Middle East Technical University
Informatics Institute

Using Deep Learning in Detecting Network Attacks

Advisor Name: Prof. Dr. Nazife Baykal (METU)

Student Name: Arsalan Vahi
(Cyber Security)

June 2024

TECHNICAL REPORT
METU/II-TR-2024-

Orta Doğu Teknik Üniversitesi
Enformatik Enstitüsü

Using Deep Learning in Detecting Network Attacks

Danışman Adı: Prof. Dr. Nazife Baykal
(ODTÜ)

Öğrenci Adı: Arsalan Vahi
(Siber Güvenlik)

Ocak 2024

TEKNİK RAPOR
ODTÜ/II-TR-2024-

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Internal Use)	**2. REPORT DATE** 26.07.2024
3. TITLE AND SUBTITLE **Using Deep Learning in Detecting Network Attacks**	
4. AUTHOR (S) Arsalan Vahi	**5. REPORT NUMBER (Internal Use)** **METU/II-TR-2024-**
6. SPONSORING/ MONITORING AGENCY NAME(S) AND SIGNATURE(S) Non-Thesis Master's Programme, Department of Cyber Security, Informatics Institute, METU Advisor: **Prof. Dr. Nazife Baykal**	Signature:
7. SUPPLEMENTARY NOTES	

8. ABSTRACT (MAXIMUM 200 WORDS)

Deep learning significantly enhances network attack detection by identifying and analyzing patterns and anomalies in network traffic. Traditional network security methods fail to recognize evolving threats; On the other hand, deep learning models can detect such threats. The important characteristics of these models are their ability to learn from data continuously, improve detection accuracy, and adapt to new attack vectors. However, the main disadvantage is the challenges of implementing network security. These challenges include the need for substantial computational resources and expertise. Despite these hurdles, deep learning provides a powerful and dynamic approach to network security, offering real-time threat detection and significantly bolstering cybersecurity defenses. In this document, we propose an idea of using image channels to find abnormal patterns in network traffic. We implemented this idea in a deep learning architecture and evaluated it on a test dataset to check the anomaly pattern detection for DNS spoofing attacks.

9. SUBJECT TERMS **Deep Learning, Network Attacks, DNS Spoofing Attacks,** **Anomaly Detection**	**10. NUMBER OF PAGES** **22**

Using Deep Learning in Detecting Network Attacks

ABSTRACT

Deep learning significantly enhances network attack detection by identifying and analyzing patterns and anomalies in network traffic. Traditional network security methods fail to recognize evolving threats; On the other hand, deep learning models can detect such threats. The important characteristics of these models are their ability to learn from data continuously, improve detection accuracy, and adapt to new attack vectors. However, the main disadvantage is the challenges of implementing network security. These challenges include the need for substantial computational resources and expertise. Despite these hurdles, deep learning provides a powerful and dynamic approach to network security, offering real-time threat detection and significantly bolstering cybersecurity defenses. In this document, we propose an idea of using image channels to find abnormal patterns in network traffic. We implemented this idea in a deep learning architecture and evaluated it on a test dataset to check the anomaly pattern detection for DNS spoofing attacks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IDS	Intrusion Detection System
IPS	Intrusion Protection System
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
DNN	Deep Neural Networks
DDoS	Distributed Denial of Services
LSTM	Long Short-Term Memory
GAN	Generative Adversarial Networks
ReLU	Rectified Linear Unit
FNN	Feedforward Neural Networks
DoS	Denial of Service
MITM	Man In The Middle
GRU	Gated Recurrent Unit

DNS	Domain Name System
IP	Internet Protocol
GNN	Graph Neural Network
DARPA	Defense Advanced Research Projects Agency
CICDS	Canadian Institute for Cybersecurity Intrusion Detection System
UDP	User Datagram Protocol

CHAPTER 1

INTRODUCTION

Network security comprises measures adopted to protect the resources and integrity of a computer network [22]. It protects underlying network infrastructure from unauthorized access, misuse, or theft. Network security involves creating a secure infrastructure for devices, applications, and users in a secure manner [15]. Network security is critical to modern information technology to protect integrity, confidentiality, and data availability as it is transmitted across computer networks. Over time, the dependency on decentralization networks has increased, amplifying the need for robust network security measures to safeguard sensitive information and ensure uninterrupted services. In addition, network security is one of the important elements of cybersecurity. In Fig 1, important elements of cybersecurity are depicted.

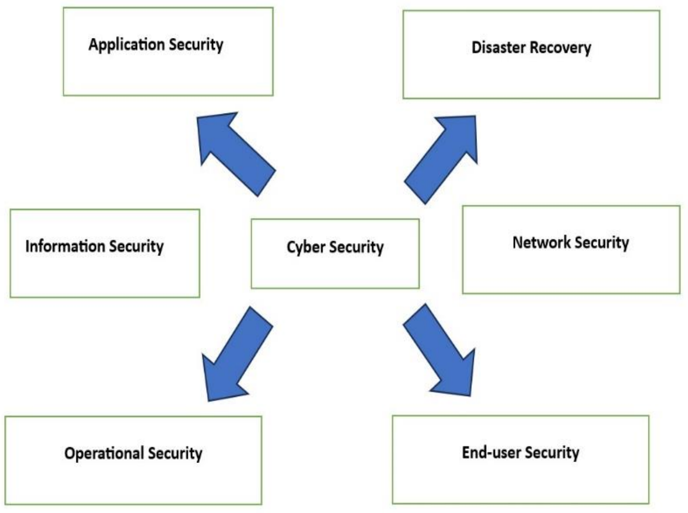

Figure 1: Important Elements of Cyber Security

As networks expanded and the Internet became ubiquitous, the range of severity of threats increased. As a result, more advanced solutions are necessitated. In figure 2, traditional network security measures are shown.

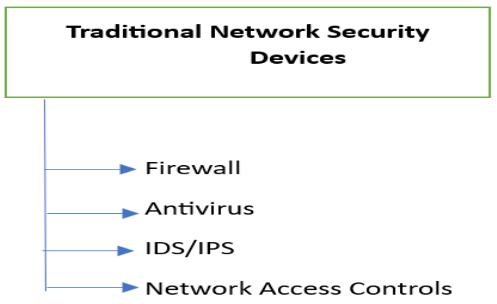

Figure 2: Traditional Network Security measures

Detecting network attacks plays a vital role in maintaining computer networks' security, integrity, and functionality. In Table 1, the key reasons are elaborated.

Table 1:Key reasons in detecting network attacks

#	**Key reason**	
1	Protection of sensitive data	Early detection of network attacks helps in safeguarding sensitive and confidential information such as personal data, financial information, proprietary business data.
2	Minimizing Downtime	Network attacks can lead to service disruptions and operational downtime. Detecting attacks early allows for prompt response and mitigation, thereby, minimizing the impact and ensuring continuety.
3	Maintaining Trust and Reputaion	Network attacks can result in substantial financial losses due to data breaches, and ransomware demands. Early detection helps to mitigate these financial aids.
4	Complinace with regulation	Organizations that fail to protect their network and data can suffer server reputational damage. Detecting and mitigating attacks helps maintain the organization’s reputation and trustworthiness.
5	Preventing spread of malware	Nework attacks can often involve the spread of malware within the network or other networks. Early detection helps eradicate malware before it can propagate.
6	Protecting critical infrastructure	Many network attacks target critical infrastrucure such as power grids, healthcare systems, and

		financial services. Early detection of these attacks is vital to protect public dafety, health and economy stability.
7	Enhancing incident responses	Early detection improves the effectiveness of incident response teams by providing timely alerts and information.
8	Identifying Vulnerabilities	Early detection can hellp identify vulnerabilities and weaknesses in an organization’s defense.

Detecting network attacks can be approached using traditional methods and neural network-based techniques. In Figure 3, traditional methods and neural network-based approaches are shown.

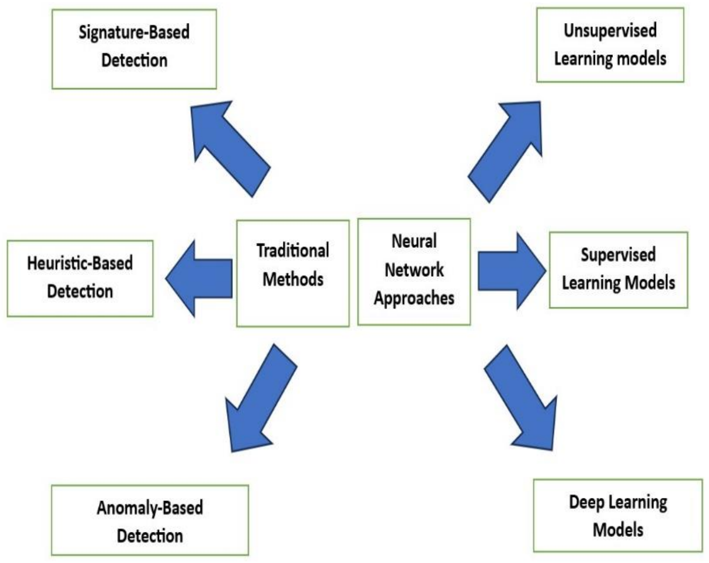
Signature-Based Detection
Unsupervised Learning models
Heuristic-Based Detection
Traditional Methods
Neural Network Approaches
Supervised Learning Models
Anomaly-Based Detection
Deep Learning Models

Figure 3:Traditional techniques and Neural network techniques

Traditional approaches for finding network attacks, like signature-based detection, rely on predefined patterns [47], whereas neural network approaches require large datasets for training [27]. Neural networks can potentially detect novel attacks through their ability to leave complex patterns, while traditional methods are often limited to known attack signatures [33]. Neural networks, especially deep learning methods can be more computationally intensive compared to traditional methods [10]. Traditional methods are generally easier to implement but may require frequent updates. Neural network approaches involve more complex implementations and tuning, but they can provide more advanced detection capabilities [1].

Signature-based detection relies on well-known patterns of malicious activity, often related to signatures, to identify attacks. Each signature corresponds to a specific type of attack or malware. This method is effective in detecting known threats with high accuracy; however, it is ineffective against new unknown attacks (zero-day threats) and variants of known attacks that do not limit existing signatures. In anomaly-based detection approach, a baseline of normal network behavior is established, and deviations from this baseline are identified as potential threats [17]. Anomaly-based detection can detect navel attacks that deviate from normal patterns. On the other hand, false-positive rates due to legitimate activities that may appear anomalous can be considered as disadvantages. Heuristic-based detection uses heuristic rules or algorithms to identify suspicious behavior or anomalies based on characteristics of known threats.

This method is more flexible than signature-based methods and can detect new variants of attacks. Conversely, this approach still results in false positives, the same as the anomaly detection method, and may not catch all novel threats.

Supervised learning models involve training a neural network on labeled datasets where network traffic features are associated with known attack types or data and normal behavior [43]. This approach can achieve high accuracy with sufficient labeled data and can generalize to new data that is similar to the training set [3]. Using this approach requires large amount of labeled data, which can be difficult to obtain. Methods that use unsupervised learning models use networks to identify patterns in network traffic without pre-labeled data [4]. Techniques like clustering are often used in this method. Although these methods can detect new and unknown attacks by identifying outliers and anomalies in the data, it can be less accurate than supervised models. It may require significant turning and validation to reduce false positives [11]. Approaching based on deep-learning models, utilize complex neural network architectures such as convolutional neural networks (CNNs), or Recurrent Neural Networks (RNNs) to analyze network traffic data [44].

These approaches can model complex patterns and dependencies in the data, potentially leading to higher detection accuracy. One important disadvantage of these approaches is their requirement of substantial computational resources and large database for training.

CHAPTER 2

BACKGROUNDS

Network attack detection plays a critical role in securing not only infrastructures but also organizations. In this chapter, we provide background on network attack detection techniques and neural networks. Additionally, we review related works.

2.1 Existing Network Attack detection methods

Network attack detection is a crucial component of cybersecurity to identify and mitigate malicious activities within a network. The field encompasses various techniques and methodologies, each designed to address different types of threats. In table 2, a summary of existing network attack detection methods is demonstrated.

Table 2:Network attack detection methods

#	Detection Method	Method	Pros	Cons
1	Signature-based detection	Predefined patterns or signature of known attacks	• Effective against known attacks • Low false positive rate for known attack types	• Ineffective against new, unknown attacks • Requires constant updates
2	Anomaly-based Detection	Establishing a baseline for typical network behavior and recognizing any deviations as possible dangers	• Capable of detecting unknown or novel attack • Can identif subtle attacks	• High false positive rate • Require significant computational resources
3	Behavioral-based detection	Simlar to anomaly-based detection, this method focues on behavior of users and systems	• Effective in detecting inside threats • Can uncover	• High false positive rate • Requires a deep

			complex attacks	understanding of normal behavior
4	Heuristic-based Detection	Using heuristic rules derived from the analysis of known attack strategies and technique (a mixture of signature and anomaly-based approach)	• Flexible and adaptable • Can detect variants of known attacks	• Potential to false-positive • Require reqular tuning and updates
5	Machine Learning-based detection	Identify patterns suggestive of attacks by analyzing large volumes of network data.	• Highly effective at identifying complex patterns and emerging threats • Can improve detection accuracy over time	• Requires large datasets • Complexity in algorithm selection and model training
6	Deep learning-based detection	Using neural networks with many layers This approach is used for more sophisticated pattern recognition	• Superior performance in detecting complex and previously unseen attacks • Capable of handling large-scale data	• Extremely resource intensive • Interpretability of results can be challenging

As it is obvious from table 2, each network attack detection has its strengths and weaknesses. However, in practice, network security often involves a layered approach, integrating several of these methods to provide robust protection against a wide range of threats.

2.2 Related Works

The application of neural networks in network attack detection has shown significant promise across various architectures, including DNNs, RNNs, CNNs, autoencoders, and hybrid models. Table 3 collectively demonstrates the application of neural networks in detecting network attacks used in different researches.

Table 3:Related Works

#	Research Name	Research Objective
1	Network Intrusion Detection System using Deep Learning Techniques [38]	The study explored the use of deep learning techniques, specifically deep neural networks (DNNs), for network intrusion detection. The research demonstrated that DNNs could automatically learn high-level features, from raw network traffic data.This results in improving detection accuracy compared to traditional machine learning methods.
2	RNNIDS: Enhancing network intrusion detection systems through deep learning [46]	In this study, RNNs (Recurrent Neural Networks) employed to capture temporal dependencies in network traffic data. This stduy showed that RNN, could effectively model the sequential nature of events, leading

		to better detection of complex attacks like (DDoS).
3	A survey of CNN-based network intrusion detection [34]	This research investigated that application of convolutional neural network (CNNs) for network intrusion detection. The study proved that CNNs chich traditionally used for image recognition could be adopted to analyze network traffic data, achieving high detection rates against various types of attacks.
4	Autoencoder-based network anomaly detection [7]	This study utilized Long Short-Term Memory (LSTM) based autoencoders to detecting network anomalies. The autoencoders were trained to reconstruct normal network traffic, and the deviations from this reconstruction, were played as potential anomalies.
5	A hybrid deep learning model for efficient intrusion detection in big data environment [16]	This research proposed a hybrid model combining CNNs, and LTSMs to leverage the traffic data. The hybrid modle autoperfomed individual deep learning models in detecting a variety of network attacks with higher accuracy and low false positive role.

Machine learning-based methods have some differences from neural networked-based detection approaches. Examples of machine learning methods, including decision trees, support vector machines (SVM), Random Forests, K-nearest neighbors, and Naive Bayes, are effective for structured data with a clear feature set, and they are relatively simple and easy to interpret [3]. Machine-learning methods may struggle with complex patterns and relationships in data and generally lower accuracy for sophisticated attacks. They are good for known attack types and finding linear patterns. On the other hand, neural network-based methods, including DNNs, RNNs, CNNs, Autoencoders, and GANs, can learn complex, non-linear relationships, which provide high accuracy for known and unknown attacks due to deep feature learning [9]. Their important weakness is that they require large datasets and significant computational resources, which makes them complex to design and train. Neural network-based methods are superior in detecting complex and previously unseen attack patterns.

2.3 An Overall of Neural Networks

A neural network is a computational model inspired by how biological neural networks in the human brain process information [18]. It contains of interconnected nodes (neurons) that work together to solve specific problems. Neural networks are used for tasks that involve pattern recognition, classification, and prediction [53]. This makes them well-suited for detecting anomalies and attacks in network traffic [51]. Basic components of neural networks are neurons (nodes), layer, weights and biases, and activation functions. Neurons (nodes) are the fundamental units of a neural network that recieve input, process it, and pass it to the next layer. Each neuron performs a weighted sum of its input and applies an application function. The input layer is the first layer that receives the raw data (e.g., network traffic features). Hidden layers are intermediate layers where the actual processing and

feature extraction occur. These layers can be one or many, and their depth often determines the complexity of the model. The output layer is the final layer that produces the output (e.g., classification of network traffic as benign or malicious)[40]. Some parameters are adjusted during training to minimize the error in prediction and weights and determine the strength of connections between neurons, while biases provide additional flexibility in decision boundaries [58]. None-linear functions are applied to the output of each neuron to introduce non-linearity into the model [5]. Common activation functions include Sigmoid, torch, and ReLU (Rectified Linear Unit) [43]. There are different types of neural networks, including Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders. FNNs are the simplest type of cycle. In other words, information moves in one direction- from input to output. CNNs specialize in processing structured grid data like images. They use convolutional layers to extract spatial features automatically[120]. In network security, CNNs can analyze raw traffic data. RNNs are designed to handle sequential data by maintaining a memory of previous inputs. They are useful for analyzing sequences of network events or logs. Auto encoders are a type of neural network used for unsupervised learning of efficient coding [28]. Because they can reconstruct inputs and spot deviations, they are especially helpful for anomaly detection.
Neural networks offer several advantages in pattern detection and anomaly detection. Neural networks, especially deep learning models, can automatically extract and learn hierarchical features from raw data without the need for manual feature engineering. This is particularly useful in network security, where relevant features may not be easily identifiable [15].

2.4 Advantages of Neural Networks

Neural networks can model complex and non-linear relationships in data, which are often present in sophisticated attack patterns and network traffic behaviors [30]. The capability enables them to detect subtle and intricate anomalies that traditional linear models might miss. In addition, neural networks generally offer higher accuracy and better detection rates for both known and unknown threats compared to traditional machine learning methods. Neural networks can efficiently handle large-scale datasets and high-dimensional data [19]. In the context of network security, this means they can process vast amounts of network traffic data and identify anomalies or attack patterns in real-time [56] . In Fig 4. the advantages of neural networks are briefly shown.

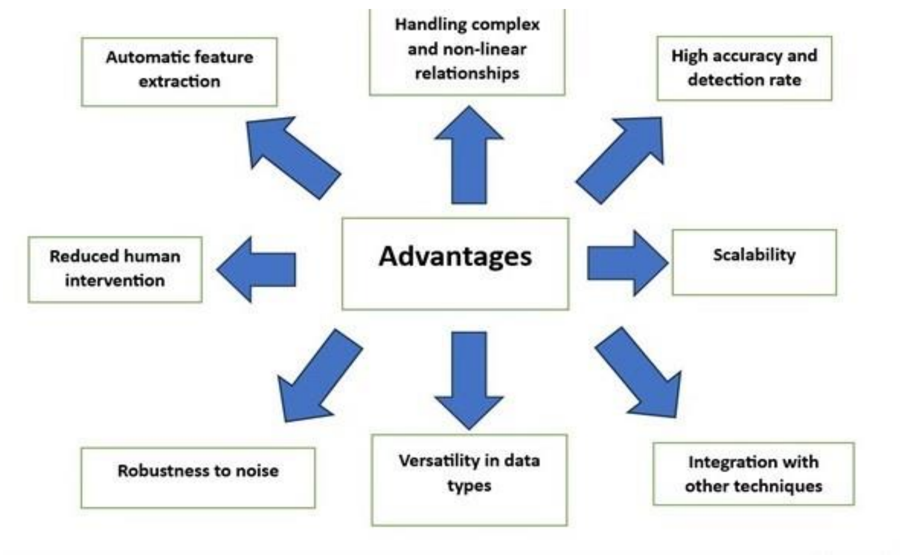
Handling complex and non-linear relationships
Automatic feature extraction
High accuracy and detection rate
Reduced human intervention
Advantages
Scalability
Robustness to noise
Versatility in data types
Integration with other techniques

Figure 4:Advantages of Neural Networks in attack detection

2.5 An Overview of Network Attacks

Network attacks are deliberate attempts to compromise the integrity, confidentiality, or availability of data and network resources. Common network attacks are:

1) Dos and DDoS attacks: these attacks aim to make a network resource unavailable to its intended users by overwhelming it with a flood of illegitimate requests. DoS attack originates from a single source, while DDoS attack originates from multiple compromised systems, such as botnets. These attacks can lead to significant downtime and damage to reputation [12][24][37].
2) Phishing and Spear-phishing: Social engineering attacks are designed to trick individuals into providing sensitive information such as usernames, passwords, and credit card numbers. In a phishing attack, generic emails sent to a large number of people. In spear phishing attack, targeted attacks directed at a specific individuals or organizations. These attacks aimed at compromising personal and financial information or potential authorized access to system [55][35][42].
3) Man-in-the-Middle (MITM) attacks: attackers intercept and potentially alter communications between two parties without their knowledge. Some well-known methods that are used are eavesdropping and session hijacking. These attacks have large impact such as data theft, session hijacking, and unauthorized access to sensitive information [54].

4) Malware attacks: Malware is a malicious software designed to infiltrate, damage, or disable computers, and networks. There are different types of malwares including viruses, worms, trojans, ransomware, and spyware [6].

Network attacks are divers and continually evolving, posing significant threats to organizations and individuals. Detecting network attacks is a complex and challenging task due to evolving cyber threats and the sophistication of attackers. In figure 5, some key challenges in detecting network attacks are shown.

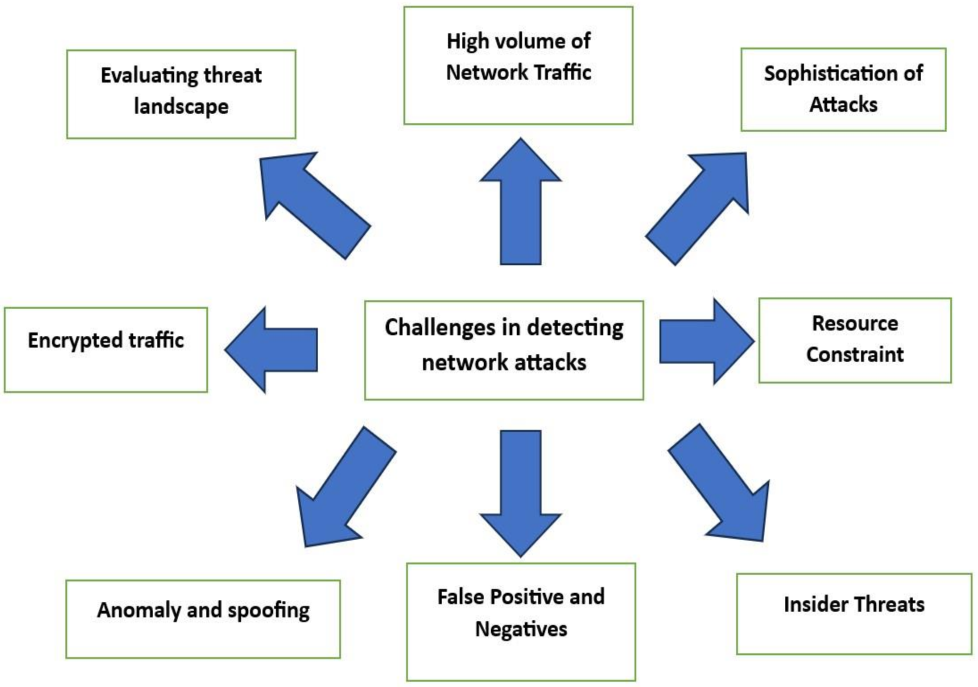
Challenges in detecting network attacks
High volume of Network Traffic
Sophistication of Attacks
Resource Constraint
Insider Threats
False Positive and Negatives
Anomaly and spoofing
Encrypted traffic
Evaluating threat landscape

Figure 5:Challenges in detecting network attacks

2.6 Neural Network Architectures

Designing a neural network architecture for network attack detection involves several considerations to ensure the model can effectively identify and classify malicious activities. Neural network architectures are used in detecting network attacks vary in complexity and application, leveraging different types of normal networks to address the unique challenges posted by network security. In table 4 several common and advanced neural network architectures that are used in detecting network attacks is given.

Table 4: Common Neural Network Architectures

#	Name	Structures	Applications	Advantages	Limitations
1	FNNs	Composed of an input layer, one or more hidden layers and an output layer with no cycles or loops [14]	Basic intrusion detection system (IDS) where simple patterns in the data needed to be identified	• Simplicity • Speed	Requires significant feature engineering (They cannot inherently capture complex features)
2	CNNs	Consist of convolutional layers, pooling layers, and fully connected layers [26]	Used for processing and analyzing data like traffic data represented in a grid or image-like format	• Can automatically learn hierarchical features from raw data • Useful for detecting patterns and anomalies in new network data log files	Require data to be formatted
3	RNNs	Designed for handle sequential data with loops that allow information to present	Ideal for examining logs or time series data from network traffic that contain sequences of network events	Can capture temporal patterns in sequential data	
4	LSTMs	A type of RNN	Ideal for detecting sophisticated attacks by analyzing long sequences of network data	• Can learn long-term dependencies and retain information over long period • Suitable for complex temporal patterns in network traffic	More complex and computationally intensive than simple RNNs and FNNs
5	GRUs	A variant of RN similar to RSTMs but with a simpler structure having feature gates	Used for similar purposes as LSTMs but with potentially faster training times	Faster to train and less computationally intensive	More complex than basic RNNs and FNNs
6	Autoencoders	Consists of an encoder that compresses the input data into a slower-dimensional representation and a decoder that reconstructs the original data from these representations	Anomaly detection	• Can learn from unlabeled data, which is abundant in network traffic • Effective at detecting anomalies where reconstruction error is high	Prone to overfitting
7	GANs	consists of two neural networks that are trained concurrently: a discriminator and a	Generating Synthetic network traffic to train other models or for detecting	• Can generate realistic network traffic for training and testing purpose	• Training GANs can be challenging due to instability

		generator. The discriminator attempts to discern between real and fake samples, while the generator attempts to produce realistic data samples.	anomalies by identifying data that discriminator finds difficult to classify.	• Effective in detecting anomalies	• Computationally intensive

As it is obvious from table 4, each neural network architecture offers various strengths and weaknesses for detecting network attacks. The choice of architecture depends on the specific requirement of the detection systems such as type of data, complexity of attack patterns and the need for real-time processing.

2.7 Neural Network Architecture Design Considerations

Designing a neural network for network attack detection involves several considerations and choices regarding the architecture, the architecture, activation functions, train algorithms, and other parameters. Network traffic data requires normalization or scaling to ensure that features have similar ranges; categorical features such as protocol types or port numbers need to be encoded, typically one-hot encoding or label encoding. Selecting relevant features from the raw data is crucial. This can include packet size, time intervals, IP addresses, port numbers, and more. Regarding the architecture design, the shape of the input layer is defined based on the preprocessed features. The number and type of hidden layers depend on the complexity of the task and the amount of data. Layer type (CNNs, RNNs, LSTMs, GRUs) is defined based on the characteristics and parameters such as number of units, return sequences, number of filters, and so on. The other adjustments (such as activation functions and evaluation metrics) can be made based on specific requirements and dataset characteristics.

2.8 An Overview of Image Processing

Image processing involves the manipulation and analysis of images to extract useful information or enhance their quality. It plays a fundamental role in computer vision, medical imaging, and graphic design, with applications ranging from facial recognition to satellite imagery. An image is composed of multiple channels, each representing a color or intensity level. For example, a typical RGB (Red, Green, Blue) image has three channels: one for each primary color. By adjusting these channels, images can be modified in terms of color balance, brightness, contrast, and more. Grayscale images, on the other hand, have just one channel, representing shades of gray. Our main idea is to take advantage of the concept of image channels. We can view network traffic as different layers and apply image processing techniques to identify anomalies. This leads to easily manipulating network traffics and extract anomalies. In the next chapter, we apply our method to detect DNS spoofing attack anomalies in targeted network traffic and present the results.

CHAPTER 3

DNS Spoofing Attack

3.1 An Overview of DNS Spoofing Attack

DNS cache poisoning (also known as DNS spoofing) is a type of cyber-attack where corrupt DNS data is inspired into the DNS resolver’s cache, causing the name server to return an incorrect IP address and deviating traffic to malicious site [36]. The attack threatens the integrity of DNS, which is a critical component of the Internet infrastructure responsible for translating human readable domain names into IP addresses that computers use on the network. In figure 6 key characteristics of DNS cache poisoning attack is shown.

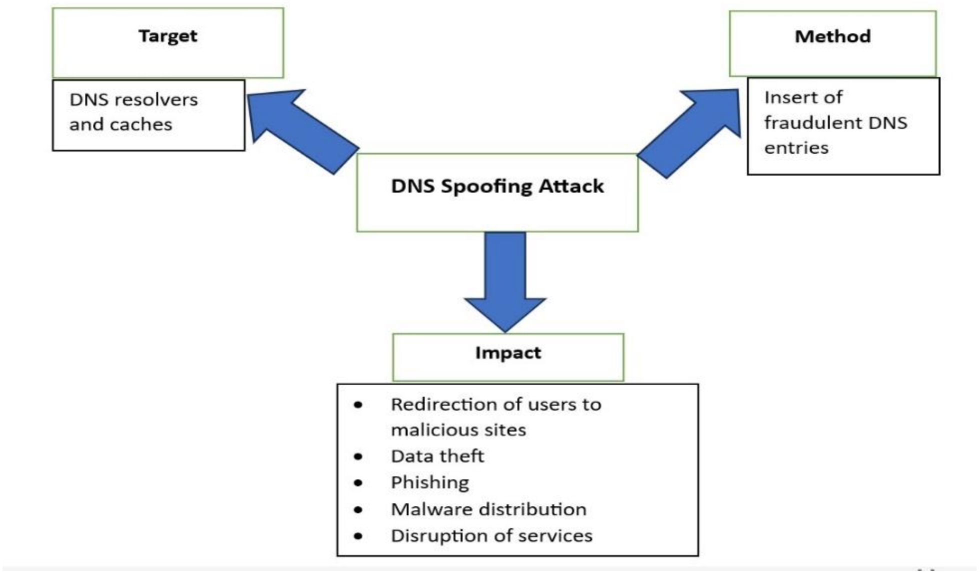

Figure 6:Key Characteristics of DNS Cache Poisoning Attack

3.2 Raw Data Collection

Raw data collection and generation are fundamental steps in the development of neural network models for network attack detection. They involve gathering and creating data that represents network activities, both benign and malicious. This data serves as the fundamental of training, validating and testing machine learning algorithms. The data can be provided either by collecting or generating where each has different methods. In figure 7 these methods are shown.

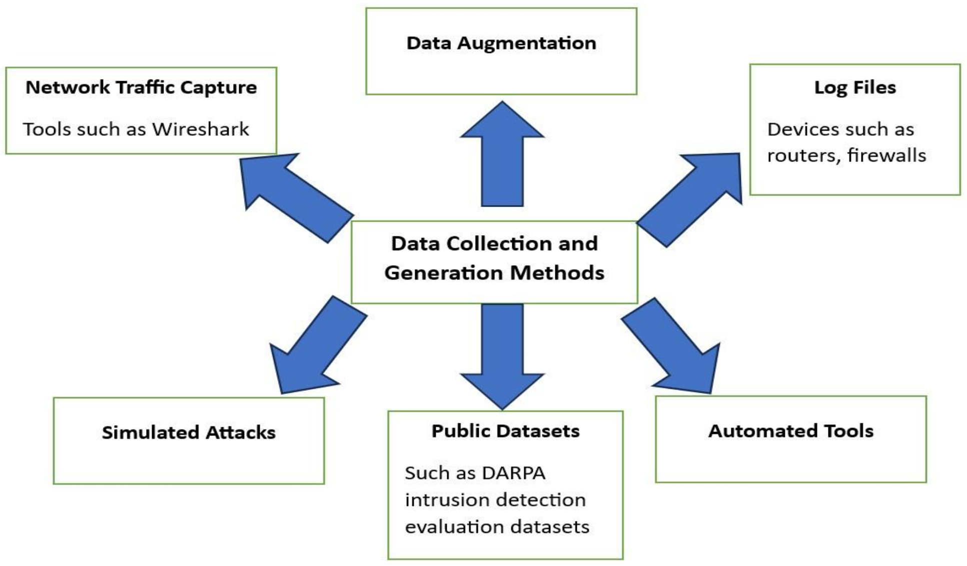
Data Augmentation
Network Traffic Capture
Tools such as Wireshark
Log Files
Devices such as routers, firewalls
Data Collection and Generation Methods
Simulated Attacks
Public Datasets
Such as DARPA intrusion detection evaluation datasets
Automated Tools

Figure 7: Data Collection and Generation Methods

Network traffic can be captured in real-time using tools such as Wireshark, tcpdump, and bro (now Zech). Each packet traversing the network is monitored and recorded by these tools, which provide detailed information about each one. Network devices, like routers, firewalls, and servers, generate log files that contain records of network activities [25]. These logs can provide insight into connection events, traffic flows, and detected anomalies. By analyzing these logs, researchers can identify patterns and behaviors associated with both normal usage and malicious activities. Publicly available datasets like KDD Cup 1999, DARPA intrusion, evaluation dataset, and CICIDS provide labeled network traffic data for various attack scenarios [45].

Simulated attacks are one of the methods used to generate data. Simulating network attacks in controlled environments allows for the generation of synthetic data [52]. These simulations can include various attacks, such as DDoS, phishing, and malware infections. In addition, Simulation provides the advantage of generating labeled data when the nature of each data point (normal traffic vs. specific traffic) is known [48]. Automated tools can generate large volumes of network traffic that include both normal and malicious activities [59]. These tools help to create comprehensive datasets that cover a wide range of scenarios, ensuring that the machine learning models are exposed to diverse types of network activities [49]. Data augmentation techniques involve creating new training examples by modifying existing data. These techniques can include duplicating samples or making slight alterations to the data. High-quality data with minimal noise and error is essential for training reliable neural network models. Data preprocessing steps, such as cleaning and normalization, help improve data quality [8]. Additionally, the dataset should be representative of the real-world environment where the model will be deployed, including a balanced mix of both normal and malicious activities [39].

For our case study, we used protocol fuzzing for data generation. Web protocol fuzzing can generate large variety of malicious network packets at a fast speed. In this method, all are labeled as malicious because they are generated from a desired network attack. In our case, all data generated and stored in order to create the dataset.

3.3 Labeling DNS Sessions

Labeling DNS sessions involves categorizing and tagging DNS traffic to distinguish between different types of network activities. This process is crucial for various network security tasks including monitoring, anomaly detection, and forensic analysis [23]. DNS session labeling enables more effective tracking of network activity by categorizing [57]. In addition, facilitates the identification of anomalies and potential security threats by comparing current traffic against labeled data [31]. Labeling DNS sessions has some challenges. DNS traffic can be volumes, requiring efficient methods to label and analyze large dataset attack patterns that evolve over time, necessitating continuous updates to labeling criteria and methods. Ensuring high accuracy in labeling to avoid false positives and negatives is another challenge in labeling DNS sessions.

In our case study, the malicious and benign raw data have already been separated during protocol fuzzing. As a result, we are not required to do another extra process.

3.4 Feature Extraction and Data Sample Representation

Feature extraction and data sample representation are crucial steps in the process of detecting network attacks, such as DNS cache poisoning. These steps involve transforming raw network data into a structured format that can be used as input for machine learning models. Feature extractions involve selecting and transforming raw data into meaningful attributes that can be used by machine learning algorithms. For network attack detection, features can be derived from various sources, such as network traffic, logs, and system events [32]. In Figure 8, key steps in feature extraction are shown.

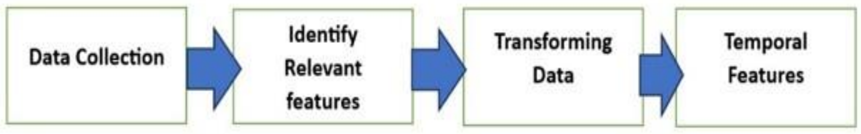

Figure 8: Key Steps in Feature Extraction

Data sample representation involves organizing the extracted features into a format suitable for input into machine learning models. Data samples can be represented in a tabular format, sequential format, or graph-based format. Each representation has its own characteristics that make it suitable for each model. Sequential formats are suitable for models like RNNs and LSTMs, where data is represented as sequences of events. Graph-based representation is useful for GNNs and detecting complex attack patterns.

In our case study, 32 bytes are chosen from every DNS packet instead of a whole packet. These 32 bytes include IP layer, UDP layer, and part of a DNS layer. After the packet processing, every packet is represented as fixed-length sequences of 32 integer numbers ranging from 0 to 255 (a 32-integer vector). For the packets with variable length, we applied sliding window to convert it to a fixed-length sequences.

In fig 9, the python code for applying window sliding is shown.

```
def window_sliding(x_list, window_size, window_step):
        x = []
        For i in range(len(x_list)):
                Line = x_list[i]
                n = len(line)
                for j in range (0 , n – window_size + 1, window_step)
                        if j+window_size <= len(line)
                                x.append(line[j:j+window_size])
                        else:
                                x.append(line[n-window_size:n])
        return np.array(x)
```

Figure 9: Python code for applying window sliding

3.5 Dataset Construction

This section involves several steps to ensure collected data is relevant, accurate, and useful for the intended analysis. In figure 9 general is covered.

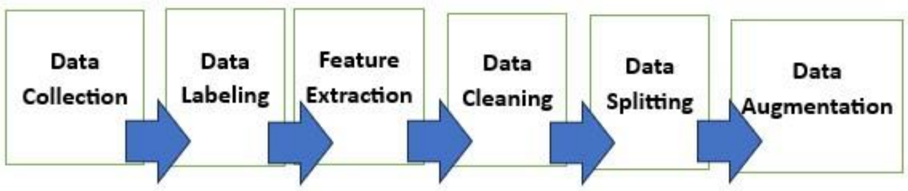

Figure 10: Data construction Process

Regarding the DNS cache poisoning attack, the dataset includes real-network traffic captured over a specific period. This ensures the data represents actual usage patterns and potential attack sessions. In addition to real traffic, simulated attack scenarios are generated and included in the dataset. This helps to create a comprehensive dataset that includes various attack patterns and methods. Key features that are selected include attributes like source and destination IP addresses, DNS query types, and response time. Each entry in the dataset is labeled as either normal traffic or attack traffic. The data undergoes a preprocessing step such as normalization and noise filtering. By doing this, the data fed into the machine-learning models is made sure to be clean and well-structured. The data is split into training, validation, and test sets to allow for robust model training and evaluation. This split helps in assessing the model's performance on unseen data.

3.6 Model Architecture

The model architecture focuses on the details of the structure and components of the neural network used for detecting network attacks, particularly focusing on the choice of layers, activities, functions, and other design patterns. The model architecture is crucial because the effectiveness of a neural network heavily relies on how well its architecture is suited to the specific task. The purpose of the input layer is to receive the raw input data. The purpose of hidden layers is to process and transform the input data through various computations to extract meaningful patterns. This combination of layers and patterns helps the model effectively learn to distinguish between normal and malicious network traffic, addressing the specific challenges posed by network attack detection. In fig 11 our purposed construction is shown.

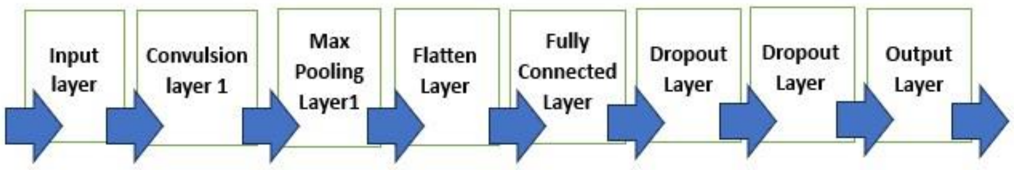

Figure 11: Proposed Architecture

3.7 Parameter Tuning

The process of parameter tuning involves altering the hyperparameters of neural network models to enhance their ability to detect network attacks [13]. Common hyperparameters include learning rate, batch size, number of epochs, number of layers/neurons, dropout rate, and activation functions. The learning rate controls the step size during the gradient descent optimization. A small learning rate convergence with a large learning rate can cause the model to converge too quickly. Batch size determines the number of training samples used to calculate the gradient descent in each iteration. Smaller batch sizes can lead to noisy updates, while layer batch sizes provide more stable updates but require more memory. The number of epochs is a measurement of how many times a complete pass through the entire training dataset has been completed. More epochs can improve learning but may also increase the risk of overfitting. An important issue is adjusting the depth (number of layers) and width (number of neurons per layer) of the neural network to find a suitable architecture that captures the complexity of data without overfitting [50]. Drop rate is a regularization technique to prevent overfitting by randomly dropping units during training [21]. The drop rate specifies the proportion of units to drop. Activation functions such as ReLU, sigmoid, or Tanh are applied to each neuron's output [40]. The choice of activation functions affects the learning dynamics and model performance [21]. Different tunning methods can be applied. Exhaustive search over a predefined set of hyperparameters is called Grid search [54]. It evaluates every combination, which can be computationally expensive but thorough. In the random search tuning method, randomly sampled hyperparameters from specific distributions are more efficient than grid search and often find good hyperparameters with fewer evaluations. Bayesian optimization uses a probabilistic model to predict the performance of different hyperparameter settings and choose the next set to evaluate [2]. Methods like hypergradient descent adjust hyperparameters based on their gradients, similar to how model parameters are optimized. Either model tuning or automated tools can be used for hyperparameter tuning [29]. In our case study set corresponds to the sliding window, and the number of units in layers are considered as hyper parameters.

3.8 Model Deployment

The deployment of a model for DNS attack detection involves integrating the trained neural network into a live environment where it can monitor and analyze real-time DNS traffic. In fig 10, Python code for model building is shown.

```
def create_network(input_size, layer_dims, filter_sizes, dropout_prob):
            input_layer = tf.keras.layers.Input(shape=(input_size, 32, 8), name='InputLayer')
            x = tf.keras.layers.Conv2D(filters=layer_dims[0], kernel_size=filter_sizes[0],
            padding='same', activation='relu', name='ConvBlock1')(input_layer)
            x = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2),
                             padding='same', name='PoolBlock1')(x)

             x = tf.keras.layers.Conv2D(filters=layer_dims[1], kernel_size=filter_sizes[1],
                             padding='same', activation='relu', name='ConvBlock2')(x)
            x = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2),
                                  padding='same', name='PoolBlock2')(x)

            x = tf.keras.layers.Flatten(name='FlattenLayer')(x)
            x = tf.keras.layers.Dense(layer_dims[2], activation='relu', name='DenseLayer')(x)
            x = tf.keras.layers.Dropout(rate=dropout_prob, name='DropoutLayer')(x)
            output_layer = tf.keras.layers.Dense(2, activation='softmax',
            name='OutputLayer')(x)
            model = tf.keras.models.Model(inputs=input_layer, outputs=output_layer,
            name='NetworkModel')
            return model
```

Figure 12: Python code for Model building

3.9 Evaluation Results

This section presents the findings from testing and validating the proposed methods or systems against established metrics. Fig 11 Covers aspects in the context of DNS cache poisoning attack detection.

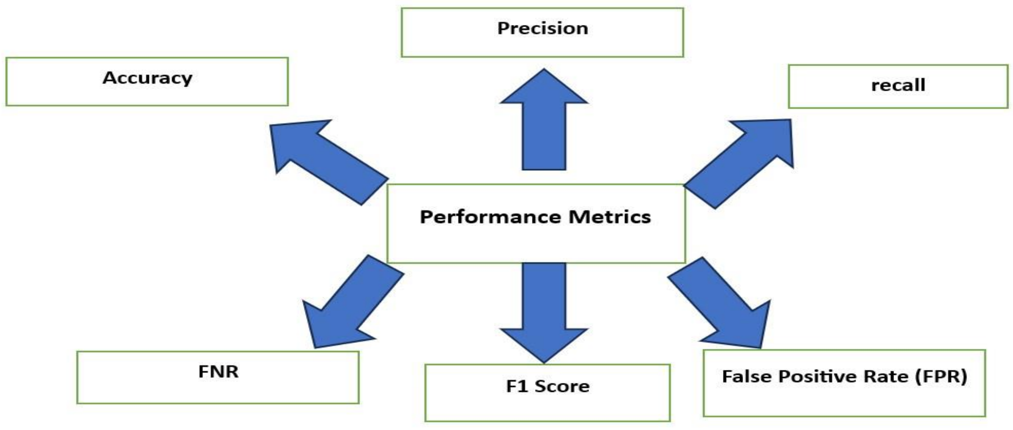

Figure 9:Performance Metrics

Accuracy is one of the performance metrics. This metric shows the ratio of the number of correct predictions to the total number of predictions. Recall metric, answers the question “of all instances that were actually positive, how many were predicted correctly?”. On the hand, precision metric defines “Of all insurances that were predicted as positive, how many were actually positive?”. F1 score metric combines precision and recall into a single metric [41]. Based on our proposed architecture, and evaluation on dataset, results are shown in table 5.

Table 5:Extracted Results

Data Set	**Accuracy**	**Precision**	**Detection rate**	**F1 Score**
Training	0.9888	0.9755	0.9756	0.9765
Test	0.9766	0.9661	0.9891	0.9971

These results illustrate how well our model performs according to the aforementioned metrics, and provide insights into areas where the model excels or may need improvement. The comprehensive evaluation helps in understanding the model’s strengths and weaknesses, guiding further refinements and optimizations.

CHAPTER 4

CONCLUSION

In this document, we introduced the idea of taking advantage of the concept of image channels and employ an image processing technique in order to find network anomalies. we have evaluated the performance of our proposed architecture using various metrics to ensure a comprehensive assessment. Accuracy, recall, precision, and F1 score have been utilized to gauge the effectiveness of our model in making correct predictions. The presented results demonstrate the effectiveness of our model based on these metrics. Our analysis reveals that the proposed architecture performs well in terms of accuracy and precision, while also maintaining a strong balance between recall and precision as indicated by the F1 score. However, the detailed evaluation also uncovers areas for potential improvement. Overall, the comprehensive assessment has validated the robustness of our proposed architecture while providing valuable insights for future refinements. This evaluation sets the stage for continued development and optimization of the model, with the goal of enhancing its performance and applicability in real-world scenarios.

REFERENCES

[1] Blin, K., Shaw, S., Kloosterman, A. M., Charlop-Powers, Z., Wezel, G. P., Medema, M., & Weber, T. (2021). antiSMASH 6.0: improving cluster detection and comparison capabilities. Nucleic Acids Research, 49, W29-W35.
[2] Brochu, Eric, Vlad M. Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning." arXiv preprint arXiv:1012.2599 (2010).
[3] Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in neural information processing systems 33 (2020): 12449-12460.
[4] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... Stoyanov, V. (2019). Unsupervised Cross-lingual Representation Learning at Scale. Annual Meeting of the Association for Computational Linguistics.
[5] Cheng, X., Chen, Y., & Sra, S. (2023). Transformers Implement Functional Gradient Descent to Learn Non-Linear Functions In Context. ArXiv, abs/2312.06528.
[6] Chai, Yuhan, Lei Du, Jing Qiu, Lihua Yin, and Zhihong Tian. "Dynamic prototype network based on sample adaptation for few-shot malware detection." IEEE Transactions on Knowledge and Data Engineering 35, no. 5 (2022): 4754-4766.
[7] Chen, Zhaomin, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. "Autoencoder-based network anomaly detection." In *2018 Wireless telecommunications symposium (WTS)*, pp. 1-5. IEEE, 2018.
[8] Chen, Shifu. "Ultrafast one-pass FASTQ data preprocessing, quality control, and deduplication using fastp." Imeta 2, no. 2 (2023): e107.
[9] Duan, Ranjie, Xiaofeng Mao, A. Kai Qin, Yuefeng Chen, Shaokai Ye, Yuan He, and Yun Yang. "Adversarial laser beam: Effective physical-world attack to dnns in a blink." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16062-16071. 2021.
[10] Deng, Li, and Dong Yu. "Deep learning: methods and applications." Foundations and trends® in signal processing 7, no. 3–4 (2014): 197-387.
[11] Dalalah, Doraid, and Osama MA Dalalah. "The false positives and false negatives of generative AI detection tools in education and academic research: The case of ChatGPT." The International Journal of Management Education 21, no. 2 (2023): 100822.
[12] Du, Hongyang, Jiacheng Wang, Dusit Niyato, Jiawen Kang, Zehui Xiong, Xuemin Shen, and Dong In Kim. "Exploring attention-aware network resource allocation for customized metaverse services." IEEE Network 37, no. 6 (2022): 166-175.
[13] Eimer, Theresa, Marius Lindauer, and Roberta Raileanu. "Hyperparameters in reinforcement learning and how to tune them." In International Conference on Machine Learning, pp. 9104-9149. PMLR, 2023.
[14] Fei, Juntao, and Cheng Lu. "Adaptive sliding mode control of dynamic systems using double loop recurrent neural network structure." *IEEE transactions on neural networks and learning systems* 29.4 (2017): 1275-1286.
[15] Guo, Hongzhi, Jingyi Li, Jiajia Liu, Na Tian, and Nei Kato. "A survey on space-air-ground-sea integrated network security in 6G." IEEE Communications Surveys & Tutorials 24, no. 1 (2021): 53-87.
[16] Hassan, Mohammad Mehedi, Abdu Gumaei, Ahmed Alsanad, Majed Alrubaian, and Giancarlo Fortino. "A hybrid deep learning model for efficient intrusion detection in big data environment." Information Sciences 513 (2020): 386-396.
[17] Heramil, James Aaron, Kyle Dumbrique, Mariah Rocita Mirarza, Lionel Kerwin Ejorango,

Roselle Wednesday Gardon, and Lorena Rabago. "Threatlocke: An Anomaly Based Detection Model." In 2023 8th International Conference on Information Technology and Digital Applications (ICITDA), pp. 1-6. IEEE, 2023.
[18] Han, Song, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems 28 (2015).
[19] Hebart, Martin N., Oliver Contier, Lina Teichmann, Adam H. Rockter, Charles Y. Zheng, Alexis Kidder, Anna Corriveau, Maryam Vaziri-Pashkam, and Chris I. Baker. "THINGS-data, a multimodal collection of large-scale datasets for investigating object representations in human brain and behavior." Elife 12 (2023): e82580.
[20] Howard, Andrew G. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv preprint arXiv:1704.04861 (2017).
[21] Hussain, Aamal Abbas, Francesco Belardinelli, and Georgios Piliouras. "Asymptotic convergence and performance of multi-agent q-learning dynamics." arXiv preprint arXiv:2301.09619 (2023).
[22] Ilić, David, and Gilles E. Gignac. "Evidence of interrelated cognitive-like capabilities in large language models: Indications of artificial general intelligence or achievement?." *Intelligence* 106 (2024): 101858.
[23] Jacobs, Arthur S., Roman Beltiukov, Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta, and Lisandro Z. Granville. "AI/ML for network security: The emperor has no clothes." In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 1537-1551. 2022.
[24] Kolias, Constantinos, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. "DDoS in the IoT: Mirai and other botnets." Computer 50, no. 7 (2017): 80-84.
[25] Karlsen, Egil, Rafael Copstein, Xiao Luo, Jeff Schwartzentruber, Bradley Niblett, Andrew Johnston, Malcolm I. Heywood, and Nur Zincir-Heywood. "Exploring semantic vs. syntactic features for unsupervised learning on application log files." In 2023 7th Cyber security in networking conference (CSNet), pp. 219-225. IEEE, 2023.
[26] Ketkar, Nikhil, et al. "Convolutional neural networks." Deep learning with Python: learn best practices of deep learning models with PyTorch (2021): 197-242.
[27] Liu, Ling. "Computational morphology with neural network approaches." arXiv preprint arXiv:2105.09404 (2021).
[28] Li, Xiang, Tiandi Ye, Caihua Shan, Dongsheng Li, and Ming Gao. "Seegera: Self-supervised semi-implicit graph variational auto-encoders with masking." In Proceedings of the ACM web conference 2023, pp. 143-153. 2023.
[29] Lian, D., Zhou, D., Feng, J., & Wang, X. (2022). Scaling & Shifting Your Features: A New Baseline for Efficient Model Tuning. Neural Information Processing Systems.
[30] Li, Liam, and Ameet Talwalkar. "Random search and reproducibility for neural architecture search." In Uncertainty in artificial intelligence, pp. 367-377. PMLR, 2020.
[31] Liu, Jiaqi, Guoyang Xie, Jinbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. "Deep industrial image anomaly detection: A survey." Machine Intelligence Research 21, no. 1 (2024): 104-135.
[32] Lu, Siyu, Yueming Ding, Mingzhe Liu, Zhengtong Yin, Lirong Yin, and Wenfeng Zheng. "Multiscale feature extraction and fusion of image and text in VQA." International Journal of Computational Intelligence Systems 16, no. 1 (2023): 54.
[33] Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).
[34] Mohammadpour, Leila, Teck Chaw Ling, Chee Sun Liew, and Alihossein Aryanfar. "A survey of CNN-based network intrusion detection." *Applied Sciences* 12, no. 16 (2022): 8162.
[35] Nielsen, Kurt, and Helle Betina Kristensen. "End-to-end mapping of a spear-phishing attack on HEI in EU." Proceedings of the European Univer 78 (2021): 89-97.
[36] Man, Keyu, Xin'an Zhou, and Zhiyun Qian. "Dns cache poisoning attack: Resurrections with

side channels." In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 3400-3414. 2021.
[37] Panda, Chakradhara, and Tilak Raj Singh. "ML-based vehicle downtime reduction: A case of air compressor failure detection." Engineering Applications of Artificial Intelligence 122 (2023): 106031.
[38] Rathee, Ashish, Parveen Malik, and Manoj Kumar Parida. "Network Intrusion Detection System using Deep Learning Techniques." *2023 International Conference on Communication, Circuits, and Systems (IC3S)*. IEEE, 2023.
[39] Raza, Ali, Kashif Munir, Mubarak S. Almutairi, and Rukhshanda Sehar. "Novel class probability features for optimizing network attack detection with machine learning." IEEE Access (2023).
[40] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).
[41] Roth, Karsten, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. "Towards total recall in industrial anomaly detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 14318-14328. 2022.
[42] Sabadash, Ivanna, Nestor Dumanskyi, and Igor Korobiichuk. "Methods and Means of Identifying Fraudulent Websites." In COAPSN, pp. 177-186. 2020.
[43] Sitzmann, Vincent, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions." Advances in neural information processing systems 33 (2020): 7462-7473.
[44] Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Interspeech (pp. 338-342).
[45] Shah, B., & Trivedi, B. (2015). Reducing Features of KDD CUP 1999 Dataset for Anomaly Detection Using Back Propagation Neural Network. In 2015 Fifth International Conference on Advanced Computing & Communication Technologies (pp. 247-251).
[46] Sohi, Soroush M., Jean-Pierre Seifert, and Fatemeh Ganji. "RNNIDS: Enhancing network intrusion detection systems through deep learning." Computers & Security 102 (2021): 102151.
[47] Szynkiewicz, Paweł. "Signature-based detection of botnet DDoS attacks." In Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools, pp. 120-135. Cham: Springer International Publishing, 2022.
[48] Shum, KaShun, Shizhe Diao, and Tong Zhang. "Automatic prompt augmentation and selection with chain-of-thought from labeled data." arXiv preprint arXiv:2302.12822 (2023).
[49] Sivanathan, Arunan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT devices in smart environments using network traffic characteristics." IEEE Transactions on Mobile Computing 18, no. 8 (2018): 1745-1759.
[50] Shi, Wenzhe, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1874-1883. 2016.
[51] Sivanathan, Arunan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT devices in smart environments using network traffic characteristics." IEEE Transactions on Mobile Computing 18, no. 8 (2018): 1745-1759.
[52] Tang, Ruixiang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. "Does synthetic data generation of llms help clinical text mining?." arXiv preprint arXiv:2303.04360 (2023).
[53] Takeuchi, Osamu, and Shizuo Akira. "Pattern recognition receptors and inflammation." Cell 140, no. 6 (2010): 805-820.
[54] Yao, Yu, Junhui Zhao, Zeqing Li, Xu Cheng, and Lenan Wu. "Jamming and eavesdropping defense scheme based on deep reinforcement learning in autonomous vehicle networks." IEEE Transactions on Information Forensics and Security 18 (2023): 1211-1224.
[55] Wang, Yanbin, Wenrui Ma, Haitao Xu, Yiwei Liu, and Peng Yin. "A lightweight multi-view

learning approach for phishing attack detection using transformer with mixture of experts." Applied Sciences 13, no. 13 (2023): 7429.
[56] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7464-7475. 2023.
[57] Yang, Zhao, Ye Hongzhi, Li Lingzi, Huang Cheng, and Zhang Tao. "Detecting DNS tunnels using session behavior and random forest method." In 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), pp. 45-52. IEEE, 2020.
[58] Zhou, Hengyi, Longjun Liu, Haonan Zhang, Hongyi He, and Nanning Zheng. "CMB: A Novel Structural Re-parameterization Block without Extra Training Parameters." In 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1-9. IEEE, 2022.
[59] Zivanov, Jasenko, Takanori Nakane, Björn O. Forsberg, Dari Kimanius, Wim JH Hagen, Erik Lindahl, and Sjors HW Scheres. "New tools for automated high-resolution cryo-EM structure determination in RELION-3." elife 7 (2018): e42166.