Sections 4.1.2.1 to 4.1.2.5. The comprehensive results are summarized in Table 4.16. Total results for all datasets can be found in Tables 4.17 and 4.18.

4.1.2.1 Banjori Dataset

The Banjori dataset examples as shown in tables 4.1 through 4.3 are organized into three distinct classes: LRS Modified (Class 1), Noise Modified (Class 2), and Original (Class 0). In the LRS Modified class, where domain names have undergone intentional modifications, the DLD ranges from 13 to 14, indicating the number

of edit operations needed for transformation. The Normalized DLD hovers around 0.778, ensuring a consistent scale regardless of string lengths. Entropy values vary between 2.90 and 3.38, reflecting unpredictability, while Compression ranges from 2.11 to 2.43, indicating compressibility. Similarity and Smith-Waterman Similarity metrics provide insights into string relationships, with values ranging from 31 to 44 and 0.44 to 1.17, respectively. Euclidean Distance spans from 75.512 to 102.274, and Jaccard Similarity ranges from 0.278 to 0.529. In the Noise Modified class, characterized by intentional character substitutions, DLD values are consistently 1, indicating minimal edit operations. Normalized DLD remains low at 0.056, ensuring a standardized scale. Entropy values vary from 2.90 to 3.38, with Compression ranging from 2.11 to 2.11. Similarity and Smith-Waterman Similarity metrics are uniformly high at 94, and Euclidean Distance and Jaccard Similarity showcase values of 1.0. The Original class, representing unchanged domain names, exhibits DLD and Normalized DLD values of 0, indicating no edit operations. Entropy values vary but typically fall within the range of 2.90 to 3.38, with Compression consistently at 2.11. Similarity and Smith-Waterman Similarity metrics are uniformly at 100, indicating identical domains, while Euclidean Distance and Jaccard Similarity reflect values of 0.0 and 1.0, respectively. Comparing these classes, variations in feature values across LRS Modified and Noise Modified examples illustrate intentional modifications, while the Original class maintains identical domain characteristics. These numerical insights contribute to the understanding and classification of Domain Generation Algorithms within the Banjori dataset.

Table 4.1: Banjori Dataset LRS Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
xjsrrsensinaix.com	hzbbbcubxpyh.com	1	14	0.778	2.90	2.25	31	0.44	102.274	0.278
hlrfrsensinaix.com	xcvbcubxpyh.com	1	14	0.778	3.24	2.33	33	0.44	87.721	0.316
fnosrsensinaix.com	vcbcubxpyh.com	1	14	0.778	2.99	2.43	36	0.44	75.888	0.278
qcwcrsensinaix.com	asfsbcubxpyh.com	1	13	0.722	3.24	2.25	44	1.17	102.274	0.368
lbzorsensinaix.com	rkbcubxpyh.com	1	13	0.722	3.38	2.43	43	1.11	76.118	0.368
sgjprsensinaix.com	cwbcubxpyh.com	1	14	0.778	3.18	2.43	36	0.44	75.512	0.316
aybarsensinaix.com	qisqbcubxpyh.com	1	14	0.778	3.04	2.25	39	0.83	102.274	0.529
tbmzrsensinaix.com	drjbcubxpyh.com	1	13	0.722	3.38	2.33	40	1.11	91.777	0.35
lzpzrsensinaix.com	jajbcubxpyh.com	1	14	0.778	3.24	2.33	40	0.83	92.558	0.368
dnkirsensinaix.com	tzybcubxpyh.com	1	14	0.778	2.99	2.33	33	0.44	89.28	0.25

Table 4.2: Banjori Dataset Noise Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
xjsrrsensinaix.com	xjsrrsensinajx.com	2	1	0.056	2.90	2.11	94	1.06	1.0	1.0
hlrfrsensinaix.com	hlrfrsensioaix.com	2	1	0.056	3.24	2.11	94	1.06	1.0	1.0
fnosrsensinaix.com	fnosrsfnsinaix.com	2	1	0.056	2.99	2.11	94	1.06	1.0	0.917
qcwcrsensinaix.com	qcwcrsensinajx.com	2	1	0.056	3.24	2.11	94	1.06	1.0	0.929
lbzorsensinaix.com	lbzorsensjnajx.com	2	2	0.111	3.38	2.11	89	1.11	1.414	0.867
sgjprsensinaix.com	sgjpssensinaix.com	2	1	0.056	3.18	2.11	94	1.06	1.0	0.929
aybarsensinaix.com	bybarsensinaix.com	2	1	0.056	3.04	2.11	94	0.94	1.0	1.0
tbmzrsensinaix.com	tbmzrsensinbix.com	2	1	0.056	3.38	2.11	94	1.06	1.0	0.929
lzpzrsensinaix.com	lzpzrtensinaix.com	2	1	0.056	3.24	2.11	94	1.06	1.0	0.933
dnkirsensinaix.com	dnkjrsensinaix.com	2	1	0.056	2.99	2.11	94	1.06	1.0	0.929

Table 4.3: Banjori Dataset Original Example

[redacted] Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
xjsrrsensinaix.com	xjsrrsensinaix.com	0	0	0.0	2.90	2.11	100	1.00	0.0	1.0
hlrfrsensinaix.com	hlrfrsensinaix.com	0	0	0.0	3.24	2.11	100	1.00	0.0	1.0
fnosrsensinaix.com	fnosrsensinaix.com	0	0	0.0	2.99	2.11	100	1.00	0.0	1.0
qcwcrsensinaix.com	qcwcrsensinaix.com	0	0	0.0	3.24	2.11	100	1.00	0.0	1.0
lbzorsensinaix.com	lbzorsensinaix.com	0	0	0.0	3.38	2.11	100	1.00	0.0	1.0
sgjprsensinaix.com	sgjprsensinaix.com	0	0	0.0	3.18	2.11	100	1.00	0.0	1.0
aybarsensinaix.com	aybarsensinaix.com	0	0	0.0	3.04	2.11	100	1.00	0.0	1.0
tbmzrsensinaix.com	tbmzrsensinaix.com	0	0	0.0	3.38	2.11	100	1.00	0.0	1.0
lzpzrsensinaix.com	lzpzrsensinaix.com	0	0	0.0	3.24	2.11	100	1.00	0.0	1.0
dnkirsensinaix.com	dnkirsensinaix.com	0	0	0.0	2.99	2.11	100	1.00	0.0	1.0

1. Training Accuracy/Loss and Validation Performance: In Figure 4.1, The outcomes of training and validating the model on the Banjori dataset unveil insightful patterns in its performance. During the training phase, both accuracy and loss exhibit notable trends. The training accuracy initiates at almost zero around the 25th epoch, indicating initial difficulties in correctly classifying the training data. Correspondingly, the training loss remains nearly zero until approximately the 50th epoch, implying gradual improvement and enhanced fitting to the training data over time. On the other hand, the validation phase showcases distinct behavior. Validation

accuracy demonstrates a swift ascent, starting around the 4th epoch and gradually reaching 100% accuracy by the 50th epoch. Simultaneously, the validation loss experiences a continuous decline, gradually approaching zero by the 50th epoch. These results collectively suggest that the model, though facing challenges in the early training stages, quickly generalizes well to new, unseen data, achieving optimal performance with high accuracy and minimal loss by the end of the validation process.

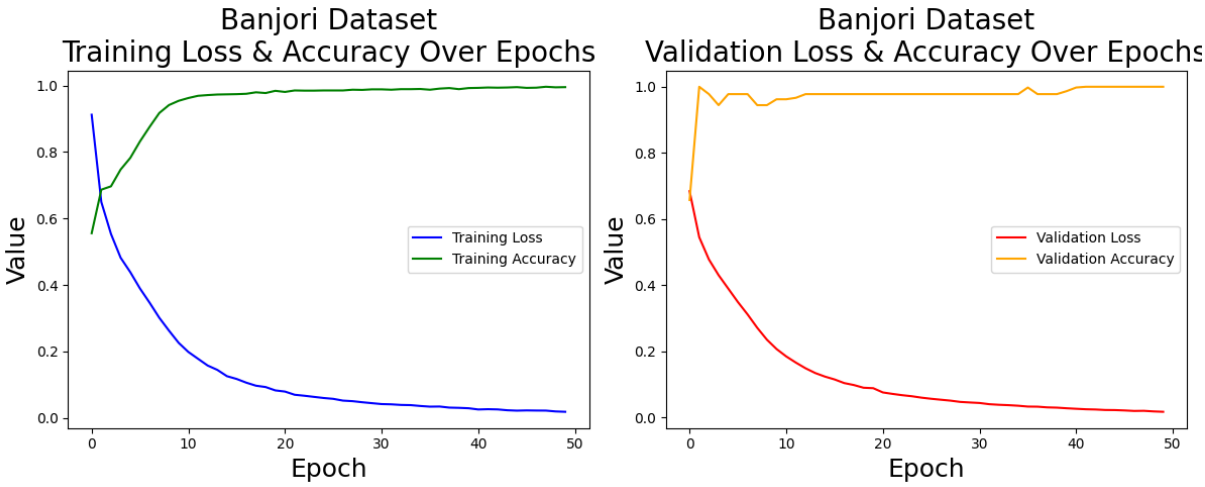

Figure 4.1: Banjori Dataset Training Loss & Accuracy / Validation Loss & Accuracy

2. Confusion Matrix: In the analysis of the Banjori dataset, a systematic approach is observed in Figure 4.2 the handling of data. Each of the initial classes starts with 750 lines of data, organized within a CSV file. Notably, approximately 80 percent, or 600 lines, are dedicated to training, while the remaining 20 percent, equivalent to 150 lines, is set aside for validation purposes.

   The evaluation of the model's performance on the validation set showcases successful recognition rates for distinct classes. The "Original DGA" class is effectively identified in 596 instances, the "DGA with LRS" class demonstrates accurate recognition with 597 instances, and the "DGA with Noise" class achieves successful recognition in 607 instances.

These outcomes signify a robust ability of the model to classify instances across different classes within the Banjori dataset. The allocation of data for training and validation, coupled with the model's accurate recognition of specific classes, highlights its proficiency in generalizing to new, unseen data and effectively discerning patterns within the dataset.

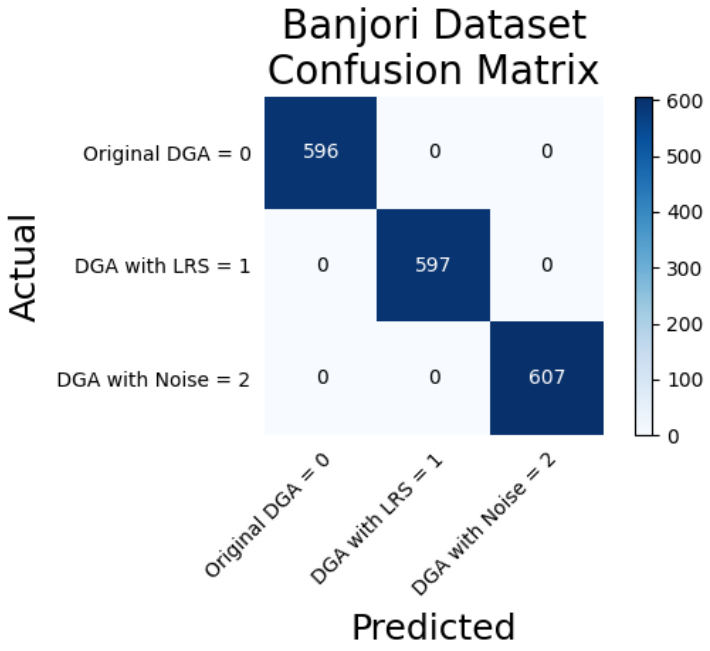

Figure 4.2: Banjori Dataset Confusion Matrix

3. F1 Score and ROC Curve: The impressive F1 Score and ROC results depicted in Figures 4.3 and 4.4 for the Banjori dataset can be attributed to specific characteristics intrinsic to the dataset. The strings undergoing comparison are of moderate length, approximately 14 alphanumeric characters each, and are composed of a restricted set of 36 options per character.

   This controlled and limited character space contributes to the model's exceptional performance in terms of F1 Score and ROC metrics. The moderate string length strikes a balance, providing enough complexity for the model to discern meaningful patterns and relationships while remaining manageable for accurate classification.

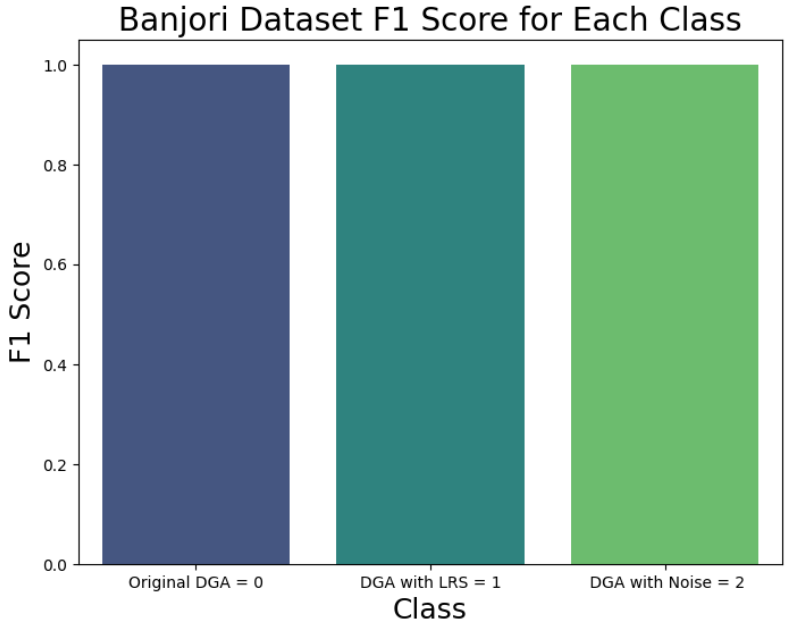

Figure 4.3: Banjori Dataset F1 Score

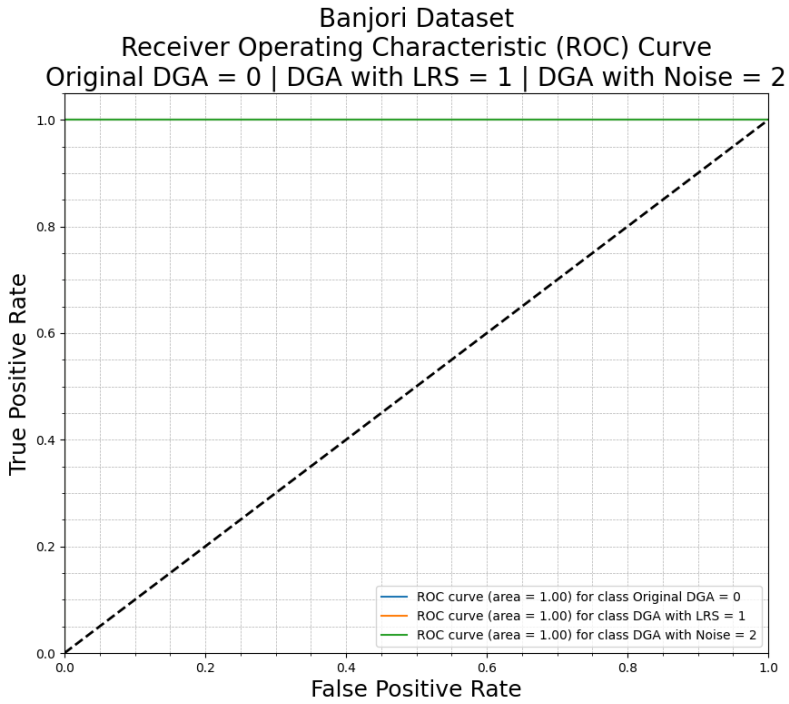

Figure 4.4: Banjori Dataset ROC Curve

4. The 2D scatter graph visually represents the model's performance across different

DGA classes in the Murofetweekly Dataset. The distinct clusters and striations provide valuable insights into the model's ability to accurately predict and classify instances of Original, DGA with Noise, and DGA with LRS classes, aiding in the interpretation and potential refinement of the model for improved performance.

The 2D scatter graph results in Figure 4.5 of the Murofetweekly Dataset provide a visual representation of the model's predictions and actual values. In the graph, a single circle in purple, representing the Actual Original class, is positioned near grid location 5 by -55. This suggests that the model effectively recognizes instances of the Original DGA class, as evidenced by the clustering of data points around this specific grid location.

Additionally, the graph illustrates the positioning of Actual and Predicted DGAs with Noise as yellow boxes, forming three distinct striations near the left side of the grid. This indicates that the model's predictions for the DGA with Noise class are grouped in these specific regions, suggesting some level of accuracy in identifying instances of this class.

Moreover, Actual and Predicted DGAs with LRS are observed on the right side of the grid, forming multiple striations from the top right to the bottom center. This spatial distribution indicates that the model's predictions for the DGA with LRS class align closely with the actual values and are concentrated in these specific areas.

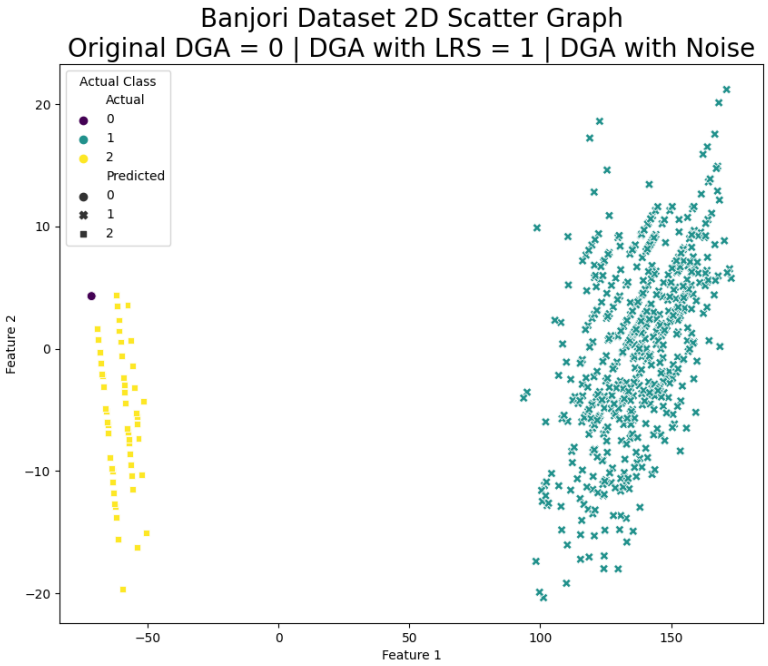

Figure 4.5: Banjori Dataset 2D scatter

5. The 3D scatter graph provides a comprehensive visualization of the model's performance across different DGA classes in the Murofetweekly Dataset. The distribution of points in the three-dimensional space offers a more detailed representation of the model's accuracy in predicting and classifying instances of Original, DGA with Noise, and DGA with LRS classes, facilitating a more nuanced interpretation and potential refinement of the model for enhanced performance.

   The 3D scatter graph results of the Murofetweekly Dataset in Figure 4.6 offer a multidimensional perspective on the model's predictions and actual values. In this visualization, the Single Circle Blue object representing the Actual Original class is situated near grid location 5 by -55. The position of this object in the three-dimensional space suggests that the model effectively recognizes instances of the Original DGA class, as it converges around this specific coordinate.

   Additionally, the graph illustrates the positioning of Actual DGAs with Noise as

green circles, forming a cluster near the left side of the grid. The three-dimensional nature of the plot allows for a more nuanced understanding of the distribution of these points, providing insights into the model's accuracy in identifying instances of the DGA with Noise class.

Furthermore, Actual DGAs with LRS are observed on the right side of the grid as orange circles, forming multiple striations from the top right to the bottom center. The three-dimensional scatter plot captures the spatial arrangement of these points, indicating that the model's predictions for the DGA with LRS class closely align with the actual values and are concentrated in specific areas.

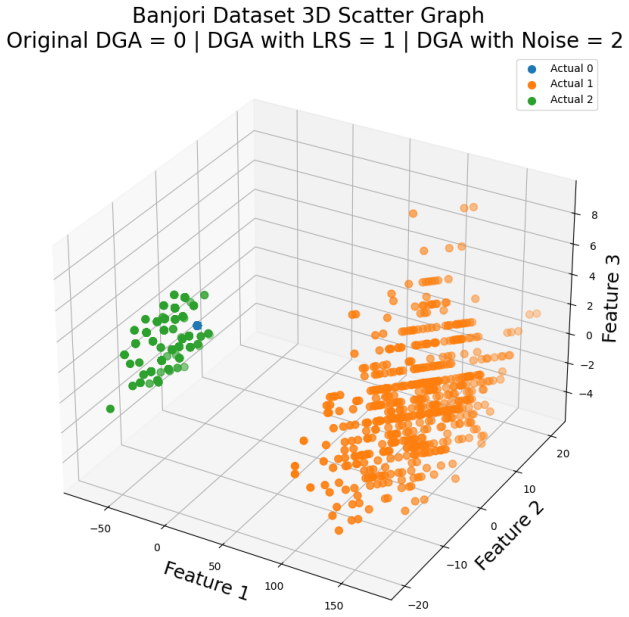

Figure 4.6: Banjori Dataset 3D Scatter

4.1.2.2 Dnschanger Dataset

Tables 4.4 through 4.6 present example dataset creation from the Dnschanger dataset categorized into three classes: LRS Modified (Class 1) and Noise Modified (Class

2). The "Initial Domains" column denotes the original domain names, "Modified Domains" indicates the intentionally modified versions, and "Class" represents the respective class assignment. Various feature metrics, including DLD, Normalized DLD, Entropy, Compressed, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity, offer insights into the characteristics of each example. In the LRS Modified class, DLD values range from 7 to 10, indicating the number of edit operations needed for transformation. Normalized DLD is between 0.5 and 0.714, maintaining a consistent scale. Entropy values span from 2.92 to 3.38, reflecting varying degrees of unpredictability. Compression ranges from 2.43 to 2.82, indicating differences in compressibility. Similarity and Smith-Waterman Similarity metrics range from 38 to 55 and 0.71 to 1.36, respectively. Euclidean Distance spans from 88.17 to 101.509, and Jaccard Similarity ranges from 0.368 to 0.538. In the Noise Modified class, DLD values are consistently 1 or 2, indicating minimal edit operations. Normalized DLD is low at 0.071 or 0.143, maintaining a standardized scale. Entropy values vary from 2.85 to 3.32, with Compression consistently at 2.43. Similarity and Smith-Waterman Similarity metrics are uniformly high at 79 to 93 and 1.07 to 1.21, respectively. Euclidean Distance is between 1.414 and 1.732, and Jaccard Similarity ranges from 0.769 to 0.923. Comparing the LRS Modified and Noise Modified classes, variations in DLD, Normalized DLD, and Euclidean Distance values indicate intentional modifications in the LRS Modified examples, while the Noise Modified examples demonstrate intentional character substitutions. Both classes exhibit varying degrees of unpredictability (Entropy) and compressibility (Compression). The numerical insights from these tables contribute to the understanding and classification of Domain Generation Algorithms within the Dnschanger dataset.

Table 4.4: Dnschanger Dataset LRS Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
duyvxjbzhq.com	tehfhzrky.com	1	10	0.714	3.32	2.54	40	0.86	98.919	0.368
upxkxxvebm.com	eihhfts.com	1	10	0.714	2.85	2.82	45	0.93	88.983	0.312
csixhvdhcj.com	scxhxftyr.com	1	7	0.5	2.92	2.54	54	1.36	93.979	0.467
xonzhsspmk.com	hjxccaz.com	1	10	0.714	3.12	2.82	48	1.21	98.737	0.538
phqghumeay.com	xwxetph.com	1	10	0.714	3.12	2.82	48	1.00	87.373	0.467
lnlfdxfirc.com	vthvxcr.com	1	8	0.571	2.92	2.82	55	0.71	95.979	0.429
vscxggbwkf.com	fcrhwwrfzw.com	1	9	0.643	3.12	2.43	50	1.29	50.636	0.4
nqduxwfnfo.com	auehgvw.com	1	9	0.643	2.92	2.82	55	1.07	94.91	0.375
zvsrtkjpre.com	jfbbdzact.com	1	10	0.714	3.12	2.54	38	1.14	88.17	0.412
tsslaueptm.com	dcbqeuae.com	1	9	0.643	2.92	2.67	50	0.71	101.509	0.5

Table 4.5: Dnschanger Dataset Noise Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
duyvxjbzhq.com	duyvyjbzhq.com	2	1	0.071	3.32	2.43	93	1.07	1.0	0.929
upxkxxvebm.com	upxkyywebm.com	2	3	0.214	2.85	2.43	79	1.21	1.732	0.769
csixhvdhcj.com	ctixhvdhcj.com	2	1	0.071	2.92	2.43	93	1.07	1.0	0.833
xonzhsspmk.com	xonzisspmk.com	2	1	0.071	3.12	2.43	93	1.07	1.0	0.833
phqghumeay.com	piqghumeay.com	2	1	0.071	3.12	2.43	93	1.07	1.0	0.923
lnlfdxfirc.com	mnlfdyfirc.com	2	2	0.143	2.92	2.43	86	1.00	1.414	0.833
vscxggbwkf.com	vscyggbwkf.com	2	1	0.071	3.12	2.43	93	1.07	1.0	0.846
nqduxwfnfo.com	nqduxxfnfp.com	2	2	0.143	2.92	2.43	86	1.14	1.414	0.833
zvsrtkjpre.com	zvsrtkkpre.com	2	1	0.071	3.12	2.43	93	1.07	1.0	0.923
tsslaueptm.com	tttlaueptm.com	2	2	0.143	2.92	2.43	86	1.00	1.414	0.909

Table 4.6: Dnschanger Dataset Original Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
duyvxjbzhq.com	duyvxjbzhq.com	0	0	0.0	3.32	2.43	100	1.00	0.0	1.0
upxkxxvebm.com	upxkxxvebm.com	0	0	0.0	2.85	2.43	100	1.00	0.0	1.0
csixhvdhcj.com	csixhvdhcj.com	0	0	0.0	2.92	2.43	100	1.00	0.0	1.0
xonzhsspmk.com	xonzhsspmk.com	0	0	0.0	3.12	2.43	100	1.00	0.0	1.0
phqghumeay.com	phqghumeay.com	0	0	0.0	3.12	2.43	100	1.00	0.0	1.0
lnlfdxfirc.com	lnlfdxfirc.com	0	0	0.0	2.92	2.43	100	1.00	0.0	1.0
vscxggbwkf.com	vscxggbwkf.com	0	0	0.0	3.12	2.43	100	1.00	0.0	1.0
nqduxwfnfo.com	nqduxwfnfo.com	0	0	0.0	2.92	2.43	100	1.00	0.0	1.0
zvsrtkjpre.com	zvsrtkjpre.com	0	0	0.0	3.12	2.43	100	1.00	0.0	1.0
tsslaueptm.com	tsslaueptm.com	0	0	0.0	2.92	2.43	100	1.00	0.0	1.0

1. Training Accuracy/Loss and Validation Performance: In Figure 4.7, The outcomes observed in the training and validation phases of the dnschanger dataset shed light on the performance dynamics of the model. In the training phase, the accuracy initiates at approximately 60 percent and progressively matures to reach 100 percent over the course of 50 epochs. Concurrently, the training loss undergoes a healthy decline, commencing at 90 percent and gradually reducing to near zero percent by the 50th epoch. Turning attention to the validation phase, the accuracy experiences a gradual increase, starting at 60 percent and steadily advancing to nearly

100 percent by the 10th epoch. In parallel, the validation loss shows a continuous decrease, starting around 70 percent and diminishing to zero by the 50th epoch. These patterns collectively signify that the model, during both training and validation, exhibits robust learning behavior. It not only attains high accuracy but also effectively minimizes loss, indicating its capability to generalize well to new data and optimize performance over the specified epochs.

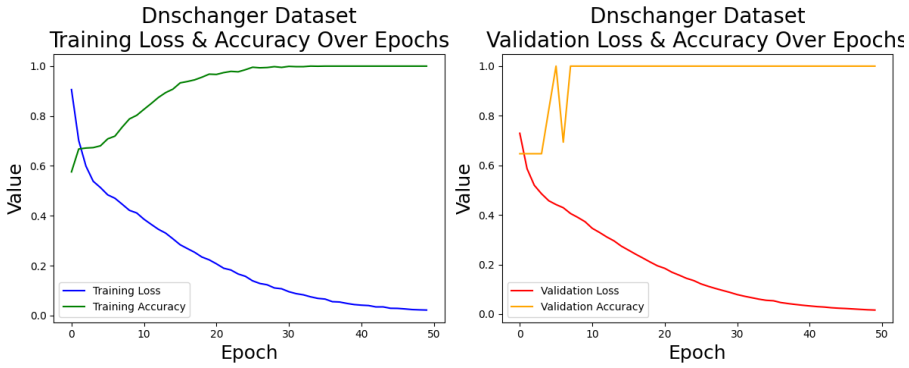

Figure 4.7: Dnschanger Dataset Training Loss & Accuracy / Validation Loss & Accuracy

2. Confusion Matrix: The Dnschanger dataset analysis reveals in Figure 4.8 a structured methodology for handling data, where each initial class begins with 750 lines of data organized within a CSV file. A deliberate allocation of approximately 80 percent, or 600 lines, is dedicated to training, while the remaining 20 percent, comprising 150 lines, is earmarked for validation. Assessing the model's performance on the validation set demonstrates a commendable recognition rate for different classes. The "Original DGA" class is successfully identified in 591 instances, mirroring the accurate recognition of 591 instances for the "DGA with LRS" class. Moreover, the "DGA with Noise" class showcases effective recognition, tallying up to 618 instances. These outcomes underscore the model's capability to accurately classify instances across various categories within the Dnschanger dataset. The meticulous allocation of data for training and validation, coupled with the model's precision in recognizing specific classes, emphasizes its adeptness in generalizing to new, unseen

data and discerning intricate patterns within the dataset.

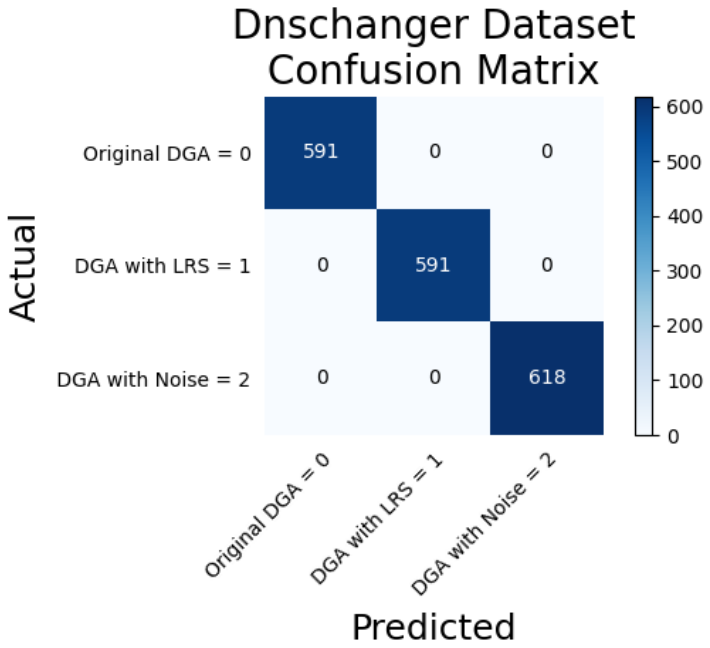

Figure 4.8: Dnschanger Dataset Confusion Matrix

3. F1-Score and ROC Curve:

   The remarkable F1 Score and ROC Curve results showcased in Figures 4.9 and 4.10 for the Dnschanger dataset can be primarily attributed to the dataset's specific characteristics. In this case, the strings being compared are relatively short, approximately 10 alphanumeric characters in length. Additionally, the strings are constructed from a limited set of 36 options per character.

   The concise length of the strings ensures that the model can effectively capture and learn intricate patterns within a manageable context. Moreover, the restricted character space facilitates the model's ability to discern and exploit the inherent structure of the dataset. These factors collectively contribute to the model's exceptional performance, reflected in the high F1 Score and ROC Curve metrics.

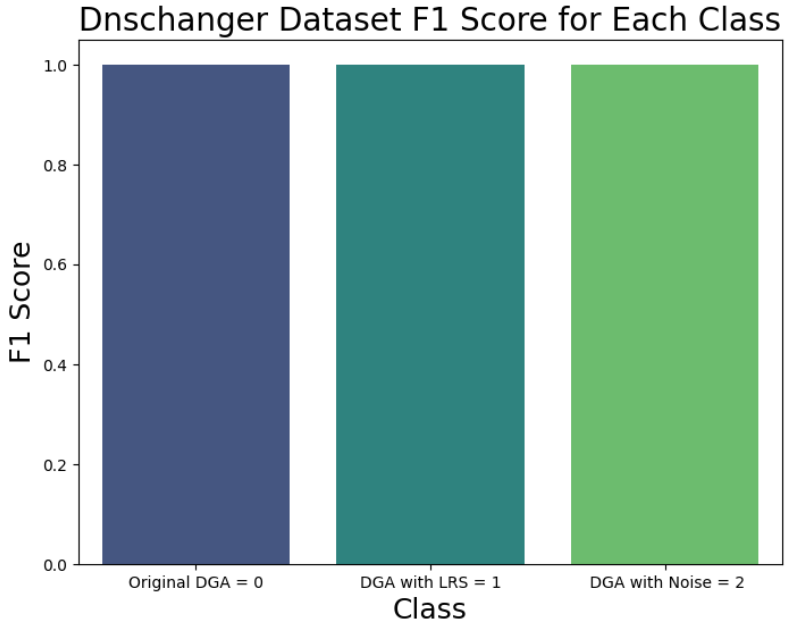

Figure 4.9: Dnschanger Dataset F1 Score

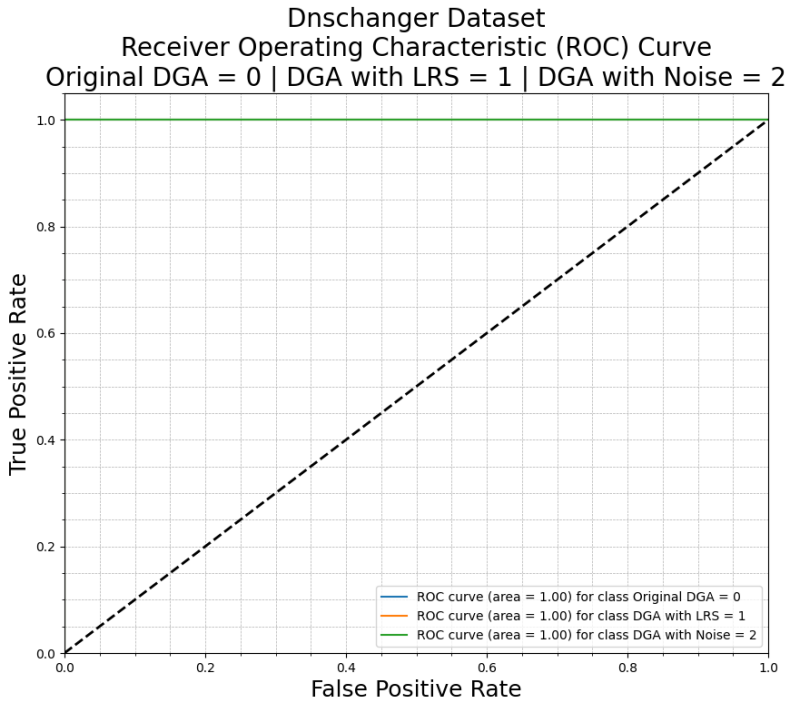

Figure 4.10: Dnschanger Dataset ROC Curve

The 2D scatter graph Figure 4.11 visualization of the Dnschanger dataset reveals

distinctive patterns and clusters that provide insights into different types of DNS activities. A single circle in purple, positioned near grid coordinates -3 to -35, likely represents actual original data points indicative of non-malicious or benign DNS behavior. On the left side of the grid, two diagonal rolls of yellow boxes annotated as noise suggest instances of Domain Generation Algorithms (DGAs) with added noise, exhibiting a discernible pattern. In contrast, on the right side of the grid, multiple rows of actual and predicted DGAs with LRS modifications are organized, leading from the center to the right corner. This arrangement implies a systematic and structured behavior associated with DGAs employing LRS.

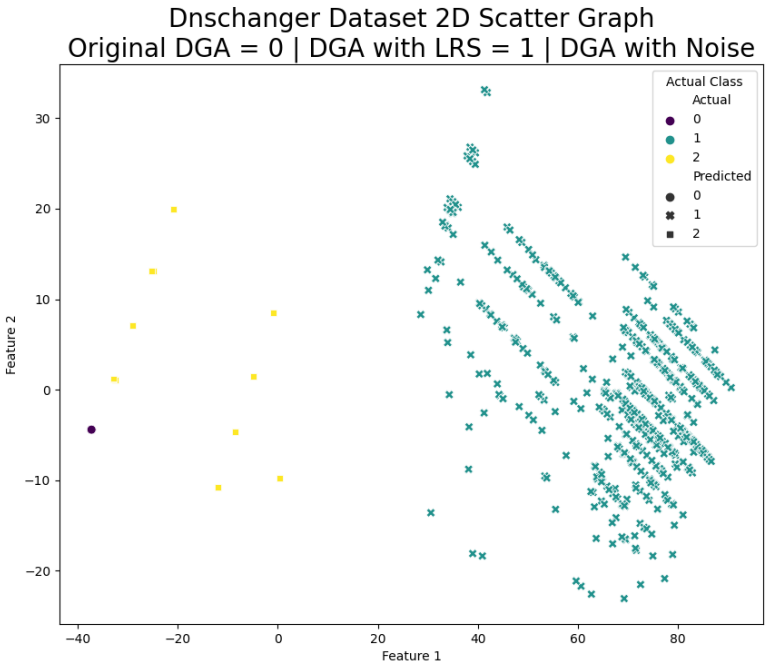

Figure 4.11: Dnschanger Dataset 2D Scatter

The 3D scatter graph visualization of the Dnschanger dataset presents a distinct pattern in its results. A single blue circle, positioned in the grid near coordinates -3, 2 to -35, signifies actual original data points, likely associated with unmodified original DNS behavior. On the left side of the grid, there is a loose grouping of 11 green

circles annotated as Actual and Predicted DGAs with noise. This suggests instances of Domain Generation Algorithms (DGAs) exhibiting less structured patterns and potentially representing a lower level of sophistication or noise in the DNS data. Conversely, on the right side of the grid, there are multiple rows of organized orange circles denoting Actual and Predicted DGAs with LRS. This organized distribution implies a more systematic and sophisticated pattern in the DNS activities, possibly indicating a higher level of complexity and intentional design.

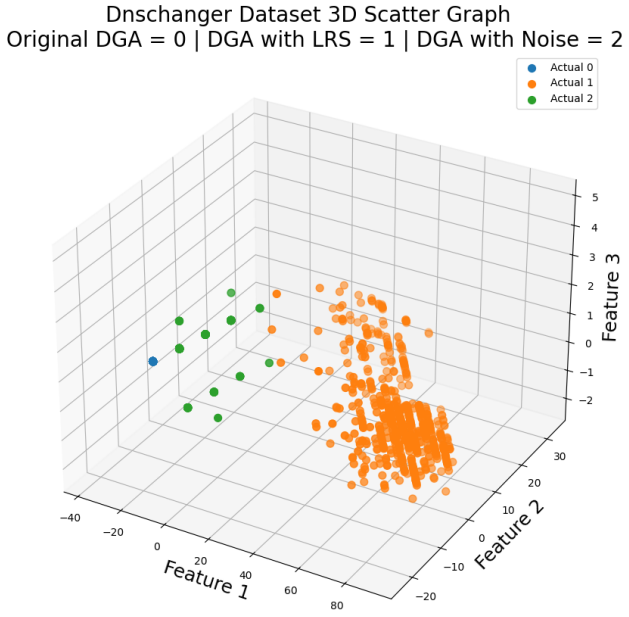

Figure 4.12: Dnschanger Dataset 3D Scatter

4.1.2.3 Dyre Dataset

The Dyre dataset tables illustrate examples categorized into three classes: LRS Modified (Class 1), Noise Modified (Class 2), and Original (Class 0). Each table provides details on Initial Domains, Modified Domains, Class assignment, DLD, NDL, Entropy, Compressed, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity.

In the LRS Modified class, DLD values range from 34, indicating a significant number

of edit operations required for transformation. Normalized DLD is consistently high at 0.919, reflecting the intentional modifications. Entropy ranges from 3.36 to 3.82, representing varying degrees of unpredictability. Compression values span from 1.54 to 2.38, indicating differences in compressibility. Similarity and Smith-Waterman Similarity metrics vary from 12 to 23 and 0.08, respectively. Euclidean Distance ranges from 161.09 to 219.854, and Jaccard Similarity ranges from 0.091 to 0.15.

In the Noise Modified class, DLD values are considerably lower, ranging from 2 to 5, indicating fewer edit operations and intentional character substitutions. Normalized DLD is also lower, varying from 0.054 to 0.135. Entropy values range from 3.36 to 3.82, with Compression consistently at 1.54. Similarity and Smith-Waterman Similarity metrics are uniformly high at 86 to 95 and 1.08 to 1.14, respectively. Euclidean Distance ranges from 1.414 to 2.236, and Jaccard Similarity varies from 0.842 to 1.0.

In the Original class, DLD values are consistently 0, indicating no edit operations. Normalized DLD is 0, and other metrics, including Entropy, Compression, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity, are at their maximum values, indicating identical Initial and Modified Domains.

Comparing the LRS Modified, Noise Modified, and Original classes, it is evident that the LRS Modified class intentionally involves more complex modifications, resulting in higher DLD and varied similarity metrics. The Noise Modified class exhibits intentional character substitutions with lower DLD and consistent high similarity metrics. The Original class represents unchanged domains with maximum similarity values. These insights aid in understanding the characteristics and intentional modifications within the Dyre dataset examples.

Table 4.7: Dyre Dataset LRS Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
b1ca5eebd8e0eb8ea6b61eaccbde527c26.ws	rrquusuuurupruqsrstts.ws	1	34	0.919	3.36	1.79	12	0.08	195.614	0.15
cdca364b71f0c8506d60eb2939f4b806d9.to	strqsvsuurvru.to	1	34	0.919	3.82	2.25	19	0.08	185.898	0.125
da3a681bc68fc6db334d1586061fe6731f.in	tqqsrwstruwuw.in	1	34	0.919	3.56	2.25	19	0.08	161.09	0.13
ecca14cc3f5be1d665cbe8992beddc6c2d.hk	usrqsrwsutssusuutssu.hk	1	34	0.919	3.52	1.70	13	0.08	208.658	0.13
fe25f54a057217ce7ecdefe7bf325b0202.cn	vuvpsutsuuvuswr.cn	1	34	0.919	3.44	2.11	17	0.08	219.854	0.136
g6854f4c620415e06da72aa3275786a6e3.tk	wvruuqqqqt.tk	1	34	0.919	3.72	2.38	23	0.08	188.085	0.13
h012e83eedf219925d9e6b3b4ee7f6788b.so	xuttuvutrrtuvs.so	1	34	0.919	3.72	2.06	18	0.08	208.394	0.13
ifcf29747fe797d6db94bb4a286777fc94.cc	yvrvwutusrrqvs.cc	1	34	0.919	3.43	2.18	18	0.08	191.937	0.091
j3df872db55a3bf2ec78fc54ff85680903.ws	zuvusprvtrvsww.ws	1	34	0.919	3.78	2.18	18	0.08	195.097	0.12
kca9453145112cbeb12d512aefb49c0a63.to	spsrusuqtwrsq.to	1	34	0.919	3.73	2.25	19	0.08	211.097	0.125

Table 4.8: Dyre Dataset Noise Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
b1ca5eebd8e0eb8ea6b61eaccbde527c26.ws	c1ca6eebd8e1eb8ea6b61eacdbde537c26.ws	2	5	0.135	3.36	1.54	86	1.08	2.236	0.875
cdca364b71f0c8506d60eb2939f4b806d9.to	cdca364c71f0c8506d60fb2939f4b806e9.to	2	3	0.081	3.82	1.54	92	1.08	1.732	1.0
da3a681bc68fc6db334d1586061fe6731f.in	da3a781bd68gc6db334d11596061ff6731f.in	2	5	0.135	3.56	1.54	86	1.14	2.236	0.842
ecca14cc3f5be1d665cbe8992beddc6c2d.hk	ecca14cc3f5ce1e665cbf8992ceddc7c2d.hk	2	5	0.135	3.52	1.54	86	1.14	2.236	0.944
fe25f54a057217ce7ecdefe7bf325b0202.cn	fe25f54a057217ce7ecdefe8cf325b0202.cn	2	2	0.054	3.44	1.54	95	1.05	1.414	0.938
g6854f4c620415e06da72aa3275786a6e3.tk	g6855f4c620425e06da72aa3285886a6e3.tk	2	4	0.108	3.72	1.54	89	1.11	2.0	0.944
h012e83eedf219925d9e6b3b4ee7f6788b.so	h012e84eeef219925d9e6b3b4fe7f6788b.so	2	3	0.081	3.72	1.54	92	1.08	1.732	1.0
ifcf29747fe797d6db94bb4a286777fc94.cc	ifcf29747fe797d6db94cb4a386777fc94.cc	2	2	0.054	3.43	1.54	95	1.05	1.414	0.933
j3df872db55a3bf2ec78fc54ff85680903.ws	j3df873db65a3bf2ec79fc54ff85681903.ws	2	4	0.108	3.78	1.54	89	1.11	2.0	0.95
kca9453145112cbeb12d512aefb49c0a63.to	kca9453145212cbeb12d512aefb40c0a63.to	2	2	0.054	3.73	1.54	95	1.05	9.055	1.0

Table 4.9: Dyre Dataset Original Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
b1ca5eebd8e0eb8ea6b61eaccbde527c26.ws	b1ca5eebd8e0eb8ea6b61eaccbde527c26.ws	0	0	0.0	3.36	1.54	100	1.00	0.0	1.0
cdca364b71f0c8506d60eb2939f4b806d9.to	cdca364b71f0c8506d60eb2939f4b806d9.to	0	0	0.0	3.82	1.49	100	1.00	0.0	1.0
da3a681bc68fc6db334d1586061fe6731f.in	da3a681bc68fc6db334d1586061fe6731f.in	0	0	0.0	3.56	1.54	100	1.00	0.0	1.0
ecca14cc3f5be1d665cbe8992beddc6c2d.hk	ecca14cc3f5be1d665cbe8992beddc6c2d.hk	0	0	0.0	3.52	1.54	100	1.00	0.0	1.0
fe25f54a057217ce7ecdefe7bf325b0202.cn	fe25f54a057217ce7ecdefe7bf325b0202.cn	0	0	0.0	3.44	1.54	100	1.00	0.0	1.0
g6854f4c620415e06da72aa3275786a6e3.tk	g6854f4c620415e06da72aa3275786a6e3.tk	0	0	0.0	3.72	1.54	100	1.00	0.0	1.0
h012e83eedf219925d9e6b3b4ee7f6788b.so	h012e83eedf219925d9e6b3b4ee7f6788b.so	0	0	0.0	3.72	1.54	100	1.00	0.0	1.0
ifcf29747fe797d6db94bb4a286777fc94.cc	ifcf29747fe797d6db94bb4a286777fc94.cc	0	0	0.0	3.43	1.54	100	1.00	0.0	1.0
j3df872db55a3bf2ec78fc54ff85680903.ws	j3df872db55a3bf2ec78fc54ff85680903.ws	0	0	0.0	3.78	1.54	100	1.00	0.0	1.0
kca9453145112cbeb12d512aefb49c0a63.to	kca9453145112cbeb12d512aefb49c0a63.to	0	0	0.0	3.73	1.54	100	1.00	0.0	1.0

1. Accuracy and Loss/Training and Validation Performance: The analysis of the Dyre dataset in Figure 4.13 yields valuable insights into the model's performance during training and validation phases. In the training phase, the accuracy commences at approximately 50 percent and robustly evolves to reach 100 percent over the span of 50 epochs. Simultaneously, the training loss demonstrates a healthy decline, initiating around 100 percent and steadily reducing to near zero percent by the 50th epoch. Shifting focus to the validation phase, the accuracy exhibits sporadic increments, starting at 60 percent and sporadically surging to nearly 100 percent by the 50th epoch. Concurrently, the validation loss experiences a gradual decrease, starting around 75 percent and steadily diminishing to zero by the 50th epoch. These patterns suggest that the model effectively learns from the Dyre dataset during the

training phase, achieving a high level of accuracy and minimizing loss. The sporadic increases in validation accuracy may indicate some variability in performance, while the steady decrease in validation loss underscores the model's ability to generalize well and optimize its performance over the specified epochs.

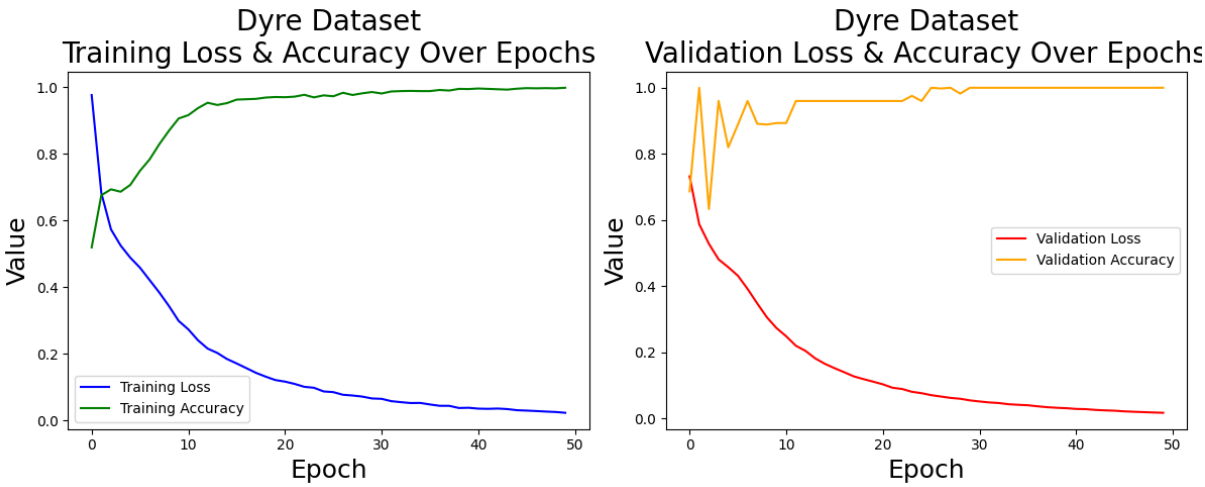

Figure 4.13: Dyre Dataset Training Loss & Accuracy / Validation Loss & Accuracy

2. Confusion Matrix: The Dyre Dataset results reveal in Figure 4.14 insightful outcomes from the classification model trained on initial classes, each comprising 750 lines of data within a CSV file. The dataset was partitioned into training and validation sets, with approximately 80% (600 lines) allocated for training and the remaining 20% (150 lines) for validation. Notably, the original DGA (Domain Generation Algorithm) class demonstrated high recognition, successfully identifying 609 instances. Similarly, the DGA with LRS (Long Range Dependencies and Syntactic Structures) class exhibited strong performance, recognizing 607 instances effectively. However, the DGA with Noise class displayed slightly lower recognition, successfully identifying 584 instances. These results suggest that the model's proficiency varies across different DGA classes, with the original DGA and DGA with LRS classes achieving higher accuracy compared to the DGA with Noise class. Further analysis and refinement of the model may be warranted to enhance its overall performance

and address specific challenges posed by distinct DGA classes.

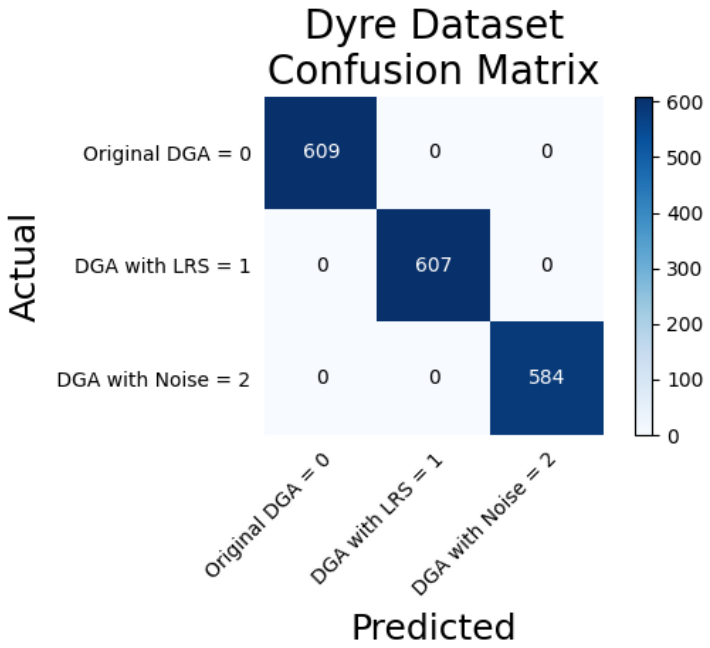

Figure 4.14: Dyre Dataset Confusion Matrix

3. F1 Score and ROC Curve: The exceptional F1 Score and ROC Curve results depicted in Figures 4.15 and 4.16 for the Dyre dataset can be attributed to specific characteristics inherent in the dataset. In this case, the strings being compared are notably longer, approximately 34 alphanumeric characters in length. Additionally, each character in the strings is chosen from a set of 36 options.

   The extended length of the strings introduces a higher degree of complexity to the dataset, allowing the model to discern intricate patterns within a more extensive context. The expansive character space further challenges the model to generalize its learning across a diverse set of possibilities for each character. These factors collectively contribute to the model's exceptional performance, resulting in high F1 Score and ROC Curve metrics.

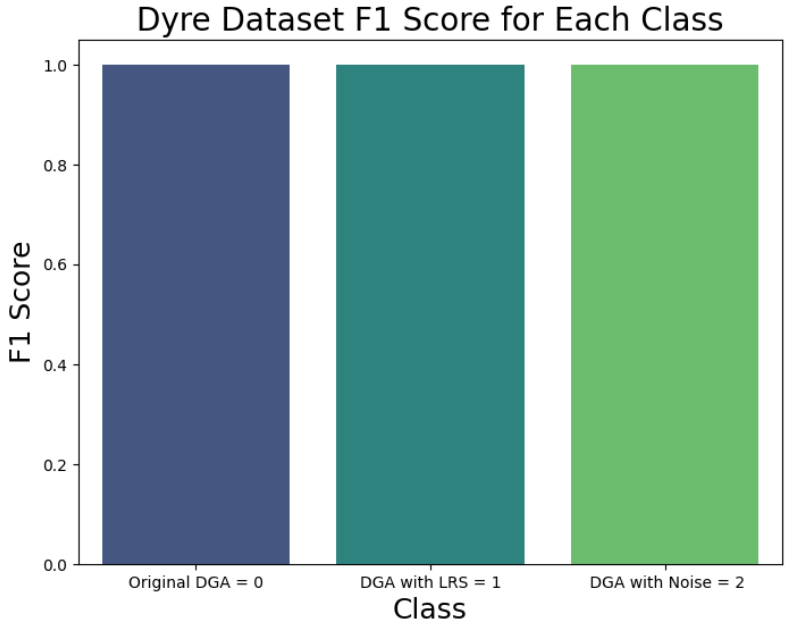

Figure 4.15: Dyre Dataset F1 Score

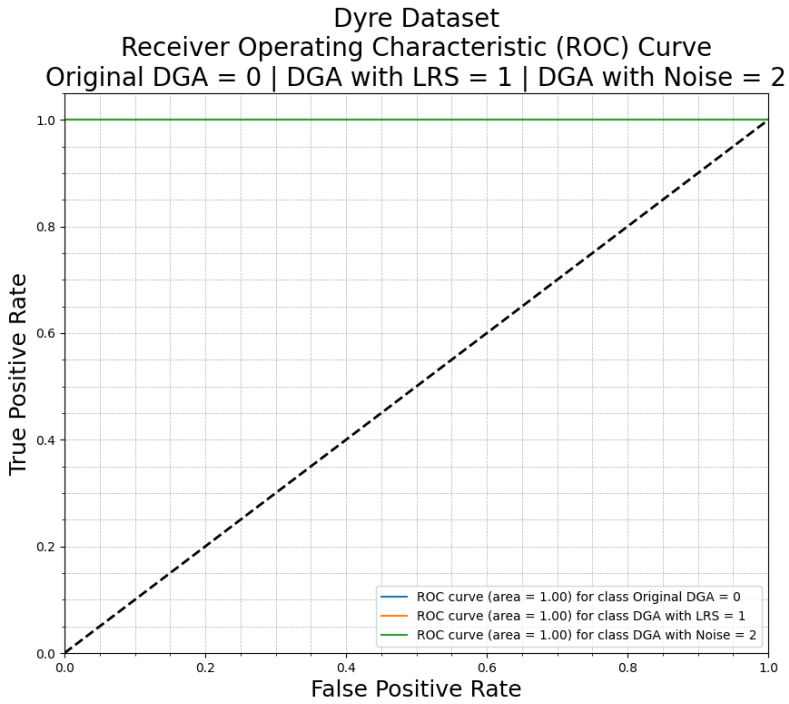

Figure 4.16: Dyre Dataset ROC Curve

re

4.1.2.4 Gameover Dataset

The Gameover dataset tables 4.10 through 4.12 showcase examples classified into three classes: LRS Modification (Class 1), Noise Modification (Class 2), and Original (Class 0). Each table includes details on Initial Domains, Modified Domains, Class assignment, DLD, NDL, Entropy, Compressed, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity. In the LRS Modification class, DLD values range from 22 to 27, signifying a notable number of edit operations in transforming the domains. Normalized DLD is relatively high, ranging from 0.742 to 0.871. Entropy varies from 3.87 to 4.28, indicating diverse levels of unpredictability. Compression values are consistently between 2.05 and 3.00. Similarity and Smith-Waterman Similarity metrics range from 28 to 40 and 0.13 to 1.10, respectively. Euclidean Distance varies from 117.013 to 194.7, and Jaccard Similarity ranges from 0.167 to 0.435. In the Noise Modification class, DLD values are notably lower, ranging from 1 to 3, indicating fewer edit operations and character substitutions. Normalized DLD is lower, varying from 0.032 to 0.103. Entropy values range from 3.87 to 4.28, and Compression is consistently at 1.62 to 1.69. Similarity and Smith-Waterman Similarity metrics are uniformly high at 90 to 97 and 1.03 to 1.10, respectively. Euclidean Distance ranges from 1.0 to 35.355, and Jaccard Similarity varies from 0.905 to 1.0. In the Original class, DLD values are consistently 0, indicating no edit operations. Normalized DLD is 0, and other metrics, including Entropy, Compression, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity, are at their maximum values, indicating identical Initial and Modified Domains. Comparing the LRS Modification, Noise Modification, and Original classes, it is evident that the LRS Modification class involves more complex modifications, resulting in higher DLD and varied similarity metrics. The Noise Modification class exhibits intentional character substitutions with lower DLD and

consistent high similarity metrics. The Original class represents unchanged domains with maximum similarity values. These insights provide an understanding of the characteristics and intentional modifications within the Gameover dataset examples.

Table 4.10: Gameover Dataset LRS Modification Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
1yjm9lf1qwhr9m2bpdtlvsapzj.net	ivfxcraudfcqk.net	1	23	0.767	4.24	2.18	35	1.07	142.51	0.357
54olr41q8dd1y19k60201437kdm.biz	butizu.biz	1	27	0.871	3.96	3.00	40	0.13	117.013	0.167
1gnvxty16g64ali7mxrx1cmjbnh.org	wfhdivqyibiszsx.org	1	25	0.806	4.11	2.05	32	1.10	145.172	0.37
15hg6yw1j60hxt1priykgo7q6f5.com	ywigyhdcxizwaw.com	1	23	0.742	4.11	2.11	33	1.10	162.539	0.4
1psuzh79jgo8a1cv9fk316uj2sm.net	bejxvqsfwezb.net	1	25	0.806	4.28	2.25	38	0.52	129.314	0.31
10qeett454mt92r7widvztieo.org	uuddebfxtgjdyu.org	1	22	0.759	3.97	2.11	33	1.10	194.7	0.269
14rwyc115c6tzw19cy2oj129xvh4.net	cgisrejgrhzigx.net	1	26	0.812	3.87	2.11	39	0.84	152.039	0.435
1hmaw3k18byc08t5j2rm1i6t8ri.com	xqgsirdbydcy.com	1	25	0.806	4.11	2.25	38	0.77	149.359	0.308
yuv2u8djz40i1qo58wq1yz5a8z.org	iegetkxafaijqk.org	1	24	0.8	3.95	2.11	28	0.80	165.641	0.32
4l15waaqjxg71x122evpc4deq.biz	gqqiwhtfastu.biz	1	24	0.828	3.97	2.25	31	1.00	151.291	0.308

Table 4.11: Gameover Dataset Noise Modification Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
1yjm9lf1qwhr9m2bpdtlvsapzj.net	1yjm9lg1qwhr9m2bqdtlvsapzj.net	2	2	0.067	4.24	1.67	93	1.07	1.414	0.917
54olr41q8dd1y19k60201437kdm.biz	54olr41q8dd1y19k60301537ldm.biz	2	3	0.097	3.96	1.65	90	1.10	1.732	0.955
1gnvxty16g64ali7mxrx1cmjbnh.org	1hovxty16g64ali7mxrx1cmjbnh.org	2	2	0.065	4.11	1.65	94	1.06	1.414	1.0
15hg6yw1j60hxt1priykgo7q6f5.com	15hh6zw1j60hxt1priykgo7q6f5.com	2	2	0.065	4.11	1.65	94	1.06	1.414	0.957
1psuzh79jgo8a1cv9fk316uj2sm.net	1psuzh79jho8a1cv9fk316uj2sm.net	2	1	0.032	4.28	1.65	97	1.03	1.0	0.96
10qeett454mt92r7widvztieo.org	11qeeut454mt02r7widvztieo.org	2	3	0.103	3.97	1.69	90	1.07	9.11	0.905
14rwyc115c6tzw19cy2oj129xvh4.net	14rwyc115c6tzw10cy2oj129xvh4.net	2	1	0.031	3.87	1.62	97	1.03	9.0	0.952
1hmaw3k18byc08t5j2rm1i6t8ri.com	1hmaw3k18byc08t5j2rm2i6t8ri.com	2	1	0.032	4.11	1.65	97	1.03	1.0	1.0
yuv2u8djz40i1qo58wq1yz5a8z.org	yuv2u8djz40i1qo58wq1ya5a8a.org	2	2	0.067	3.95	1.67	93	1.07	35.355	1.0
4l15waaqjxg71x122evpc4deq.biz	4l15waaqjxg71x123evqc4deq.biz	2	2	0.069	3.97	1.69	93	1.07	1.414	0.909

Table 4.12: Gameover Dataset Original Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
1yjm9lf1qwhr9m2bpdtlvsapzj.net	1yjm9lf1qwhr9m2bpdtlvsapzj.net	0	0	0.0	4.24	1.67	100	1.00	0.0	1.0
54olr41q8dd1y19k60201437kdm.biz	54olr41q8dd1y19k60201437kdm.biz	0	0	0.0	3.96	1.65	100	1.00	0.0	1.0
1gnvxty16g64ali7mxrx1cmjbnh.org	1gnvxty16g64ali7mxrx1cmjbnh.org	0	0	0.0	4.11	1.65	100	1.00	0.0	1.0
15hg6yw1j60hxt1priykgo7q6f5.com	15hg6yw1j60hxt1priykgo7q6f5.com	0	0	0.0	4.11	1.65	100	1.00	0.0	1.0
1psuzh79jgo8a1cv9fk316uj2sm.net	1psuzh79jgo8a1cv9fk316uj2sm.net	0	0	0.0	4.28	1.65	100	1.00	0.0	1.0
10qeett454mt92r7widvztieo.org	10qeett454mt92r7widvztieo.org	0	0	0.0	3.97	1.69	100	1.00	0.0	1.0
14rwyc115c6tzw19cy2oj129xvh4.net	14rwyc115c6tzw19cy2oj129xvh4.net	0	0	0.0	3.87	1.62	100	1.00	0.0	1.0
1hmaw3k18byc08t5j2rm1i6t8ri.com	1hmaw3k18byc08t5j2rm1i6t8ri.com	0	0	0.0	4.11	1.65	100	1.00	0.0	1.0
yuv2u8djz40i1qo58wq1yz5a8z.org	yuv2u8djz40i1qo58wq1yz5a8z.org	0	0	0.0	3.95	1.67	100	1.00	0.0	1.0
4l15waaqjxg71x122evpc4deq.biz	4l15waaqjxg71x122evpc4deq.biz	0	0	0.0	3.97	1.69	100	1.00	0.0	1.0

1. Accuracy and Loss/Training and Validation Performance: An examination of the Gameover dataset results in Figure 4.17 provides valuable insights into the model's performance throughout the training and validation phases. In the training phase, the accuracy initiates at approximately 55 percent and robustly increases, reaching a perfect 100 percent by the 50th epoch. Concurrently, the training loss undergoes a healthy decline, starting around 95 percent and gradually reducing to near zero percent over the 50 epochs. Transitioning to the validation phase, the accuracy displays sporadic increments, starting at 63 percent and occasionally surging to nearly

100 percent by the 50th epoch. Simultaneously, the validation loss experiences a gradual decrease, starting around 75 percent and steadily diminishing to zero by the 50th epoch. These observed patterns suggest that the model effectively learns from the Gameover dataset during the training phase, achieving high accuracy and minimizing loss. The sporadic increases in validation accuracy may indicate some variability in performance, while the steady decrease in validation loss emphasizes the model's capability to generalize well and optimize its performance throughout the specified epochs.

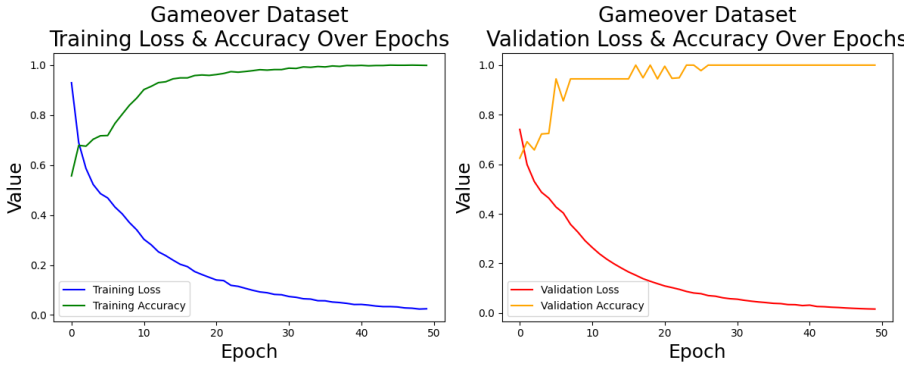

Figure 4.17: Gameover Dataset Training Loss & Accuracy / Validation Loss & Accuracy

2. Confusion Matrix: The results obtained from the Gameover Dataset shed light on the performance of a classification model trained on initial classes, each consisting of 750 lines of data in a CSV file. The dataset was meticulously split, allocating around 80% (600 lines) for training and reserving the remaining 20% (150 lines) for validation. Notably, the original DGA (Domain Generation Algorithm) class exhibited robust recognition, successfully identifying 613 instances. Similarly, the DGA with LRS (Long Range Dependencies and Syntactic Structures) class demonstrated a commendable performance, recognizing 606 instances effectively. However, the DGA with Noise class displayed a slightly lower recognition rate, successfully identifying 581 instances. These results imply varying levels of proficiency within the model across different DGA classes. The original DGA and DGA with LRS

classes showcased higher accuracy compared to the DGA with Noise class, prompting further analysis and potential refinement of the model to optimize its overall performance and address specific challenges posed by individual DGA classes.

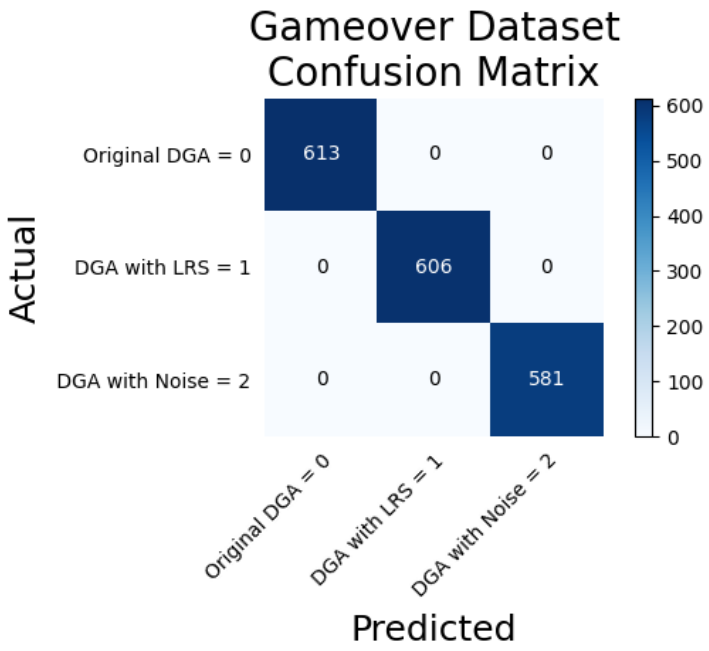

Figure 4.18: Gameover Dataset Confusion Matrix

3. F1 Score and ROC Curve: The remarkable F1 Score and ROC Curve results illustrated in Figures 4.19 and 4.20 for the Gameover dataset can be elucidated by the specific characteristics intrinsic to the dataset. In this instance, the strings undergoing comparison are notably longer, each comprising approximately 34 alphanumeric characters. Additionally, every character in these strings is selected from a set of 36 possible options.

   The extended length of the strings introduces a heightened level of complexity to the dataset, challenging the model to discern intricate patterns within a more extensive context. Moreover, the expansive character space intensifies the model's task by requiring it to generalize its learning across a diverse set of possibilities for each character. These combined factors contribute to the exceptional performance of the

model, resulting in elevated F1 Score and ROC Curve metrics.

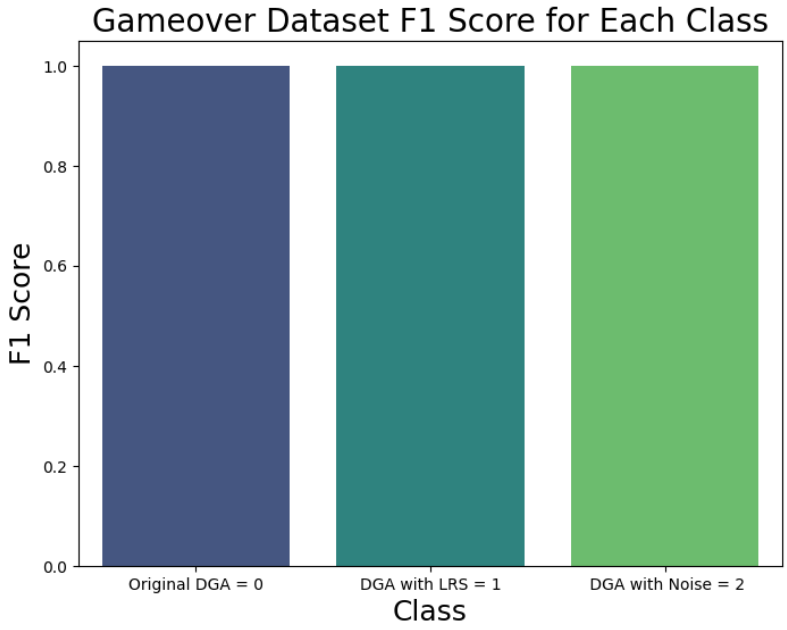

Figure 4.19: Gameover Dataset F1 Score

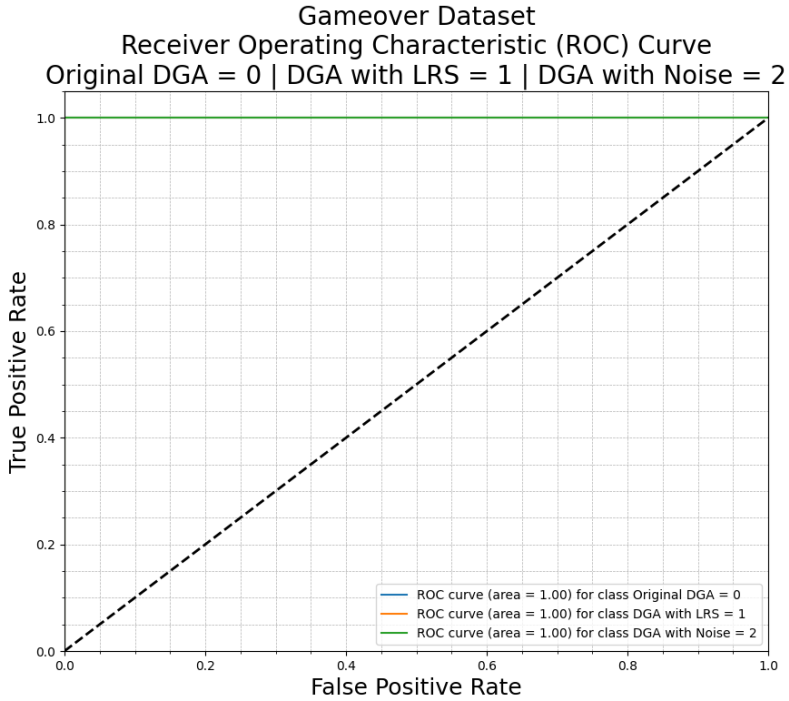

Figure 4.20: Gameover Dataset ROC Curve

The 2D scatter graph results for the Gameover dataset offer a visual representation that aids in understanding the distinct patterns and characteristics of different DNS activities within the dataset. A single purple circle, denoted as the Actual Original, is located near grid coordinates -53, -3, representing benign DNS behavior. On the left side of the grid, three long strands of yellow boxes are annotated as Actual and Predicted DGAs with noise. This arrangement suggests instances of Domain Generation Algorithms (DGAs) exhibiting less structured or potentially noisy patterns in the DNS data. The scattered and elongated nature of these yellow boxes indicates a lack of a well-defined pattern or regularity in the DNS activities associated with DGAs featuring noise.

On the right side of the grid, blue Xs are positioned in a left-to-right loose swarm diagonal pattern, representing Actual and Predicted DGAs with Long Range Dependencies (LRS). This pattern suggests a more organized and intentional structure in the DNS behaviors associated with DGAs with LRS modifications. The loose swarm diagonal arrangement implies a certain level of complexity and intentional design in the DNS activities.

In summary, the 2D scatter graph provides a clear visual separation of different DNS behaviors within the Gameover dataset, enabling the identification of patterns associated with noise and LRS in the context of DGAs.

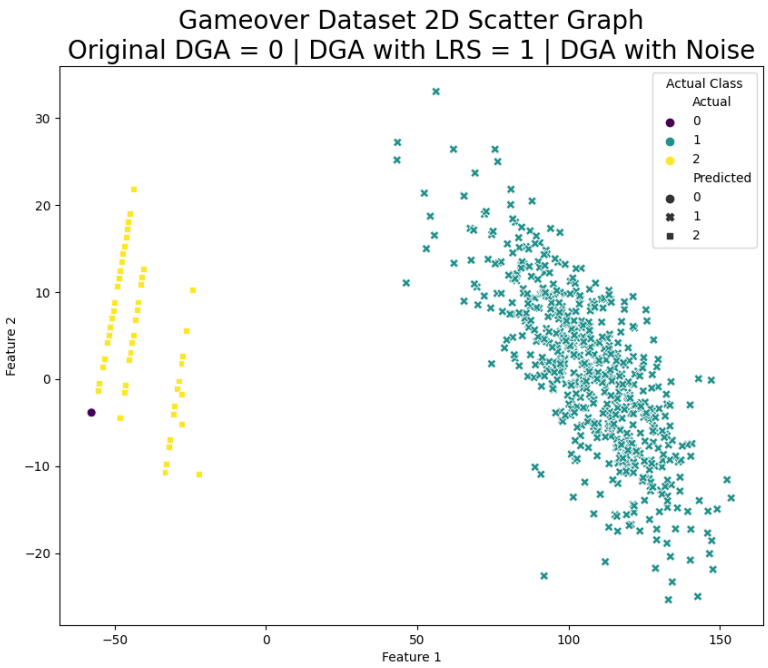

Figure 4.21: Gameover Dataset 2D Scatter

The 3D scatter graph Figure 4.22 results for the Gameover dataset provide a comprehensive view of the DNS activities, revealing distinct patterns and characteristics associated with different types of behaviors. A single blue circle, representing the Actual Original, is positioned near grid coordinates 0, -1, indicating benign DNS behavior. On the left side of the grid, there is a loose grouping of two rows of clustered green circles, annotated as Actual and Predicted DGAs with noise. This arrangement suggests instances of Domain Generation Algorithms (DGAs) that exhibit less structured or potentially noisy patterns in the DNS data. The clustered nature of the green circles implies a concentration of DNS activities associated with DGAs featuring noise, showcasing a distinctive pattern in the left region of the grid.

On the right side of the grid, there is a large vertical cluster of orange circles, denoted as Actual and Predicted DGAs with Long Range Dependencies (LRS). This arrangement suggests a more organized and intentional structure in the DNS behav-

iors associated with DGAs with LRS modifications. The vertical cluster indicates a concentration of DNS activities with specific characteristics associated with DGAs that exhibit Long Range Dependencies.

In summary, the 3D scatter graph provides a comprehensive visualization of DNS behaviors within the Gameover dataset, effectively highlighting the separation of patterns associated with noise and LRSs in the context of DGAs.

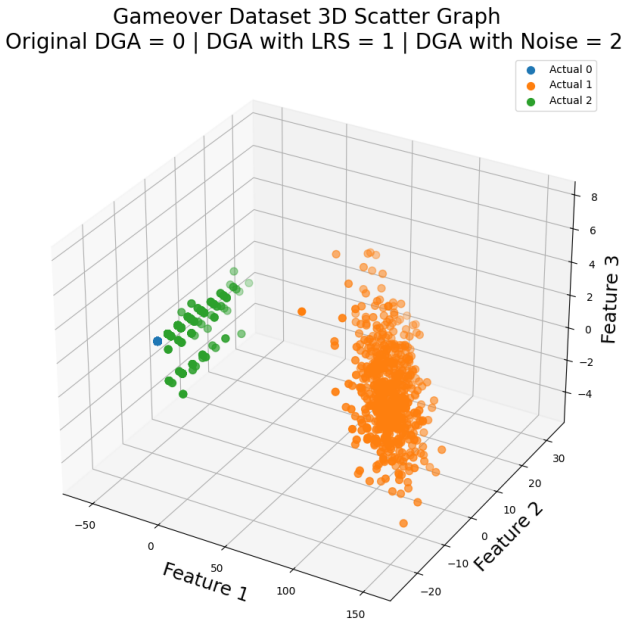

Figure 4.22: Gameover Dataset 3D Scatter

4.1.2.5 Murofetweekly Dataset

The Murofetweekly dataset tables 4.13 through 4.15 provide insights into three classes: LRS Modified (Class 1), Noise Modified (Class 2), and Original (Class 0). Each table contains information on Initial Domains, Modified Domains, Class assignment, DLD, NDL, Entropy, Compressed, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity. In the LRS Modified class, DLD

values range from 31 to 39, indicating a moderate to high number of edit operations in transforming the domains. Normalized DLD varies from 0.738 to 0.886. Entropy ranges from 4.26 to 4.63, reflecting diverse levels of unpredictability. Compression values consistently fall between 1.41 and 1.95. Similarity and Smith-Waterman Similarity metrics range from 26 to 38 and 0.11 to 1.24, respectively. Euclidean Distance varies from 151.539 to 193.822, and Jaccard Similarity ranges from 0.273 to 0.429. In the Noise Modified class, DLD values are notably lower, ranging from 2 to 8, indicating fewer edit operations and character substitutions. Normalized DLD is lower, varying from 0.049 to 0.19. Entropy values range from 4.26 to 4.63, and Compression is consistently at 1.41 to 1.95. Similarity and Smith-Waterman Similarity metrics are uniformly high at 81 to 95 and 1.02 to 1.19, respectively. Euclidean Distance ranges from 1.414 to 9.381, and Jaccard Similarity varies from 0.8 to 0.966. In the Original class, DLD values are consistently 0, indicating no edit operations. Normalized DLD is 0, and other metrics, including Entropy, Compression, Similarity, Smith-Waterman Similarity, Euclidean Distance, and Jaccard Similarity, are at their maximum values, indicating identical Initial and Modified Domains. Comparing the LRS Modified, Noise Modified, and Original classes, it is evident that the LRS Modified class involves more complex modifications, resulting in higher DLD and varied similarity metrics. The Noise Modified class exhibits intentional character substitutions with lower DLD and consistent high similarity metrics. The Original class represents unchanged domains with maximum similarity values. These insights provide an understanding of the characteristics and intentional modifications within the Murofetweekly dataset examples.

Table 4.13: Murofetweekly Dataset LRS Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
ovpzl48k57fvhycrcscuatosp42c39n50jxhx.com	fajzvgxisbrbsdqdcarhyi.com	1	32	0.78	4.56	1.77	35	1.12	153.935	0.424
mzbtfza57bygzbrl28d60a57o41d20drowc19h14.net	jsdvjqsivjrbtqutbfsy.net	1	39	0.886	4.45	1.83	26	1.07	151.539	0.273
ouoso31aymwp62l38dsmwi65j16f42hxiuj66or.org	ecphgaucgvwxhxdzc.org	1	35	0.814	4.32	1.95	29	1.00	159.54	0.429
f32mriyl68lskteydxe61jvg33p12o21psmso11.info	vbyibduiuiuzfwabb.info	1	37	0.841	4.26	1.82	27	0.11	162.398	0.276
c39e41dqhznzo11m49gqbudyhym69bqgqpzl58.biz	sutykkwretiyhrawaj.biz	1	36	0.857	4.29	1.91	32	0.83	173.11	0.258
m59a67huhta27cziskybuixlsb38h64iuaql28.ru	qxdyeqsjybziseyhbsxydpa.ru	1	32	0.78	4.38	1.77	38	0.80	163.003	0.414
c19gwcukvh54gyftk37dxowl68gqe51bumwd30.com	swgsezgywivdzuhgwurdgu.com	1	31	0.738	4.63	1.77	38	1.12	188.457	0.375
bto21oynrn50dyhypsn60oya27nufvjtewhv.info	rdictixiabiqdvfzdtfxg.info	1	32	0.78	4.32	1.77	31	1.20	181.538	0.367
ivfzoudun10o61budzi65d10i25mug33prgxk47.org	yfwjetdreukytxewacwiz.org	1	37	0.86	4.36	1.80	29	0.79	193.822	0.355
l18htaxktcyitfxn30puguc39i55ete31fvj46.biz	xdqhzerixdvhdwesxuduwf.biz	1	34	0.81	4.49	1.77	27	1.24	157.487	0.303

Table 4.14: Murofetweekly Dataset Noise Modified Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
ovpzl48k57fvhycrcscuatosp42c39n50jxhx.com	pvpzl48k67fvhycrcscuatosp42c49n50jxhx.com	2	3	0.073	4.56	1.49	93	1.02	1.732	0.931
mzbtfza57bygzbrl28d60a57o41d20drowc19h14.net	mzbufza57bygzbrl28d60a67o51d30drowc19h14.net	2	4	0.091	4.45	1.45	91	1.09	2.0	0.931
ouoso31aymwp62l38dsmwi65j16f42hxiuj66or.org	ouoso31azmwp62l38dsmwi65j16f52hyiuj66or.org	2	3	0.07	4.32	1.47	93	1.07	1.732	0.885
f32mriyl68lskteydxe61jvg33p12o21psmso11.info	f32nsiyl68lsktezdye61jvg33p12o21psnso21.info	2	6	0.136	4.26	1.41	86	1.14	2.449	0.84
c39e41dqhznzo11m49gqbudyhym69bqgqpzl58.biz	c49f41dqhznzo11m49grbueyhym60cqgrpzl68.biz	2	8	0.19	4.29	1.48	81	1.19	9.381	0.815
m59a67huhta27cziskybuixlsb38h64iuaql28.ru	m59a67huhta27cziskybuixmsb38h65iuaql28.ru	2	2	0.049	4.38	1.49	95	1.05	1.414	0.96
c19gwcukvh54gyftk37dxowl68gqe51bumwd30.com	c29gwcvkvh54gyftk47dxowl68gqe51bumxd30.com	2	4	0.095	4.63	1.48	90	1.10	2.0	0.966
bto21oynrn50dyhypsn60oya27nufvjtewhv.info	bto21ozoro60dyhypsn60oya27nufvjtexhw.info	2	6	0.146	4.32	1.49	85	1.15	2.449	0.889
ivfzoudun10o61budzi65d10i25mug33prgxk47.org	ivgzoueun10o61budzi65d10i25mug43prgxk48.org	2	4	0.093	4.36	1.47	91	1.09	2.0	0.846
l18htaxktcyitfxn30puguc39i55ete31fvj46.biz	m19htaxktcyitfxn30puguc39i55ete31fvj57.biz	2	4	0.095	4.49	1.48	90	1.05	2.0	0.8

Table 4.15: Murofetweekly Dataset Original Example

Initial Domains	Modified Domains	Class	DLD	Normalized	Entropy	Compressed	Similarity	Smith Waterman Similarity	Euclidean Distance	Jaccard Similarity
ovpzl48k57fvhycrcscuatosp42c39n50jxhx.com	ovpzl48k57fvhycrcscuatosp42c39n50jxhx.com	0	0	0.0	4.56	1.49	100	1.00	0.0	1.0
mzbtfza57bygzbrl28d60a57o41d20drowc19h14.net	mzbtfza57bygzbrl28d60a57o41d20drowc19h14.net	0	0	0.0	4.45	1.41	100	1.00	0.0	1.0
ouoso31aymwp62l38dsmwi65j16f42hxiuj66or.org	ouoso31aymwp62l38dsmwi65j16f42hxiuj66or.org	0	0	0.0	4.32	1.47	100	1.00	0.0	1.0
f32mriyl68lskteydxe61jvg33p12o21psmso11.info	f32mriyl68lskteydxe61jvg33p12o21psmso11.info	0	0	0.0	4.26	1.45	100	1.00	0.0	1.0
c39e41dqhznzo11m49gqbudyhym69bqgqpzl58.biz	c39e41dqhznzo11m49gqbudyhym69bqgqpzl58.biz	0	0	0.0	4.29	1.48	100	1.00	0.0	1.0
m59a67huhta27cziskybuixlsb38h64iuaql28.ru	m59a67huhta27cziskybuixlsb38h64iuaql28.ru	0	0	0.0	4.38	1.49	100	1.00	0.0	1.0
c19gwcukvh54gyftk37dxowl68gqe51bumwd30.com	c19gwcukvh54gyftk37dxowl68gqe51bumwd30.com	0	0	0.0	4.63	1.48	100	1.00	0.0	1.0
bto21oynrn50dyhypsn60oya27nufvjtewhv.info	bto21oynrn50dyhypsn60oya27nufvjtewhv.info	0	0	0.0	4.32	1.49	100	1.00	0.0	1.0
ivfzoudun10o61budzi65d10i25mug33prgxk47.org	ivfzoudun10o61budzi65d10i25mug33prgxk47.org	0	0	0.0	4.36	1.47	100	1.00	0.0	1.0
l18htaxktcyitfxn30puguc39i55ete31fvj46.biz	l18htaxktcyitfxn30puguc39i55ete31fvj46.biz	0	0	0.0	4.49	1.48	100	1.00	0.0	1.0

1. Accuracy and Loss/Training and Validation Performance: The outcomes from the analysis of the Murofetweekly dataset unveil in Figure 4.23 significant trends in the model's performance across the training and validation phases. In the training phase, the accuracy initiates at approximately 55 percent and undergoes a healthy growth, eventually achieving a perfect 100 percent by the 50th epoch. Concurrently, the training loss demonstrates a robust decline, commencing around 95 percent and gradually reducing to near zero percent over the course of 50 epochs.

   Shifting focus to the validation phase, the accuracy displays a dynamic pattern. It sporadically increases from an initial 95 percent, experiences some fluctuations until about epoch 10, and then consistently progresses to reach nearly 100 percent accuracy by the 50th epoch. Simultaneously, the validation loss exhibits a gradual

decrease, starting around 75 percent and steadily diminishing to zero by the 50th epoch.

These findings collectively suggest that the model effectively learns from the Murofetweekly dataset during the training phase, achieving a substantial increase in accuracy while minimizing loss. The sporadic increments in validation accuracy may indicate some variability in performance during the early stages, but the model stabilizes and consistently improves over time. The gradual decrease in validation loss underscores the model's ability to generalize well to new data and optimize its overall performance throughout the specified 50 epochs.

Figure 4.23: Murofetweekly Dataset Training Loss & Accuracy / Validation Loss & Accuracy

2. Confusion Matrix: The results from the Murofetweekly Dataset offer insights into the performance of a classification model trained on initial classes, each comprising 750 lines of data within a CSV file. The dataset was systematically partitioned, with approximately 80% (600 lines) allocated for training and the remaining 20% (150 lines) reserved for validation. Notably, the original DGA (Domain Generation Algorithm) class demonstrated a solid performance, successfully recognizing 600 instances. Similarly, the DGA with LRS (Long Range Dependencies and Syntactic Structures) class displayed commendable recognition, successfully identifying 588 instances. Intriguingly, the DGA with Noise class outperformed the others, achiev-

ing the highest recognition with 612 instances successfully identified. These results suggest that the model exhibits varying levels of proficiency across different DGA classes. While the original DGA and DGA with LRS classes showed robust performance, the DGA with Noise class surpassed them, indicating potential nuances in the characteristics of the datasets. Further analysis and refinement may be necessary to optimize the model's accuracy and address specific challenges posed by distinct DGA classes within the Murofetweekly Dataset.

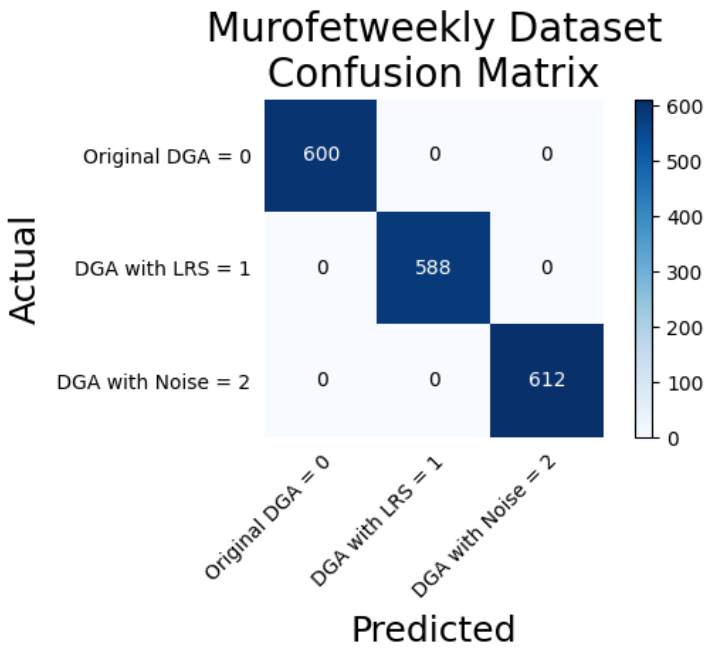

Figure 4.24: Murofetweekly Dataset Confusion Matrix

3. F1 Score and ROC Curve: The impressive F1 Score and ROC Curve results depicted in Figures 4.25 and 4.26 for the Murofetweekly dataset can be attributed to the unique characteristics of the dataset itself. In this scenario, the strings subjected to comparison are notably longer, ranging approximately between 31 to 39 alphanumeric characters. Furthermore, each character in these strings is chosen from a set of 36 possible options.

   The increased length of the strings introduces a heightened level of complexity to the

dataset, challenging the model to discern intricate patterns within a more extensive context. Moreover, the expansive character space intensifies the model's task by requiring it to generalize its learning across a diverse set of possibilities for each character. These combined factors contribute to the exceptional performance of the model, resulting in elevated F1 Score and ROC Curve metrics.

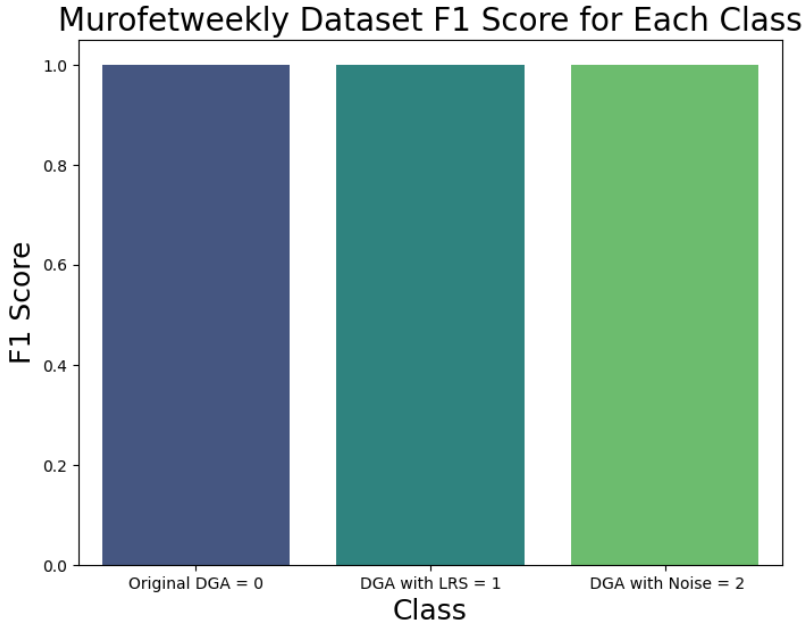

Figure 4.25: Murofetweekly Dataset F1 Score

4. ROC Curve: Review the Receiver Operating Characteristic (ROC) curve to assess the trade-off between true and false positive rates. AUC (Area Under the Curve) can quantify the model's discriminative ability.

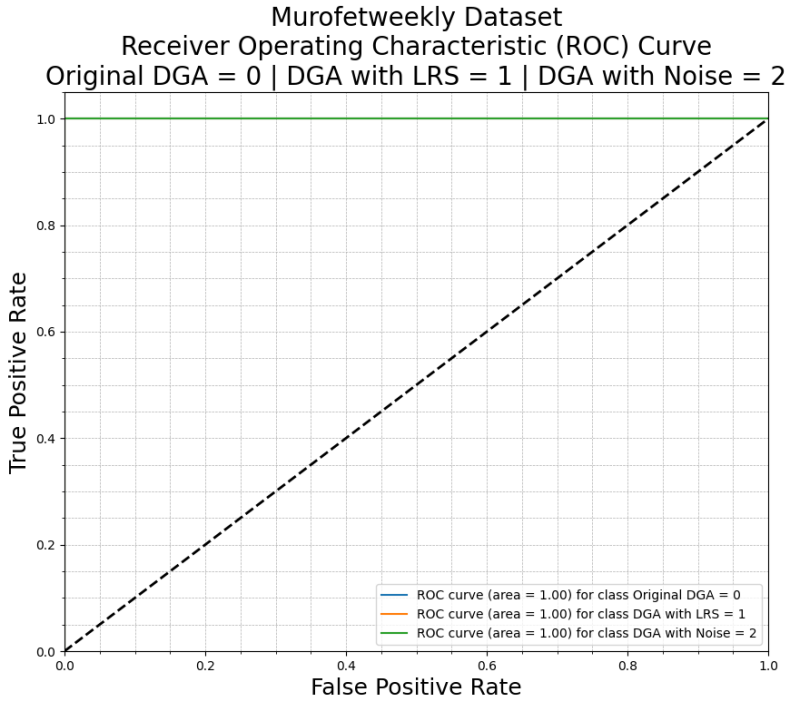

Figure 4.26: Murofetweekly Dataset ROC Curve

The 2D scatter graph Figure 4.27 results for the MurofetWeekly dataset showcase distinct patterns and clusters, aiding in the interpretation of various DNS activities. A single purple circle, situated near grid coordinates -55, -3, represents actual original data points associated with DNS behavior. On the left side of the grid, three long strands of yellow boxes, annotated as Actual and Predicted DGAs with noise, suggest instances of Domain Generation Algorithms (DGAs) exhibiting less structured patterns or possibly noise in the DNS data. Conversely, on the right side of the grid, a series of blue Xs, denoting Actual and Predicted DGAs with Long Range Dependencies (LRS), are arranged in a left-to-right swarm diagonal pattern. This organized distribution implies a more systematic and sophisticated pattern in the DNS activities, potentially indicating a higher level of complexity and intentional design. The visual representation of the MurofetWeekly dataset through this scatter graph facilitates the identification and differentiation of various DNS behaviors, enabling further analysis of distinct patterns associated with different types of DNS

activities.

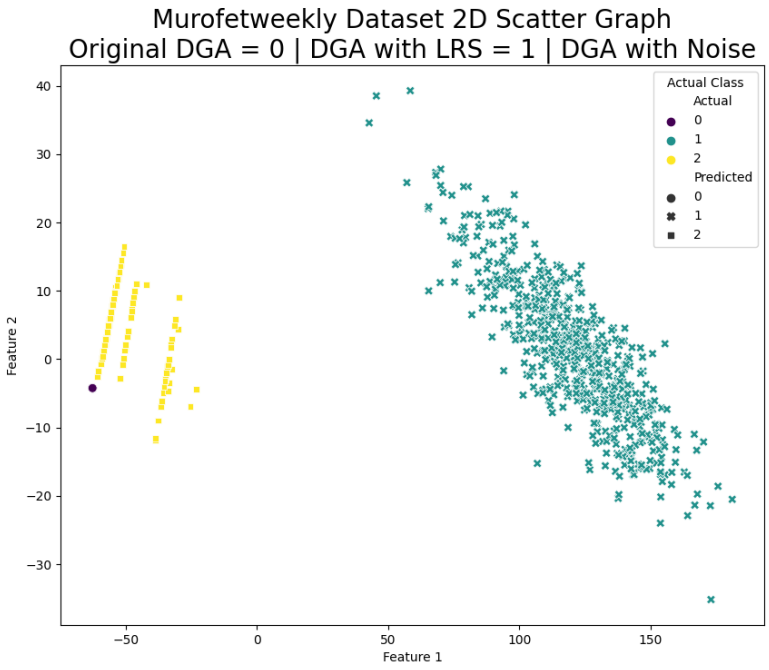

Figure 4.27: Murofetweekly Dataset 2D Scatter

The 3D scatter graph Figure 4.27 results for the MurofetWeekly dataset provide a comprehensive visualization of different DNS activities, allowing for a nuanced understanding of the dataset's characteristics. A single blue circle positioned near grid coordinates 0, -1 represents actual original data points associated with benign DNS behavior. On the left side of the grid, a loose grouping of green circles, annotated as Actual and Predicted DGAs with noise, forms two rows, suggesting instances of Domain Generation Algorithms (DGAs) exhibiting less structured or potentially noisy patterns in the DNS data.

On the right side of the grid, a large vertical cluster of orange circles, denoting Actual and Predicted DGAs with LRS, indicates a distinct and organized pattern in the DNS activities. This clustered arrangement suggests a higher level of complexity and intentional design in the DNS behaviors associated with DGAs featuring

LRSs. Overall, the 3D scatter graph provides a visual representation that facilitates the identification and differentiation of various DNS behaviors within the MurofetWeekly dataset, offering valuable insights into the nature and structure of DNS activities associated with differences between DGAs with no modifications and ones with noise or LRS modifications.

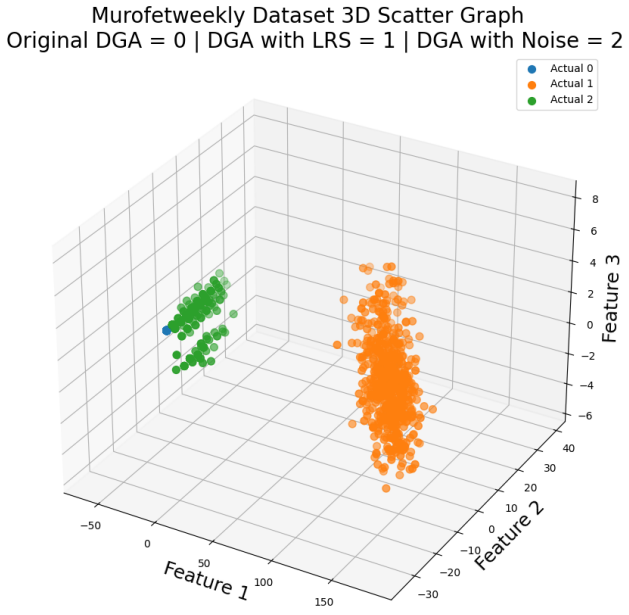

Figure 4.28: Murofetweekly Dataset 3D Scatter

Table 4.16: Datasets Reviewed

Dataset	Class	Precision	Recall	F1-Score	Support
Banjori	Original DGA = 0	1.00	1.00	1.00	596
	DGA with LRS = 1	1.00	1.00	1.00	597
	DGA with Noise = 2	1.00	1.00	1.00	607
	Accuracy			1.00	1800
	Macro Avg	1.00	1.00	1.00	1800
	Weight Avg	1.00	1.00	1.00	1800
Dnschanger	Original DGA = 0	1.00	1.00	1.00	591
	DGA with LRS = 1	1.00	1.00	1.00	591
	DGA with Noise = 2	1.00	1.00	1.00	618
	Accuracy			1.00	1800
	Macro Avg	1.00	1.00	1.00	1800
	Weight Avg	1.00	1.00	1.00	1800
Dyre	Original DGA = 0	1.00	1.00	1.00	609
	DGA with LRS = 1	1.00	1.00	1.00	607
	DGA with Noise = 2	1.00	1.00	1.00	584
	Accuracy			1.00	1800
	Macro Avg	1.00	1.00	1.00	1800
	Weight Avg	1.00	1.00	1.00	1800
Gameover	Original DGA = 0	1.00	1.00	1.00	613
	DGA with LRS = 1	1.00	1.00	1.00	606
	DGA with Noise = 2	1.00	1.00	1.00	581
	Accuracy			1.00	1800
	Macro Avg	1.00	1.00	1.00	1800
	Weight Avg	1.00	1.00	1.00	1800
Murfetweekly	Original DGA = 0	1.00	1.00	1.00	600
	DGA with LRS = 1	1.00	1.00	1.00	588
	DGA with Noise = 2	1.00	1.00	1.00	612
	Accuracy			1.00	1800
	Macro Avg	1.00	1.00	1.00	1800
	Weight Avg	1.00	1.00	1.00	1800

Table 4.17: Classification Reports by Accuracy

	DGA		Original DGA = 0			DGA with LRS = 1			DGA with Noise = 2			Data
	Datasets	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Size
	bamital	1	1	1	1	1	1	1	1	1	1	1800
	banjori	1	1	1	1	1	1	1	1	1	1	1800
	bedep	1	1	1	1	1	1	1	1	1	1	1800
	beebone	1	1	1	1	1	1	1	1	1	1	501
	blackhole	1	1	1	1	1	1	1	1	1	1	1800
	bobax	1	1	1	1	1	1	1	1	1	1	1800
	chinad	1	1	1	1	1	1	1	1	1	1	1800
	conficker	1	1	1	1	1	1	1	1	1	1	1800
	corebot	1	1	1	1	1	1	1	1	1	1	1800
	diamondfox	1	1	1	1	1	1	1	1	1	1	1800
	dircrypt	1	1	1	1	1	1	1	1	1	1	1800
	dnschanger	1	1	1	1	1	1	1	1	1	1	1800
	downloader	1	1	1	1	1	1	1	1	1	1	285
	dyre	1	1	1	1	1	1	1	1	1	1	1800
	ebury	1	1	1	1	1	1	1	1	1	1	1800
	ekforward	1	1	1	1	1	1	1	1	1	1	1800
	emotet	1	1	1	1	1	1	1	1	1	1	1800
	fobber	1	1	1	1	1	1	1	1	1	1	1800
	gameover	1	1	1	1	1	1	1	1	1	1	1800
	gameoverp2p	1	1	1	1	1	1	1	1	1	1	1800
	gozi	1	1	1	1	1	1	1	1	1	1	1800
	goznym	1	1	1	1	1	1	1	1	1	1	871
	locky	1	1	1	1	1	1	1	1	1	1	1800
	matsnu	1	1	1	1	1	1	1	1	1	1	1800
	mirai	1	1	1	1	1	1	1	1	1	1	1800
	modpack	1	1	1	1	1	1	1	1	1	1	1024
	murofet	1	1	1	1	1	1	1	1	1	1	1800
	murofetweekly	1	1	1	1	1	1	1	1	1	1	1800
	mydoom	1	1	1	1	1	1	1	1	1	1	1800
	necurs	1	1	1	1	1	1	1	1	1	1	1800
	nymaim	1	1	1	1	1	1	1	1	1	1	1800
	nymaim2	1	1	1	1	1	1	1	1	1	1	1800
	padcrypt	1	1	1	1	1	1	1	1	1	1	1800
	pandabanker	1	1	1	1	1	1	1	1	1	1	1800
	pitou	1	1	1	1	1	1	1	1	1	1	1800
	proslikefan	1	1	1	1	1	1	1	1	1	1	1800
	pushdo	1	1	1	1	1	1	1	1	1	1	1800
	pushdotid	1	1	1	1	1	1	1	1	1	1	1800
	pykspa	1	1	1	1	1	1	1	1	1	1	1800
	pykspa2	1	1	1	1	1	1	1	1	1	1	1800
	qadars	1	1	1	1	1	1	1	1	1	1	1800
	qakbot	1	1	1	1	1	1	1	1	1	1	1800
	qsnatch	1	1	1	1	1	1	1	1	1	1	1800
	ramdo	1	1	1	1	1	1	1	1	1	1	1800
	ramnit	1	1	1	1	1	1	1	1	1	1	1800
	ranbyus	1	1	1	1	1	1	1	1	1	1	1800
	rovnix	1	1	1	1	1	1	1	1	1	1	1800
	sedhgUKM9	1	1	1	1	1	1	1	1	1	1	1800
	shifu	1	1	1	1	1	1	1	1	1	1	1800
	simda	1	1	1	1	1	1	1	1	1	1	1800

Table 4.18: Classification Reports by Accuracy Continued

DGA		Original DGA = 0			DGA with LRS = 1			DGA with Noise = 2			Data
Datasets	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Size
sisron	1	1	1	1	1	1	1	1	1	1	1800
sphinx	1	1	1	1	1	1	1	1	1	1	1800
suppobox	1	1	1	1	1	1	1	1	1	1	1800
sutra	1	1	1	1	1	1	1	1	1	1	1800
symmi	1	1	1	1	1	1	1	1	1	1	1800
szribi	1	1	1	1	1	1	1	1	1	1	1800
tempedrevetdd	1	1	1	1	1	1	1	1	1	1	1800
tinba	1	1	1	1	1	1	1	1	1	1	1800
tofsee	1	1	1	1	1	1	1	1	1	1	1800
torpig	1	1	1	1	1	1	1	1	1	1	1800
ud2	1	1	1	1	1	1	1	1	1	1	1800
vawtrak	1	1	1	1	1	1	1	1	1	1	1800
vidro	1	1	1	1	1	1	1	1	1	1	1800
vidrotid	1	1	1	1	1	1	1	1	1	1	1800
virut	1	1	1	1	1	1	1	1	1	1	1800
wd	1	1	1	1	1	1	1	1	1	1	1800
xxhex	1	1	1	1	1	1	1	1	1	1	1800
madmax	0.99	1	1	1	0.98	1	0.99	1	0.98	0.99	1149
makloader	0.99	0.96	1	0.98	1	1	1	1	0.96	0.99	1226
feodo	0.92	0.79	1	0.88	1	1	1	1	0.77	0.87	458
ud4	0.89	0.75	1	0.86	1	1	1	1	0.66	0.79	237
volatilecedar	0.88	0.74	1	0.85	1	1	1	1	0.65	0.79	1192
Avg	0.995	0.989	1	0.994	1	1	1	1	0.986	0.992	1671

Chapter 5

Conclusion

5.1 Contributions

This section covers contributions thorough evaluation of Domain Generating Algorithm (DGA) datasets, analyzing their performance in various scenarios. Notably, consistent high accuracy and robust metrics are observed, with resilience to challenges in specific datasets. The subsequent section distills key lessons in cybersecurity, highlighting the dynamic evolution of malware tactics and the crucial role of adaptive defense against Domain Generating Algorithms (DGAs), facilitated by machine learning. The dissertation outlines future research directions, including multi-classifications of LRS-modified DGAs, variations in LRS starting points, assessments of different LRS versions, and visual identification through 2D modeling and neural networks. The dissertation provides insights into cybersecurity challenges and charts a roadmap for future research and defense strategies.

The contributions within this research extend to the application of specific techniques, including DLD and NDL feature creation, including noise and LRS obfuscation, which further enhance the robustness and effectiveness of the models in combating Domain Generating Algorithm (DGA) challenges.

Within the domain of natural language processing tasks, the utilization of DLD and NDL features emerges as the optimal choice for feedforward neural networks, mainly when tasked with discerning modifications within Domain Generating Algorithms (DGAs). These features excel by offering a robust framework to measure sequence similarity, taking into account a comprehensive set of operations, including insertions, deletions, substitu-

tions, and crucially, transpositions of adjacent characters. Their superiority is particularly evident in their adeptness at detecting transpositions, a nuanced evasion strategy often employed by DGAs to mimic legitimate domains. The inclusion of DL Normalization further solidifies their prowess by enhancing the comparability of strings, especially crucial when handling varied string lengths. The adaptability of these features in accommodating the diverse modifications inherent in DGA-generated domains establishes them as the best choice, fortified by their historical success in diverse applications. Their effectiveness, validated over time, makes a compelling case for their superiority, offering a comprehensive solution for identifying patterns within DGAs and underscoring their indispensability in the realm of neural network-based analysis.

Employing noise and Linear Recursive Sequence (LRS) obfuscation techniques within the context of Feedforward Neural Networks for training on Domain Generating Algorithms (DGAs) offers the best and most effective strategy. Noise injection involves introducing random variations to the input data, and when applied to DGAs, it enhances the network's resilience against overfitting by introducing variability into the training set. This is crucial when dealing with DGAs, as they often exhibit diverse patterns and generate domains with subtle variations. The inclusion of noise aids in creating a more generalized model that can better adapt to the dynamic nature of DGA-generated domains.

Linear Recursive Sequence obfuscation involves utilizing mathematical functions to generate sequences, adding an additional layer of complexity to the input data. In the context of DGAs, LRS obfuscation introduces a controlled form of complexity that mimics the unpredictable nature of algorithmically generated domains. This enhances the network's ability to capture intricate patterns and improves its robustness against adversarial attacks that may attempt to exploit identifiable patterns within the data.

Incorporating noise and LRS obfuscation within the training process enables feedforward neural networks to learn more effectively from diverse and complex DGA datasets.

These techniques contribute to the model's adaptability, generalization, and resistance to overfitting, making them well-suited for training on DGAs where variability and obfuscation are inherent challenges. The synergy between these techniques and feedforward neural networks reinforces the model's capacity to discern subtle modifications within the domains generated by DGAs, ultimately leading to a more robust and accurate detection system.

5.1.1 Final Results

This section provides detailed insights into the classification results for various Domain Generation Algorithm (DGA) datasets. The tables present comprehensive metrics, including accuracy, precision, recall, and F1-score, which thoroughly evaluate the model's performance on different DGAs. The classification reports are categorized by DGA type, with each table focusing on a specific DGA variation. Three variations in classes are considered: Original DGA (0), DGA with Long Range Sequences (LRS) (1), and DGA with Noise (2). Each dataset's performance is assessed across these DGA variations.

Tables include datasets like bamital, dyre, gameover, and others, evaluated under different DGA scenarios. These results provide valuable insights into the model's efficacy in distinguishing benign and malicious domains across diverse DGA patterns.

The table comprehensively assesses the performance metrics for various Domain Generation Algorithm (DGA) datasets. Each dataset undergoes evaluation under three distinct scenarios: original DGA, DGA with (LRS), and DGA with Noise. The metrics considered include Accuracy, Precision, Recall, and F1-Score. Across the board, the datasets generally demonstrate high accuracy, ranging from 0.89 to a perfect score of 1. This indicates the effectiveness of the models in correctly categorizing domains generated by DGAs. Precision, Recall, and F1-Score consistently exhibit commendable values, reflecting the robustness of the models' performance. Most of the datasets achieve flawless accuracy in all scenarios, highlighting exceptional model accuracy. Conversely, datasets like 'Omexo'

or 'Volatilecedar' exhibit lower precision, recall, and F1-Score values, suggesting potential challenges in accurate classification for these instances due to domain name issues and dataset sample size. As the datasets progress further down the table and decrease in size, the accuracy decreases. Noteworthy is the models' ability to maintain consistent performance across scenarios, including DGA with LRS and DGA with Noise, underscoring their resilience to variations introduced by long-range dependencies and noise in the datasets. In summary, the outcomes presented in the table underscore the overall strength of the models in effectively classifying DGA domains, with observed challenges in specific datasets mitigated by the models' adaptability to variations.

5.2 Lessons Learned

5.2.1 Understanding the Evolution of Malware Tactics

Recognizing the historical context of Domain Generating Algorithms (DGAs) is crucial. The evolution from static blacklists to dynamic, dynamically generated domains highlights the adaptability of malicious actors in response to cybersecurity measures.

5.2.2 Sophistication of Malicious Software

Malware strains like Conficker, Emotet, and GameOverZeus showcase the multifaceted capabilities of malicious software. Understanding their persistence, propagation methods, and adaptability is vital for developing effective defense strategies.

5.2.3 Cat-and-Mouse Game in Cybersecurity

The dynamic evolution of Domain Generation Algorithms (DGAs) underscores the crucial need for cybersecurity professionals to adopt adaptive and proactive defense mechanisms. The ongoing cat-and-mouse game between defenders and attackers emphasizes the impor-

tance of continuous vigilance to stay ahead of rapidly evolving tactics. Traditional static security measures are insufficient in countering the sophistication of modern cyber threats, particularly those leveraging DGAs. To effectively mitigate risks, organizations must embrace agile security strategies, incorporating advanced threat intelligence, machine learning, and artificial intelligence for real-time threat detection and response. Additionally, fostering collaboration and information-sharing within the cybersecurity community is essential to create a unified defense against the ever-changing tactics employed by cyber adversaries.

5.2.4 Challenges in DGA Defense

Protecting against DGAs involves addressing challenges such as the dynamic and evolving nature of DGAs, the large number of potential domains, algorithm variability, fast flux networks, false positives, encryption, tunneling, and resource-intensive analysis. A comprehensive defense requires a multi-faceted approach.

5.2.5 Role of Machine Learning

Machine learning, particularly Feedforward Neural Networks (FNNs), plays a pivotal role in combating DGAs. ML models, trained on extensive datasets, can effectively discern patterns and anomalies in domain generation, contributing to enhanced threat detection.

5.2.6 Complexity of Linear Recursive Sequences (LRS)

Linear Recursive Sequences (LRS) introduce complexity to DGAs, posing challenges in detection. Understanding the mathematical properties and behaviors of LRS-based DGAs is crucial for developing advanced defense strategies.

5.2.7 Interdisciplinary Applications of LRSs

Linear Recursive Sequences find applications in various fields, including computer science, signal processing, and cryptography. Recognizing their interdisciplinary significance provides a broader perspective on their implications in the context of cybersecurity.

5.2.8 Machine Learning Challenges and Future Directions

Acknowledging challenges in machine learning, such as model interpretability, bias, and ethical considerations, highlights the ongoing need for research to address these issues. Responsible deployment and continuous improvement are essential in leveraging machine learning effectively.

5.2.9 Versatility of Feedforward Neural Networks

FNNs, as universal approximators, demonstrate versatility across many different applications, including pattern recognition, image processing, and natural language understanding. Their simplicity and effectiveness contribute to their widespread use in machine learning.

5.2.10 Ongoing Evolution in Cybersecurity Practices

The dynamic nature of cyber threats, as exemplified by evolving DGAs, emphasizes the need for ongoing research and adaptation in cybersecurity practices. Staying informed about emerging tactics and technologies is critical for effective defense.

5.3 Future Work

5.3.1 2D FNN Modeling

Another avenue for exploration involves extending the identification of Linear Recursive Sequences (LRS) modified Domain Generating Algorithms (DGAs) by incorporating visual identification through 2D modeling and neural networks. Leveraging advanced neural network architectures for visual pattern recognition, particularly in 2D modeling, could provide a novel approach to discerning subtle variations in LRS-modified DGAs. This entails exploring the use of Convolutional Neural Networks (CNNs) or similar visual processing models to analyze patterns in the representation of DGAs. By visualizing the modifications introduced by LRS in a 2D space, the neural network could potentially uncover distinct visual signatures associated with these modifications. This interdisciplinary approach, combining mathematical sequence analysis with visual pattern recognition through neural networks, can enhance the accuracy and efficiency of identifying LRS-modified DGAs, contributing to the continuous evolution of cybersecurity defense mechanisms.

References

[1] M. A. Ayub, S. Smith, A. Siraj, and P. Tinker, "Domain generating algorithm based malicious domains detection," in *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2021, pp. 77–82. DOI: 10.1109/CSCloud-EdgeCom52276.2021.00024.

[2] Z. Mu, "Predicting domain generating algorithms with n-grams," 2022.

[3] C. R. J. Ahmed H. H. Gharakheili and V. Sivaraman, "Automatic detection of dga-enabled malware using sdn and traffic behavioral modeling," 2022, pp. 2922–2939.

[4] M. Antonakakis, R. Perdisci, W. Lee, N. V. II, and D. Dagon, "Detecting malware domains at the upper DNS hierarchy," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*, Internet Society, 2012, pp. 1–10.

[5] M. Yahia, "Effective threat investigation for soc analysts," in *Effective Threat Investigation for SOC Analysts*, Packt Publishing, 2012, pp. 1–314.

[6] S. Yadav, P. Malik, and D. Patel, "A review on domain generation algorithms," *International Journal of Computer Applications*, vol. 85, no. 3, pp. 1–5, 2014.

[7] G. Wondracek, T. Holz, and E. Kirda, "A practical attack to de-anonymize social network users," in *Proceedings of the 2010 ACM SIGCOMM conference on Computer communication*, ACM, 2010, pp. 345–356.

[8] C. Umbrella. "Cisco umbrella." Accessed: November 2023. (), [Online]. Available: https://umbrella.cisco.com/.

[9] FireEye. "Fireeye dns analytics." Accessed: November 2023. (), [Online]. Available: https://www.fireeye.com/products/dns-analytics.html.

[10] S. Inc., *Splunk*. Splunk Inc., 2003.

[11] LogRhythm, "Logrhythm," *LogRhythm Inc.*, 2003.

[12] Y. Huang, Y. Cao, Z. Chen, J. Li, Z. Li, and W. Zou, "Dga-based botnet traffic detection using dns traffic," *Computers & Security*, vol. 56, pp. 45–59, 2016.

[13] F. Bisio, S. Saeli, P. Lombardo, D. Bernardi, A. Perotti, and D. Massa, "Real-time behavioral dga detection through machine learning," in *2017 International Carnahan Conference on Security Technology (ICCST)*, 2017, pp. 1–6. DOI: 10.1109/CCST.2017.8167790.

[14] Z. Wang, Z. Jia, and B. Zhang, "A detection scheme for dga domain names based on svm," in *Proceedings of the 2018 International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)*, Atlantis Press, 2018/03, pp. 257–263, ISBN: 978-94-6252-499-6. DOI: 10.2991/mmsa-18.2018.58. [Online]. Available: https://doi.org/10.2991/mmsa-18.2018.58.

[15] M. Kwon, J. Choi, and H.-W. Lee, "Detecting dga domains with recurrent neural networks and side information," *Computers & Security*, vol. 67, pp. 20–34, 2017.

[16] Z. Liu, X. Yun, Y. Zhang, and Y. Wang, "Ccga: Clustering and capturing group activities for dga-based botnets detection," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 136–143. DOI: 10.1109/TrustCom/BigDataSE.2019.00027.

[17] B. Poonen, "Linear recursive sequences," *Berkeley Math Circle*, n.d. Page 2. [Online]. Available: https://mathcircle.berkeley.edu/sites/default/files/BMC6/ps/linear.pdf.

[18] J. P. Fillmore and M. L. Marx, "Linear recursive sequences," *Siam Review*, vol. 10, pp. 342–353, 1968. [Online]. Available: https://api.semanticscholar.org/CorpusID:14722188.

[19] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 281–290.

[20] J. Ahmed, H. H. Gharakheili, C. Russell, and V. Sivaraman, "Automatic detection of dga-enabled malware using sdn and traffic behavioral modeling," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2922–2939, 2022. DOI: 10.1109/TNSE.2022.3173591.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[23] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2006.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[25] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," 2015, pp. 436–444. DOI: 10.1038/nature14539.

[28] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016. DOI: 10.1109/MGRS.2016.2540798.

[29] Q. V. L. Prajit Ramachandran Barret Zoph, "Searching for activation functions," 2018. DOI: arXiv:1710.05941.

[30] D. W. K. He, "Implicit self-regularization in deep neural networks: Evidence from random matrix theory," 2020, pp. 11 081–11 089. DOI: 10.1109/CVPR42600.2020.01105.

[31] U. Pujianto, A. P. Wibawa, and R. Ulfah, "Dictionary distribution based on number of characters for damerau-levenshtein distance spell checker optimization," in *2020 6th International Conference on Science in Information Technology (ICSITech)*, 2020, pp. 180–183. DOI: 10.1109/ICSITech49800.2020.9392059.

[32] A. A. Chandio, M. Asikuzzaman, M. R. Pickering, and M. Leghari, "Cursive text recognition in natural scene images using deep convolutional recurrent neural network," *IEEE Access*, vol. 10, pp. 10 062–10 078, 2022. DOI: 10.1109/ACCESS.2022.3144844.

[33] A. K. Sood and S. Zeadally, "A taxonomy of domain-generation algorithms," *IEEE Security Privacy*, vol. 14, no. 4, pp. 46–53, 2016. DOI: 10.1109/MSP.2016.76.

[34] C. C, P. K. Pareek, V. H. Costa de Albuquerque, A. Khanna, and D. Gupta, "Deep learning technique based intrusion detection in cyber-security networks," in *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, 2022, pp. 1–7. DOI: 10.1109/MysuruCon55714.2022.9972350.

[35] K. Alrawashdeh and C. Purdy, "Ransomware detection using limited precision deep learning structure in fpga," in *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, 2018, pp. 152–157. DOI: 10.1109/NAECON.2018.8556824.

[36] V. Ravi, M. Alazab, S. Srinivasan, A. Arunachalam, and K. P. Soman, "Adversarial defense: Dga-based botnets and dns homographs detection through integrated deep learning," 1, vol. 70, 2023, pp. 249–266. DOI: 10.1109/TEM.2021.3059664.

[37] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, "On the use of dgas in malware: An everlasting competition of detection and evasion," acm sigapp applied computing review," 2019, pp. 31–43.

[38] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, "Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework," *IEEE Access*, vol. 8, pp. 82 876–82 889, 2020. DOI: 10.1109/ACCESS.2020.2988877.

[39] X. Shen, X. Zhang, and Y. Chen, "Deep learning powered adversarial sample attack approach for security detection of dga domain name in cyber physical systems," *IEEE Wireless Communications*, vol. 29, no. 2, pp. 16–21, 2022. DOI: 10.1109/MWC.001.2100247.

[40] H. Vranken and H. Alizadeh, "Detection of dga-generated domain names with tf-idf," 2022, p. 414. DOI: doi.org/10.3390/electronics11030414.

[41] M. N, R. D, S Murali, and V. Sharma, "Performance analysis of dga-driven botnets using artificial neural networks," in *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2022, pp. 1–6. DOI: 10.1109/ICRITO56286.2022.9965044.

[42] S. L. N. R. J. Kate Highnam Domenic Puzio, "Real-time detection of dictionary dga network traffic using deep learning," 2021.

[43] B. Yu, J. Pan, J. Gray, *et al.*, "Weakly supervised deep learning for the detection of domain generation algorithms," 2019, pp. 51 542–51 556. DOI: 10.1109/ACCESS.2019.2911522.

[44] F. Panneton and P. L'Ecuyer, "On the xorshift random number generators," 2005, pp. 346–361.

[45] G. Marsaglia, "A current view of random number generators," 1985, pp. 3–10.

[46] G. Marsaglia, "Xorshift rngs," 2003. DOI: 10.18637/jss.v008.i14.

[47] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "Survey on malicious domains detection through dns data analysis," 2019, pp. 1–36. DOI: 10.1145/3191329.

[48] M. Louhichi, R. Nesmaoui, M. Mbarek, and M. Lazaar, "Shapley values for explaining the black box nature of machine learning model clustering," *Procedia Computer Science*, vol. 220, pp. 806–811, 2023, The 14th International Conference on Am-

bient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.03.107. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923006427.