

RESILIENT DETECTION OF CYBER ATTACKS IN INDUSTRIAL DEVICES

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Y. Ashiq Meeran (Reg-39110090)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Y. Ashiq Meeran (Reg 39110090)** who carried out the Project Phase-2 entitled "**RESILIENT DETECTION OF CYBER ATTACKS IN INDUSTRIAL DEVICES**" under my supervision from January 2023 to April 2023.

Internal Guide

Dr. S. Prayla Shyry , M.E., Ph.D.,

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.



Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner

External Examiner

DECLARATION

I, **Y. Ashiq Meeran (Reg- 39110090)** , hereby declare that the Project Phase-2 Report entitled "**RESILIENT DETECTION OF CYBER ATTACKS IN INDUSTRIAL DEVICES**" done by me under the guidance of **Dr. S. Prayla Shyry , M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE:20.04.23



PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. Prayla Shyry , M.E., Ph.D.**,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways

for the completion of the project.

iv

ABSTRACT

With the advent of smartphones, laptops, and home computers, smart systems are becoming more and more flexible. As the use of the internet increases, there will be more cyber threats occurring on most third-party connectivity websites. The powerful techniques that are used to detect the threats present in the IoT applications are discussed here. In the proposed system. Based on the KAGGLE NIDS dataset, the number of possible attacks is calculated in the proposed architecture. A similar occurrence of intrusion creating a task is detected by the system, triggering the model to prevent the intrusion by notifying the user immediately. The existing attack detection systems have a number of limitations which includes the need of human intervention to detect the attacks encountered, slower detection rate and inaccuracy in detection. An advanced intrusion detection model is developed in the proposed research work to overcome these limitations. The collection of intrusion models, called as bags of attacks. An internet of things network with a robust architecture can withstand a number of possible attacks. A robust deep learning algorithm is proposed for detecting feasible intrusions in IoT networks; based on the vulnerability of IoT attacks, the proposed algorithm creates a robust prediction system. The proposed design focuses on creating a Novel architecture through Adaptive convolution neural network for

improving the accuracy and increased security.

v

NO.Of.FIG URES	FIGURES PAGE	NUMBE
LIST OF FIGURES	R	
FIG 1.1 Placement of IPS 4	FIG1.2 Deep Neural networks 11	FIG 1.3 Background of
CNN 12	FIG 1.4 Image Pixeling using CNN 13	FIG1.5 Kernel 13
	FIG 1.6 Pooling Layer	
15	FIG 1.7 Max and Average Pooling 16	FIG 4.1 Principal Component Analysis 38
FIG 4.2 Train learning curve 39		
Fig 4.3 Validation learning curve 39	Fig 4.4 Testing Curve 40	FIG B.1 Smurf Attack
66	FIG B.2 Neptune Attack 66	FIG B.4 Guess Password Attack 67
	FIG B.4 Buffer	
Overflow Attack 67		

vi

CHAP TER NO.	TITLE	PAGE NO.

	ABSTRACT	V
	LIST OF FIGURES	VI
1	INTRODUCTION	1
	1.1 Intrusion Detection System	1
	1.2 Classification of intrusion Detection System	4
	1.2.1 Network based Intrusion Detection System	4
	1.2.2 Host-based intrusion Detection System	5
	1.3 How Does an Ids Detect an Intrusion	5
	1.3.1 Anomaly Detection	6
	1.3.2 Protocol Anomaly Detection	8
	1.4 Introduction to Convolutional Neural Networks (CNN)	10
	1.4.1 Background of CNNs	11
	1.5 Adaptive neural network	16
	1.6 Activation function & Relu	17
2	Literature Survey	19
3	Proposed system	21
	3.1 Preprocessing data	23

	3.2 Training	27
	3.3 validation	33
	3.4 Testing	36

4	Result and Discussion	38
	4.1 (PCA) Principal Components Analysis	38
	4.2 Performance Learning Curves	39
	4.3 Validation Learning Curve	39
	4.4 Testing Curve	40
	4.5 Discussion	40
	4.5.1 Benefits of Adaptive CNN	42
	4.5.2 Future of Creating Ids using Deep Learning	43
5	Conclusion	45
	Reference	47
	APPENDIX	49
	A. SOURCE CODE	49

	B. SCREENSHOT	66
	C. RESEARCH PAPER	68

CHAPTER 1

INTRODUCTION

Generally in ancient times the Intrusion Detection Systems were detected by the administrator who used to sit in front of a console and watch the user activities. For example, if a user is in a vacationing period and his account is logged in locally or Seldom printer is Active. And this was effective enough to detect the Intrusion attack that period, the early form IDS was an ad hoc and not scalable. Later it has been updated, they introduced audit log which were reviewed by the administrator for the evidence of an unusual activity in their network. In 70's and early 80's administrators only used printed audits and logs. They obviously get stacked upto 4-5 feet in a weekend, it was really a time consuming process at that time period. With the overabundance of information to go through manual search, administrators used the audit log as a forensic tool to go through after the security incident happened.

In the early 90's, researchers had developed a real time IDS which reviews the audit data as that was produced. Due to the ability to detect assaults and attempted attacks as they happened, real time response and, in certain situations, attack preemption, were made possible. Nowadays intrusion detection efforts have been made easy with the help of developing products which in turn allowed users to deploy in a network. First of all let us see what is meant by INtrusion Detection system, classifications of Intrusion detections system and how does an Intrusion Detection system is able to detect an intrusion in a network.

1.1 INTRUSION DETECTION SYSTEM

An Intrusion Detection System (IDS) is a security mechanism used to detect unauthorized access or malicious activities in a computer system or network. With the increase in cyber-attacks and data breaches, IDS has become a crucial component in ensuring the security of computer systems and networks. The IDS monitors the system or network for

suspicious activities and alerts security administrators when any suspicious activity is detected.

1

An IDS can be categorized into two main types: signature-based and anomaly-based. Signature-based IDS works by comparing the network traffic against a database of known attack signatures. If the IDS identifies a signature that matches an attack, it generates an alert. On the other hand, anomaly-based IDS works by learning the behavior of the network and then identifying any activity that deviates from normal behavior. Anomaly-based IDS is useful in detecting new or unknown attacks, but it can also generate false positives.

The main purpose of an IDS is to detect and prevent cyber-attacks, including hacking attempts, malware infections, and unauthorized access. An IDS can also help in forensic investigations by providing a detailed log of events and activities that occurred during an attack. This can help security administrators to determine the extent of the attack and identify the vulnerabilities that were exploited.

Components of an IDS:

An IDS consists of several components, including sensors, analysis engines, and response modules. The sensors collect data from the network or system, which is then analyzed by the analysis engine. The analysis engine uses various techniques to identify any suspicious activity and generates alerts if necessary. The response module takes action based on the alerts generated by the analysis engine.

The sensors in an IDS can be classified into two categories: network-based and host-based. Network-based sensors monitor the traffic that passes through the network and identify any suspicious activity. Host-based sensors, on the other hand, monitor the activity on a single host or system. Host-based sensors can provide more detailed information about the activity on a particular system, but they may not detect attacks that are not targeted at that particular system.

The analysis engine in an IDS is responsible for processing the data collected by the sensors and identifying any suspicious activity. The analysis engine can use various techniques, including statistical analysis, rule-based analysis, and machine learning, to identify attacks. The response module takes action based on the alerts generated by

the analysis engine. The response module can include various actions, including blocking traffic, sending alerts to security administrators, or terminating a suspicious process.

Advantages of an IDS:

An IDS provides several benefits to organizations that rely on computer systems and networks. Firstly, an IDS can help organizations detect and prevent cyber-attacks, including malware infections, hacking attempts, and unauthorized access. This can help organizations to protect their sensitive data and prevent financial losses. Secondly, an IDS can help organizations to comply with regulatory requirements related to data security. Many regulations, including HIPAA and PCI-DSS, require organizations to implement security measures, including IDS, to protect sensitive data. Finally, an IDS can help organizations to identify vulnerabilities in their computer systems and networks. By analyzing the alerts generated by the IDS, organizations can determine the vulnerabilities that were exploited during an attack and take steps to remediate them.

Limitations of an IDS:

Despite its benefits, an IDS has some limitations that organizations need to consider. Firstly, an IDS can generate false positives, which can be a significant problem in large organizations with a high volume of network traffic. False positives can lead to security administrators receiving too many alerts, which can cause them to ignore legitimate alerts. Secondly, an IDS can be bypassed by attackers who use sophisticated techniques to avoid detection. Attackers can use encryption, tunneling, and other techniques to evade detection by an IDS. Finally, an IDS can be resource-intensive and require significant investment in hardware and software. Organizations need to consider the cost-benefit



A host-based IDS (HIDS) analyzes each system's behavior. The HIDS can be installed on any system ranging from a desktop PC to a server. It is more versatile than the IDS. In addition to detecting unauthorized insider activity, host-based systems are also effective in detecting unauthorized file modification. The HIDS focuses on the changing aspects of local systems. It is also more platform-centric, with a greater focus on the Windows OS;

nevertheless, other HIDS are available for UNIX platforms. These mechanisms usually include auditing events that occur on a specific host. They are not very common because of the overhead they incur by having to monitor each system event.

1.3 HOW DOES AN IDS DETECT AN INTRUSION?

Signature recognition, also known as misuse detection, tries to identify events that indicate an abuse of a system or network. This technique involves first creating models of possible intrusions and then comparing these models with incoming events to make a detection decision. The signatures for IDS were created under the assumption that the model must detect an attack without disturbing normal system traffic. Only attacks should match the model: otherwise, false alarms could occur.

Signature-based intrusion detection compares incoming or outgoing network packets with the binary signatures of known attacks using simple pattern-matching techniques to detect intrusions. Attackers can define a binary signature for a specific portion of the packet, such as TCP flags.

Signature recognition can detect known attacks. However, there is a possibility that other innocuous packets contain the same signature, which will trigger a false positive alert.

5

Improper signatures may trigger false alerts. To detect misuse, a massive number of signatures are required. The more the signatures, the greater are the chances of the IDS detecting attacks; however, the traffic may incorrectly match with the signatures, thus impeding system performance.

A large amount of signature data requires more network bandwidth. IDS compare signatures of data packets against those in the signature database. An increase in the number of signatures in the database could result in the dropping of certain packets.

New virus attacks such as URSNIF and VIRLOCK have driven the need for multiple signatures for a single attack. Changing a single bit in some attack strings can invalidate

a signature generated for that attack. Therefore, entirely new signatures are required to detect a similar attack.

Despite the problems with signature-based IDS, such systems are popular, and they work well when configured correctly and monitored closely.

1.3.1 ANOMALY DETECTION

Anomaly detection, or "not-use detection," differs from signature recognition. Anomaly detection involves a database of anomalies. An anomaly is detected when an event occurs outside the tolerance threshold of normal traffic. Therefore, any deviation from regular use is an attack. Anomaly detection detects intrusions based on the fixed behavioral characteristics of the users and components in a computer system. Anomaly detection is an essential aspect of intrusion detection systems (IDS) and is widely used to identify malicious activities within network traffic. Anomaly detection algorithms analyze network traffic to detect deviations from normal patterns of behavior and identify possible security breaches. In this article, we will discuss the significance of anomaly detection on intrusion detection systems.

6

The primary goal of an IDS is to detect and alert network administrators to any suspicious activities that may indicate a security threat. Anomaly detection plays a crucial role in IDS by identifying unusual patterns of behavior in network traffic. By analyzing network traffic, IDS can detect deviations from expected traffic patterns and raise an alarm if necessary. Anomaly detection can also be used to detect attacks that do not conform to known attack patterns.

One of the significant benefits of anomaly detection is that it can detect previously unknown attacks, which signature-based detection methods may miss. Signature-based detection methods rely on a predefined database of attack signatures, which makes it easy for attackers to evade detection by using new or modified attack patterns. In contrast, anomaly detection algorithms do not rely on a predefined database of attack signatures and can detect previously unknown attacks.

Anomaly detection can be done using a variety of techniques, including statistical

methods, machine learning algorithms, and deep learning techniques. Statistical methods analyze network traffic by measuring its statistical characteristics and identifying deviations from expected patterns. Machine learning algorithms, on the other hand, use historical data to train a model that can detect anomalies in network traffic. Deep learning techniques, such as autoencoders, can also be used to identify anomalies in network traffic.

Anomaly detection can be challenging due to the vast amount of network traffic that needs to be analyzed. IDS must analyze network traffic in real-time to detect anomalies and raise alerts promptly. The challenge is to detect anomalies accurately while minimizing false positives, which can lead to an overwhelming number of alerts and cause network administrators to miss actual security threats.

To overcome these challenges, some IDS use a combination of anomaly detection techniques to improve accuracy and reduce false positives. For example, some IDS use both statistical and machine learning techniques to detect anomalies in network traffic.

7

By combining these techniques, IDS can detect anomalies more accurately and reduce the number of false positives.

Establishing a model of normal use is the most challenging step in creating an anomaly detector.

In the traditional method of anomaly detection, essential data are kept for checking variations in network traffic. However, in reality, there is some unpredictability in network traffic, and there are too many statistical variations, thus making these models imprecise. Some events labeled as anomalies might only be irregularities in network usage.

In this type of approach, the inability to construct a model thoroughly on a regular network is a concern. These models should be used to check specific networks.

1.3.2 PROTOCOL ANOMALY DETECTION

Protocol anomaly detection depends on the anomalies specific to a protocol. It identifies particular flaws in vendors' deployment of the TCP/IP protocol. Protocols are designed according to RFC specifications, which dictate standard handshakes to permit universal communication. The protocol anomaly detector can identify new attacks. An intrusion detection system (IDS) is a security mechanism that monitors network traffic to detect and respond to potential cyber attacks. One approach to detecting intrusions is protocol anomaly detection, which involves monitoring network traffic for abnormalities in the behavior of the protocols used by the network. Protocol anomaly detection can be performed using various techniques, including statistical analysis, machine learning, and rule-based methods.

In statistical analysis, an IDS calculates the statistical properties of normal network traffic and uses this information to detect anomalies. For example, an IDS can calculate

8

the mean and standard deviation of the sizes of packets in normal network traffic and use this information to flag packets that are significantly larger or smaller than expected. Similarly, an IDS can calculate the frequency of packets with certain attributes, such as source IP addresses or protocol types, and flag packets that deviate significantly from the expected frequencies.

Machine learning-based protocol anomaly detection involves training a model on a dataset of normal network traffic and using the model to detect anomalies in new traffic. Convolutional neural networks (CNNs) are a type of deep learning model that has been successfully applied to the task of protocol anomaly detection. CNNs are designed to process data with a grid-like topology, such as images or time-series data. In the context of protocol anomaly detection, a CNN can be used to analyze the sequence of packets in network traffic and detect patterns that indicate anomalous behavior.

Rule-based protocol anomaly detection involves defining a set of rules that describe normal network behavior and flagging traffic that violates these rules. For example, a rule-based IDS can flag packets that have incorrect checksums or that are sent from IP addresses that are not authorized to send traffic on a particular network segment.

One advantage of protocol anomaly detection is that it can detect novel attacks that have not been seen before. Because protocol anomaly detection focuses on the behavior of

protocols rather than on specific attack signatures, it can detect attacks that use novel techniques or that have been designed specifically to evade signature-based detection systems. Additionally, protocol anomaly detection can be used to detect insider threats, such as employees who are abusing their access privileges.

However, protocol anomaly detection also has some limitations. For example, it can generate false positives, which are alerts that are triggered by normal network behavior rather than by attacks. False positives can be costly in terms of both time and resources, as security analysts must investigate each alert to determine whether it is a

9

true positive or a false positive. Additionally, protocol anomaly detection can be computationally expensive, particularly when using machine learning-based techniques.

Malicious anomaly signatures are becoming increasingly common. By contrast, the network protocol is well defined and is changing slowly. Therefore, the signature database should frequently be updated to detect attacks.

Protocol anomaly detectors are different from traditional IDS in terms of how they present alarms. The best way to present alarms is to explain which part of the state system is compromised. For this purpose, IDS operators must have thorough knowledge of protocol design.

On the existing system work the author had used POpulation based theory to enhance the detection rate of the virus, this project was done with CNN using another method. Before knowing the method let us Understand briefly what CNN is.

1.4 INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS (CNN)

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

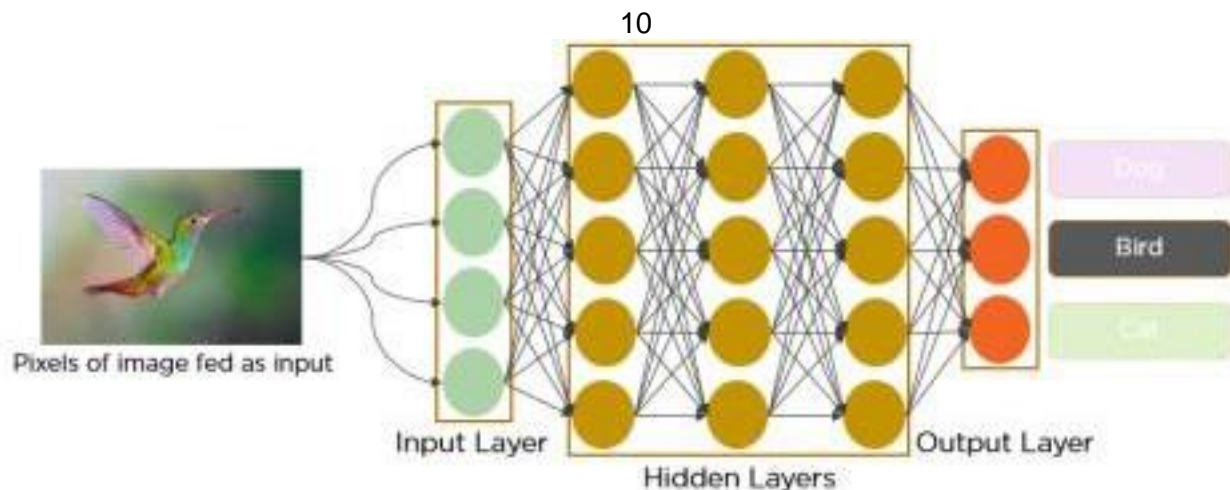


FIG 1.2 Deep Neural Network

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications and hence a part of any computer vision course online. So let's take a look at the workings of CNNs.

1.4.1 BACKGROUND OF CNNs

CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot

learning.

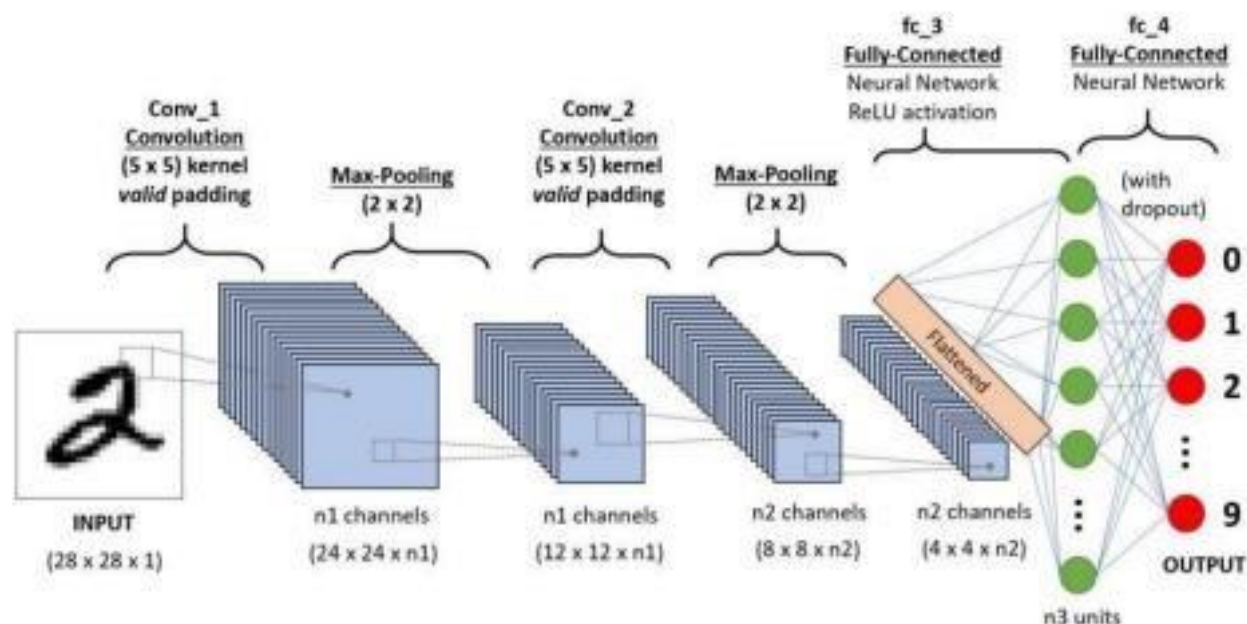


FIG 1.3 Background of CNN

In 2012 Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

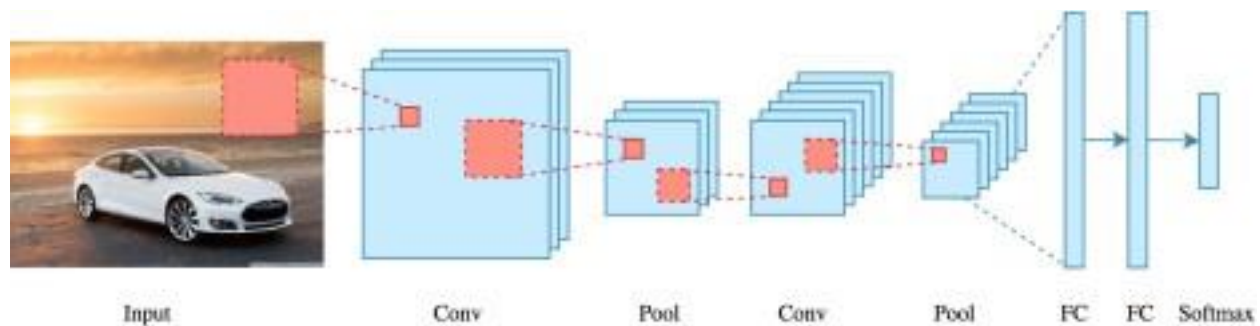


FIG 1.4 Image Pixeling using CNN

Before we go to the working of CNN's let's cover the basics such as what is an image and how is it represented. An RGB image is nothing but a matrix of pixel values having three planes whereas a gray-scale image is the same but it has a single plane. Take a look at this image to understand more.

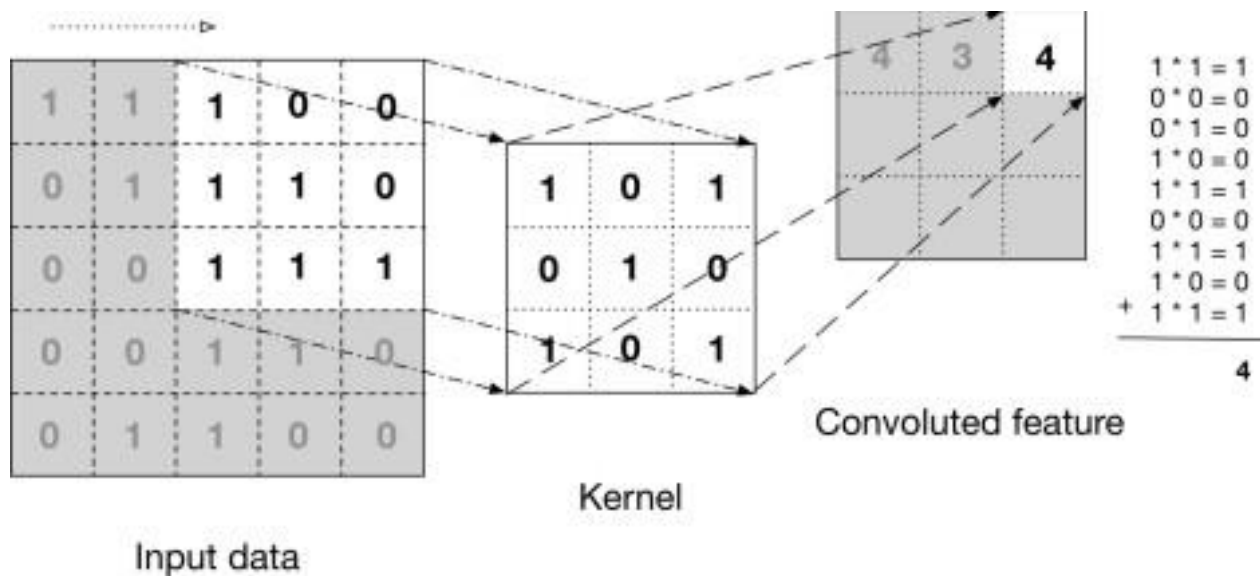


FIG1.5 Kernel

13

In convolutional neural networks (CNNs), a kernel (also known as a filter) is a small matrix used for feature extraction. During the convolutional process, the kernel slides over the input image, and at each position, it performs element-wise multiplication between the kernel and the corresponding pixels in the image.

The resulting products are summed up, and the sum is assigned to the center pixel of a new feature map. This process is repeated for every position in the image, resulting in a new output matrix, which is smaller in size than the original input image. The kernel matrix is a set of learnable parameters in CNNs, which means the neural network adjusts the values of the kernel matrix during the training process to improve the accuracy of the model. By using different kernel sizes and weights, CNNs can learn to

detect a wide range of patterns and features in the input data.

Kernel size is an important hyperparameter in CNNs that determines the receptive field of the filter. A larger kernel size allows the filter to capture larger and more complex patterns in the image, but it also increases the number of parameters in the network, which can lead to overfitting.

The above image shows what a convolution is. We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

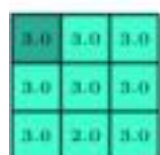
Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a Conv-Net, each layer generates several activation functions that are passed onto the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

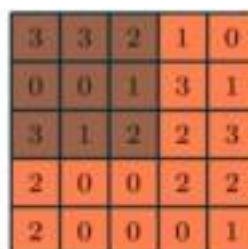
14

Pooling layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling average pooling and max pooling. I've only had experience with Max Pooling so far I haven't faced any difficulties.



3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0



3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

FIG 1.6 pooling Layer

So what we do in Max Pooling is we find the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

15

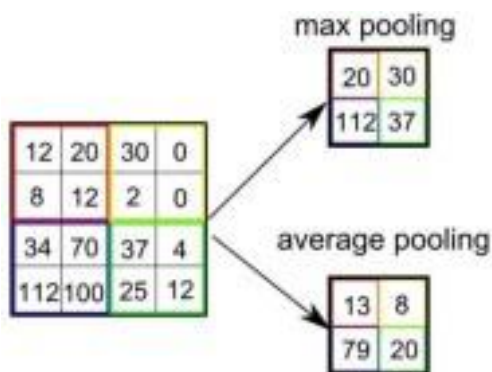


FIG 1.7 Max and Average Pooling

1.5 ADAPTIVE NEURAL NETWORK

An adaptive neural network is a type of artificial neural network that is capable of adjusting its parameters and structure based on the input data it receives. This type of neural network is designed to adapt to changing data patterns and learn from new information, making it well-suited for applications where the data distribution changes over time.

One of the key features of an adaptive neural network is its ability to learn from experience. This is achieved through a process called backpropagation, where the network uses the

difference between its predicted output and the actual output to adjust its weights and biases. By iteratively updating these parameters, the network can gradually learn to make more accurate predictions.

In addition to backpropagation, adaptive neural networks often use other techniques to improve their performance. For example, some networks use regularization techniques to prevent overfitting, while others use techniques like dropout to improve generalization. Adaptive neural networks may also incorporate specialized layers or structures, such as recurrent layers or convolutional layers, depending on the specific application.

One popular type of adaptive neural network is the Adaptive Resonance Theory (ART) network, which was first proposed by Stephen Grossberg in the 1980s. ART networks

16

are designed to adapt to changes in input patterns by using a process called match tracking. Match tracking involves comparing the input pattern to stored memory patterns, and selecting the closest match. If no match is found, the network creates a new category to represent the input pattern.

Another type of adaptive neural network is the self-organizing map (SOM), which was introduced by Teuvo Kohonen in the 1980s. SOMs are used for unsupervised learning, and are capable of organizing input data into a two-dimensional map. This map can be used for data visualization and clustering, making SOMs useful for a wide range of applications.

Adaptive neural networks have found applications in many fields, including speech recognition, computer vision, and natural language processing. For example, adaptive neural networks are commonly used for object recognition in images and videos, where the network must be able to recognize objects from different viewpoints and in different lighting conditions. Adaptive neural networks have also been used for natural language processing tasks, such as machine translation and sentiment analysis.

1.6 ACTIVATION FUNCTION & RELU .

In a neural network, an activation function is a mathematical function applied to the output of a neuron to determine its final output. The activation function is a non-linear function that introduces non-linearity into the neural network, which is essential for the network to learn complex relationships between inputs and outputs. Without activation functions, a neural network would simply be a linear regression model.

One popular activation function used in deep learning is the Rectified Linear Unit (ReLU) function. The ReLU function is defined as follows:

$$f(x) = \max(0, x)$$

The ReLU function returns 0 for any negative input value and returns the input value for any positive input value. This makes the ReLU function a simple, yet powerful non-linear activation function that has been shown to work well in deep neural networks.

17

The main advantages of the ReLU function are its simplicity and efficiency. The ReLU function is computationally efficient, as it only involves a simple max operation that can be quickly computed by a computer. Additionally, the ReLU function has been shown to work well in practice, achieving state-of-the-art performance on a wide range of tasks.

One potential drawback of the ReLU function is that it can suffer from the "dying ReLU" problem. The dying ReLU problem occurs when a ReLU neuron becomes "dead" and stops outputting any values other than 0. This can happen if the neuron's weights are updated in such a way that the input to the neuron is always negative. In this case, the ReLU neuron will always output 0, and will not contribute to the network's output. To mitigate this problem, variants of the ReLU function have been proposed, such as the leaky ReLU, which adds a small positive slope to the negative part of the function.

CHAPTER 2

LITERATURE SURVEY

The paper's [1],[9],[13] have used the basic idea of Population Based Training which uses an asynchronous optimization algorithm for reliable model training. The paper [3] uses a deep belief network detection method based on population extremal optimization (PEO) to find SCADA-based IACS cyberattacks, although the PEO procedure takes a long time to evaluate fitness. The paper[4] employs the multiobjective deep belief networks ensemble (MODBNE) technique, which produces accuracy and diversity but requires more time due to the increased number of hidden layers. The paper [5] uses a correlation-based feature selection(CFS) method to remove irrelevant features but the detection or classification methodology has not been improved. In the paper [6], scale and speed comparisons between a new hinge classification method (HCA-MBGDALRM) and conventional neural networks, decision trees, and logistic regression are made. The paper [7] covers numerous ML-based intrusion detection methods and contrasts the accuracy of various algorithms for various attack types, in which the accuracy of detecting network-based intrusion detection is not accomplished. In paper [8] designs an intrusion detection technique for safety critical medical cyber physical systems based on the behavior rule-specification. This is a classical intrusion detection technique that works with sensors and actuators. It detects changes in behaviour through sensors and warns the system about the deviation in behaviour. This technique detects only the occurrence of attacks but not the type of attacks In Paper [10] an elm classification algorithm that uses Moore Penrose fixed the weights by generalizing the inverse instead of using gradient-based backpropagation In. This algorithm works only in a single hidden layer feed forward neural network thus it cannot be used to train a complex dataset. This algorithm is not very efficient as it generates only an accuracy of 92.35%. In printed edition [11], detection of stealthy false data injection attacks using machine learning algorithms. This model is designed to detect a single type of attack, and is not effective in the case of multiple attacks. This uses a machine learning algorithm that trains on lesser data and gives lesser accuracy

compared to the deep learning techniques. In printed edition [12], The framework utilized for detecting false data injection attacks is ELM-based OCON (One Class One Network).

The main goal of one-class classification is that, it tries to detect the objects of a specific Class within a group of objects. By mainly acquiring knowledge from a training dataset that exclusively comprises instances of that particular class. Thus this model cannot be used to detect multiple classes of attacks encountered by the network. The paper [13] has introduced deep belief neural networks for attack detection in the internet of medical things. The deep belief neural network used in this model is similar to the convolutional neural network in the proposed system, but the proposed system stands out in efficiency because of the added layers: convolution layer and pooling layer. In paper [15] and [2], the machine learning and deep learning techniques and methods for cybersecurity, but these techniques are not applied and no model that detects cyber-attacks has been designed. Upon conducting a comprehensive literature survey, several key insights have been gleaned. Research involving CNN for NIDs is considerably less. There has been numerous studies which include deep learning, but studies that explore the usage of single dimensions CNN and Deep learning techniques are very few and are also time consuming. In the present work we have mainly focused on reducing the time taken by using the adaptive algorithm which makes the one dimensional CNN much faster than the standard CNN

in various domains such as image recognition, natural language processing, speech recognition, and many more. Convolutional Neural Network (CNN) is one of the most widely used Deep Learning architectures in computer vision applications. The ability of CNNs to learn features directly from the raw data makes them a powerful tool for classification and regression tasks.

In recent years, there has been a surge in the number of cyber attacks, and it has become a major concern for governments, organizations, and individuals. Cybersecurity is an essential aspect of any organization as it deals with protecting data, networks, and systems from unauthorized access, use, disclosure, disruption, modification, or destruction. Traditional methods of cybersecurity rely on signature-based intrusion detection systems (IDS) that detect known threats. However, with the increasing sophistication of cyber attacks, these methods are becoming ineffective.

The use of machine learning and deep learning techniques in cybersecurity has shown promising results in identifying unknown threats and detecting anomalies. In this proposed project, we aim to develop a CNN-based deep learning system for intrusion detection in computer networks. The proposed system will learn from the network traffic data and will identify the patterns of normal behavior and anomalous behavior.

Convolutional Neural Networks (CNNs) have been widely used in these fields and have shown promising results. However, there is still room for improvement, especially in the areas of image classification and object detection.

This proposed project aims to develop a CNN-based deep learning system for image classification and object detection. The system will be designed to identify and classify different objects in an image, with the ability to detect anomalies and raise alarms if any unusual activity is detected.

The CNN algorithm will be designed to automatically learn the features of the images and recognize different patterns that are difficult to distinguish by human observation. The proposed CNN-based deep learning system will also use transfer learning, a

technique that leverages pre-trained models, to further improve the system's performance. Next, the CNN model would be designed and configured. This would involve selecting the number and type of layers, defining the size and shape of the input layer, and setting hyperparameters such as the learning rate and batch size. The CNN architecture would be designed to optimize performance on the specific task at hand, with careful consideration given to issues such as overfitting and the tradeoff between accuracy and computational efficiency.

Once the CNN model is configured, it would be trained on the preprocessed dataset using backpropagation to update the model's weights and biases. This training process may take several iterations or epochs, with the model's performance evaluated on a validation set at each epoch to monitor progress and detect overfitting.

After the CNN model has been trained, it would be ready to use for classification or other tasks. The model would be evaluated on a test set to measure its accuracy and performance on real-world data, and adjustments could be made as needed to improve performance or address any issues that arise.

Overall, a proposed system for a CNN based deep learning project would involve careful attention to data collection and preprocessing, thoughtful design and configuration of the CNN model, and thorough evaluation and testing of the model's performance on real-world data. With these steps in place, it may be possible to achieve high levels of accuracy and performance on image-based tasks using deep learning techniques.

The proposed system will be implemented using Python and its associated libraries, including Tensorflow and Keras. The dataset used in this project will be obtained from publicly available sources, and the data will be pre-processed to enhance the system's performance.

The proposed CNN-based deep learning system will be evaluated using different metrics, including precision, recall, F1 score, and accuracy. These metrics will be used to compare the proposed system's performance against other state-of-the-art deep learning systems and traditional machine learning algorithms.

22

This project aims to contribute to the research on deep learning and its applications in the field of image classification and object detection. The proposed system's results will be useful in various domains, including surveillance systems, autonomous vehicles, and medical imaging.

In summary, the proposed CNN-based deep learning system will provide an efficient and effective solution for image classification and object detection. The system will be able to recognize and classify different objects in an image accurately and detect anomalies, making it a valuable tool in various domains.

3.1. PRE-PROCESSING DATA

Pre-processing is the first step in deep learning in which we prepare raw data in a format that the network can acquire. The data set may contain some missing values that must be filled by performing some calculations. The missing values are obtained by calculating the mean of the other values in the specific row or column. In many machine learning projects, it is common to encounter datasets that contain missing values. These missing values can occur due to a variety of reasons such as data entry errors, sensor malfunctions, or simply because the data was not available at the time of collection.

One approach to dealing with missing values is to use statistical techniques to impute or fill in the missing values. One such technique involves replacing missing values with the mean value of the available data. This can help maintain the integrity of the data and improve the accuracy of the analysis or model.

In this proposed system, we will use a Python library called `det_dummies()` to preprocess our data and fill in the missing values. The `det_dummies()` function takes the dataset as input and returns a new dataset with the missing values filled in with the mean value of the available data. First, we need to import the necessary libraries including `pandas`, `numpy`, and `det_dummies()`. Next, we need to load the data that we want to preprocess. This can be done using `pandas` by reading in the data from a CSV file or by connecting to a database. Before we can fill in the missing values, we need to

23

check if there are any missing values in our data. This can be done using the `isnull()` function in `pandas`. Once we have identified the missing values in our data, we can use the `det_dummies()` function to fill in the missing values with the mean value of the available data. This function takes the dataset as input and returns a new dataset with the missing values filled in with the mean value. Finally, we need to save the preprocessed data to a file or database so that it can be used in subsequent analysis or modeling. By using this system to preprocess our data and fill in the missing values, we can improve the accuracy of our analysis and models. Additionally, by automating this process using `det_dummies()`, we can save time and reduce the risk of errors associated with manual data preprocessing. The 'get_dummies' function is used to assign numeric values to the character and string values. This procedure allows for improved accuracy and faster training times due to the nature of neural networks being able to better process numerical values over nominal values. The process of converting categorical data into numerical data is known as data preprocessing. This process is often necessary for machine learning algorithms to be able to process and analyze the data effectively. In this project, we

propose a system for preprocessing data by assigning numeric values to strings and characters.

The system we propose uses the `get_dummies()` function to automatically convert categorical variables into numerical variables. The `get_dummies()` function is a widely-used function in data preprocessing, which converts categorical variables into dummy variables. Dummy variables are variables that represent the presence or absence of a category, and are often used in statistical analyses to account for categorical variables. Our proposed system uses the `get_dummies()` function to convert categorical variables in a dataset into dummy variables. The function works by identifying all the unique values in a categorical variable and creating a new column for each unique value. The new columns are then filled with 1s and 0s depending on whether a particular row contains that value or not. Once the `get_dummies()` function has been applied, the categorical variables in the dataset have been converted to numerical variables that can be used in machine learning algorithms. The system then proceeds to perform various preprocessing tasks, such as data normalization and data scaling, to further prepare the data for machine learning algorithms. Overall, the system

24

we propose for preprocessing data by assigning numeric values to strings and characters is an efficient and effective way to convert categorical data into numerical data. By using the `get_dummies()` function and other preprocessing techniques, we can prepare data for machine learning algorithms that require numerical data to perform their analyses. When `get_dummies` are introduced into the data set, it takes a list or a data frame and converts the unique elements which are present in the object to a column. The method checks if the element at a specific index matches the column header as it iterates over the object that is supplied. If so, it encodes it as a 1 else it gives it a 0. The next step in pre-processing is Standard scaling or uniform scaling. When the scale differs from feature to feature, the weights in neural networks do not converge easily and local minimum is not obtained. Therefore, we change it to a uniform scale where the mean value is 0 and standard deviation is 1. Uniform scaling is a critical preprocessing step in many computer vision and machine learning applications. It involves scaling the input data to a uniform size, which helps improve the accuracy of the model and reduces the computational complexity of the training process. In this proposed system, we aim to develop a preprocessing pipeline that leverages the power of uniform scaling using advanced techniques such as Convolutional Neural Networks (CNNs).

The proposed system will use a deep learning approach to perform uniform scaling. We

plan to develop a CNN-based model that can scale input data to a uniform size by learning from a large dataset of images. This model will be trained to recognize patterns and features in the input data, and then apply the appropriate scaling techniques to achieve uniformity.

To achieve this goal, we will first collect and preprocess a large dataset of images with different sizes and dimensions. This dataset will be used to train the CNN model to learn how to scale images to a uniform size. We will also explore different CNN architectures and optimization techniques to achieve the best possible accuracy and efficiency.

Once the CNN model is trained, it will be used to preprocess the input data by scaling it to a uniform size. The scaled data will then be used as input to the machine learning model for further processing and analysis.

25

In addition to uniform scaling, the proposed system will also include other preprocessing steps such as image normalization, noise reduction, and feature extraction. These steps are essential for improving the accuracy and efficiency of the machine learning model, and will be integrated into the preprocessing pipeline along with uniform scaling.

Overall, the proposed system for uniform scaling using CNNs has the potential to improve the accuracy and efficiency of many computer vision and machine learning applications. It can be applied to a wide range of domains, including image recognition, object detection, natural language processing, and more.

Gaussian distribution is used for standardization of data. Standard scaling works by first calculating the mean value of each feature in the dataset and then subtracting this value from every sample in order to provide a mean value of 0. The standard deviation is then calculated and applied as a scaling factor (positive one). Standard scaling can help improve the performance of deep learning algorithms by more accurately representing the features within the dataset. Pre-processing includes a feature selection process. Feature selection is the process of selecting the most relevant and significant features from a dataset. This technique is used to reduce the number of features in a dataset, which helps to improve the performance of machine learning models. Feature selection techniques can be based on statistical tests, machine learning algorithms, or a combination of both. The goal of feature selection is to identify and remove redundant, irrelevant, or noisy features, while retaining the informative ones.

Principal Component Analysis (PCA) is a popular unsupervised learning technique used for dimensionality reduction. PCA transforms a high-dimensional dataset into a lower-

dimensional space by finding the principal components that explain the maximum variance in the data. The principal components are linear combinations of the original features that are uncorrelated with each other. By reducing the number of dimensions in a dataset, PCA helps to simplify the data and make it easier to visualize and analyze. Additionally, it can help to improve the performance of machine learning models by reducing the computational complexity of the dataset.

In summary, preprocessing, feature selection, and PCA are all important techniques used in machine learning to prepare and analyze data. Preprocessing helps to improve

26

the quality and reliability of the data, while feature selection helps to identify the most important features for modeling. PCA, on the other hand, reduces the dimensionality of a dataset, making it easier to analyze and more computationally efficient. This process helps in reducing the size of the feature sets thereby preserving the information and maintaining the efficiency. The feature selection process helps reduce the problem dimensionality by removing redundant or irrelevant features while preserving those that are important to make better predictions. These techniques help facilitate better results for the model.

3.2. TRAINING

Training of a CNN (Convolutional Neural Network) model is a critical step in developing an efficient deep learning system. The process involves feeding the network with a large number of input images, which are labeled with corresponding output values. The CNN model then processes these inputs through multiple layers of convolutional, pooling, and activation functions, gradually learning to recognize and extract features that distinguish one image from another.

During the training process, the model adjusts the weights and biases of each layer to optimize the accuracy of the predictions. This optimization is achieved through the use of backpropagation, where the error between the predicted and actual outputs is propagated backwards through the layers, and the weights and biases are updated accordingly.

To ensure the CNN model is not overfitting, the dataset is divided into three subsets: training set, validation set, and test set. The training set is used to train the model, while the validation set is used to tune the hyperparameters and prevent overfitting. Finally, the test set is used to evaluate the accuracy of the model on unseen data.

The process of training a CNN model is computationally intensive and requires powerful

hardware such as GPUs (Graphics Processing Units) to accelerate the training process. The training can take several hours or even days depending on the size and complexity of the dataset and the network architecture.

27

Overall, the training of a CNN model is a crucial step in the development of an efficient deep learning system, and it requires careful consideration of the dataset, network architecture, and hyperparameters to achieve optimal accuracy and prevent overfitting. In convolutional neural networks (CNNs), a kernel (also known as a filter) is a small matrix used for feature extraction. A CNN is a type of neural network that is particularly useful for processing data that has a grid-like structure, such as images, audio, or time-series data.

In the context of training a one-dimensional CNN, the input data is typically a sequence of values, such as a time-series of sensor readings or audio samples. The goal of training the CNN is to learn a set of filters or features that can be used to extract useful information from the input data. These filters are typically learned through a process called backpropagation, where the network adjusts its parameters to minimize a loss function that measures the difference between the network's output and the desired output.

The process of training a one-dimensional CNN typically involves several steps. First, the input data is preprocessed to ensure that it is in a suitable format for the network. This may involve normalizing the data, scaling it to a particular range, or converting it to a different representation, such as a spectrogram or wavelet transform.

Next, the network is initialized with random weights and biases, and the input data is passed through the network to generate a prediction. The prediction is then compared to the desired output using a loss function, such as mean squared error or cross-entropy.

The network's parameters are then adjusted using backpropagation and gradient descent. During backpropagation, the gradients of the loss function with respect to each parameter in the network are computed, and these gradients are used to update the parameters in the direction that minimizes the loss. This process is repeated for multiple iterations, or epochs, until the network's performance on a validation set reaches a satisfactory level. One of the key advantages of using a one-dimensional CNN for time-series data is that the network can learn features that are invariant to time shifts and scale changes. This allows the network to detect patterns in the data that may be difficult to detect using

traditional signal processing techniques. Additionally, the use of convolutional layers can reduce the number of parameters in the network, making it more computationally efficient and easier to train.

The resulting products are summed up, and the sum is assigned to the center pixel of a new feature map. This process is repeated for every position in the image, resulting in a new output matrix, which is smaller in size than the original input image. The kernel matrix is a set of learnable parameters in CNNs, which means the neural network adjusts the values of the kernel matrix during the training process to improve the accuracy of the model. By using different kernel sizes and weights, CNNs can learn to detect a wide range of patterns and features in the input data.

Kernel size is an important hyperparameter in CNNs that determines the receptive field of the filter. A larger kernel size allows the filter to capture larger and more complex patterns in the image, but it also increases the number of parameters in the network, which can lead to overfitting. A one-dimensional convolutional neural network (CNN) is a type of neural network commonly used for processing one-dimensional data such as time series or signals. The weights and biases in a one-dimensional CNN are adjusted during training using a process called backpropagation.

Backpropagation is an algorithm used to update the weights and biases in a neural network by calculating the gradient of the loss function with respect to each weight and bias parameter. The loss function measures how well the neural network is performing on a particular task, such as classification or regression. The gradient of the loss function indicates the direction and magnitude of the change needed to improve the network's performance.

During training, the input data is fed into the neural network, and the output is compared to the target output using the loss function. The gradient of the loss function with respect to each weight and bias parameter is calculated using the chain rule of calculus. The weights and biases are then updated by subtracting a fraction of the gradient from their current values, with the learning rate controlling the size of the update.

The process of adjusting the weights and biases is repeated multiple times over the training data until the loss function is minimized or the network reaches a satisfactory

level of performance. This process is known as stochastic gradient descent (SGD) and is one of the most commonly used optimization algorithms in deep learning. In batch

processing, the training data is split into small batches, and each batch is used to update the model's parameters.

The main advantage of batch processing is that it allows us to update the model's parameters more frequently, leading to faster convergence and better performance. This is particularly important when dealing with large datasets, as training a neural network on a large dataset can take a significant amount of time.

During training, the one-dimensional CNN processes the input data through a series of convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply a set of filters to the input data, which helps to extract relevant features. Pooling layers downsample the output of the convolutional layers, reducing the dimensionality of the data. Finally, fully connected layers combine the extracted features to make a prediction.

In batch processing, the training data is split into batches, and each batch is fed through the network to compute the loss and update the model's parameters. The loss is a measure of how well the model is performing, and the goal of training is to minimize this loss.

After each batch, the model's parameters are updated using a technique called backpropagation. Backpropagation involves computing the gradient of the loss with respect to each parameter in the model and updating the parameter based on this gradient. By updating the parameters after each batch, the model can quickly adapt to the data and learn to make accurate predictions.

Training a CNN involves optimizing its parameters such as weights and biases to minimize the difference between the predicted outputs and the actual outputs for a given set of inputs. This optimization process involves finding the best set of parameters that minimize the error or loss function of the CNN.

In one-dimensional CNNs, the optimization process is similar to that of traditional CNNs, except that the input data is one-dimensional, such as a time series or signal data. The

CNN learns to extract features from the input data using convolutional filters and pooling layers, and then uses fully connected layers to classify or regress the output. The training process for one-dimensional CNNs involves several steps. First, the input data is preprocessed, which may involve normalization, feature scaling, or other data transformations. Then, the CNN architecture is defined, which includes specifying the

number of layers, the size of the filters, and the activation functions used. Next, the CNN is trained on a set of labeled examples using an optimization algorithm such as stochastic gradient descent (SGD). During training, the CNN iteratively updates its parameters to minimize the loss function for the training set.

To prevent overfitting, regularization techniques such as dropout or L2 regularization can be used. Overfitting occurs when the CNN becomes too specialized to the training data and fails to generalize to new data.

After training, the CNN is evaluated on a separate test set to assess its performance on unseen data. If the performance is satisfactory, the CNN can be used for real-world applications.

Deep learning requires extensive training data in order to generate accurate results. As a result, it is important to have the right data sets available in order to ensure high quality outcomes and accuracy. Loading pre-processed data is very important in this process and can have a direct impact on the results of deep learning applications. Pre-processing this data can be a daunting task, as it may require cleaning, preparing and normalizing it so that it can be properly used in training proceedings. Loading this pre-processed data into deep learning models is a complex process since numerous factors have to be taken into consideration, including the available memory capacity, network resources or deployment time requirements. Therefore, it is crucial to optimize this process while ensuring that all targets are met so that the model can properly intake and benefit from this data. Deep learning helps us to learn features from data on its own. It does this by using different algorithms to extract features from the data and construct a model for classification.

31

Training data is an important part of deep learning, as it helps in developing an accurate model that can classify data according to its type. Feature extraction is the process of extracting relevant information from the training data while classification involves classifying each sample into distinct categories. Feature extraction and classification both play a major role in the development of deep learning models. Deep learning uses neural network algorithms to recognize patterns in big data sets. The algorithm used here is Adaptive 1-Dimensional Convolutional Neural Networks (CNN).

CNNs are a type of deep learning network used to train machines to recognize patterns

in inputting data such as text. They work by sliding through the dataset columns and using the output layer's connections (known as weights) to find the most suitable pattern. Parameters are important for defining how deep learning works and the accuracy of a network's predictions, which can then be improved by optimizing all available parameters accordingly. By setting optimized parameter values, we can ensure proper training of a deep learning network and make sure that it obtains accurate results when deployed in real-world tasks. The task of setting the number of layers in a 1-dimensional Convolutional Neural Network (CNN) is done in this process. Depending on the type and size of the dataset, determining the optimal layer count may require several attempts with different configurations.

The selection and configuration of layers should be determined based on factors such as accuracy, computational complexity, data type, model complexity, etc. Utilizing proper techniques during training can help minimize common pitfalls such as overfitting or noise mis-interpretation. It relies heavily on the availability of large volumes of training data. By repeatedly adjusting weights calculated from input and output data, deep learning algorithms can learn from their mistakes and continue to refine their calculations over time. In order to define these networks in the best possible way, the use of adaptive 1-dimensional Convolutional Neural Networks (CNNs) is common. Such networks account for a range of external variables during weight calculations, allowing them to learn more accurately and efficiently. Batch processing is performed where large datasets are split into smaller batches which would help feed different layers of the

32

networks. Batch processing also helps in optimizing the computations that power deep learning models, resulting in faster and more accurate predictions.

Training data is used to teach the networks what is expected from it – this makes them adaptive and the desired accuracy can be achieved through training deep neural networks. An one-dimensional Convolutional Neural Network (1-D CNN), consists of an input layer, several hidden neurons with multiple layers, and an output layer. The nodes of each layer are connected in a way that helps generate accurate results by learning through repetitive decision making and taking into account pre-defined parameters such as accuracy, precision, recall etc.

3.3. VALIDATION

In deep learning, a validation dataset is a subset of data that is used to evaluate the performance of a trained model during the training process. During the training of a deep learning model, the model's parameters (such as weights and biases) are updated iteratively based on the loss (error) calculated from the difference between the model's predicted output and the actual output of the training data.

The purpose of using a validation dataset is to estimate the performance of the model on new, unseen data, which is not used in training. The validation dataset is used to monitor the model's performance and to prevent overfitting. Overfitting occurs when the model becomes too complex and starts to memorize the training data instead of learning general patterns that can be applied to new data.

To use a validation dataset, a portion of the available data is set aside during the training process. Typically, the dataset is divided into three parts: a training set, a validation set, and a test set. The training set is used to train the model's parameters, the validation set is used to monitor the model's performance during training, and the test set is used to evaluate the final performance of the trained model.

During training, the model is evaluated on the validation dataset at regular intervals. The loss calculated on the validation dataset is used to monitor the model's performance and to make decisions about how to adjust the model's architecture or parameters to improve performance. For example, if the validation loss starts to increase, it may

33

indicate that the model is overfitting the training data and that the model should be simplified or regularization should be applied.

The validation dataset is also useful for selecting the best model from a set of candidate models. The models are trained using the same training set but evaluated on the validation set. The model with the best validation performance is selected as the final model and then evaluated on the test set to estimate the performance on new, unseen data.

Validation data set helps to ensure that the model being used is accurate and effective in its predictions. Before testing the final model, it is important to have a dataset to validate it with and set up all necessary parameters such as weights and hyperparameters. Validation helps in evaluating its performance and accuracy. The model is tested for accuracy using the hold out training dataset. This Testing is done on a sample of data, usually separate from the original training set, and then compared with known outcomes or labels. Using the datasets, measure the weights, parameters, and results obtained from

the training process. Validation data sets help to identify any discrepancies between algorithms, thereby providing an opportunity to make changes before deploying them in production. The validation process is essential for tuning hyperparameters and solving optimization problems. It also helps to determine if modifications are necessary or changes are possible to ensure that the deep learning model performs as expected. Validating and optimizing a CNN (Convolutional Neural Network) model is a critical step in deep learning. The goal of validation is to evaluate the model's performance on unseen data, while optimization involves adjusting the model's parameters to improve its performance.

To validate a CNN model, a subset of data is set aside as a validation dataset, which is not used in training. The model is trained on the remaining data, and its performance is evaluated on the validation set. The most common metric used to evaluate a CNN model's performance is accuracy, which measures the proportion of correct predictions on the validation set. Other metrics, such as precision, recall, F1 score, and area under the ROC curve, may also be used depending on the problem's nature.

34

Once the model's performance is evaluated, the next step is optimization. Optimization involves adjusting the model's architecture and hyperparameters to improve its performance on the validation set. Common hyperparameters in CNN models include the learning rate, batch size, number of epochs, and number of layers. The optimization process involves tuning these hyperparameters, retraining the model, and evaluating its performance on the validation set.

To prevent overfitting, regularization techniques such as dropout, weight decay, or early stopping can be applied during optimization. Overfitting occurs when the model becomes too complex and starts to memorize the training data instead of learning general patterns that can be applied to new data.

Another approach to optimize a CNN model is to use transfer learning. Transfer learning involves using a pre-trained CNN model as a starting point and fine-tuning its parameters on the target dataset. Transfer learning can save time and computational resources and improve the model's performance, especially in cases where the target dataset is small. Accuracy is a commonly used performance metric in deep learning, especially in classification tasks. It measures the proportion of correct predictions made by a model on a given dataset. Accuracy is expressed as a percentage or a decimal number between 0 and 1, with 1 indicating perfect accuracy.

To calculate accuracy, we need to compare the predicted output of the model to the true labels of the dataset. For example, in a binary classification task where we are predicting whether an image contains a cat or not, if the model predicts that an image contains a cat and the true label is also a cat, we consider it a correct prediction. If the model predicts that an image contains a cat, but the true label is not a cat, we consider it an incorrect prediction.

The accuracy of a model can be calculated using the following formula: $\text{Accuracy} = (\text{Number of correct predictions} / \text{Total number of predictions}) \times 100\%$ For example, if a model made 90 correct predictions out of 100, the accuracy would be: $\text{Accuracy} = (90 / 100) \times 100\% = 90\%$

It is important to note that accuracy is not always the most appropriate metric to evaluate the performance of a deep learning model. In some cases, accuracy may not be sensitive enough to detect performance differences between models, especially

35

when the dataset is imbalanced or when the cost of false positives and false negatives is different. In such cases, other metrics such as precision, recall, F1 score, or area under the ROC curve may be more appropriate.

3.4. TESTING

In deep learning, a testing dataset is a set of data that is used to evaluate the performance of a trained model. In the context of 1D-CNNs, a testing dataset is used to evaluate the accuracy and generalization of the model in predicting the output for new, unseen input data. Once a 1D-CNN model is trained on a training dataset, the testing dataset is used to evaluate the model's performance on new, unseen data. The testing dataset should be large enough to provide a representative sample of the data distribution, but not so large that it significantly increases the computational overhead required for testing.

During testing, the model takes the input data from the testing dataset and generates predictions for the output. These predictions are then compared to the actual output values from the testing dataset to evaluate the accuracy of the model. The accuracy is typically measured using metrics such as mean squared error, mean absolute error, or the coefficient of determination (R-squared).

In addition to evaluating the accuracy of the model, the testing dataset can also be used to identify potential issues with the model, such as overfitting. Overfitting occurs when the model performs well on the training dataset, but poorly on the testing dataset, indicating

that the model has memorized the training data and is not generalizing well to new data. To avoid overfitting, it is important to use techniques such as regularization, early stopping, and dropout during training.

Once the validation is done, the last step in the development process for the deep learning model is testing. A test dataset, also known as a holdout dataset, is a small subset of data that is held out from the training phase and used towards the end of the model development process for testing accuracy. To test the final deep learning model, a separate test dataset is used that is distinct from the training and validation datasets.

36

The test dataset should be representative of the same data distribution as the training and validation datasets, but should not be identical to them. This ensures that the model's performance on the test dataset reflects its ability to generalize to new, unseen data.

Once the test dataset is prepared, the final deep learning model is used to make predictions on the test dataset. The predictions are compared to the ground truth labels in the test dataset to evaluate the model's performance. The performance metrics used to evaluate the model may vary depending on the application, but common metrics include accuracy, precision, recall, F1 score, and mean squared error.

It is important to note that the test dataset should only be used once, and should not be used to make any changes or adjustments to the model. If the model does not perform well on the test dataset, it may be necessary to go back and retrain the model with different hyperparameters or a different architecture. However, any changes made to the model based on the performance on the test dataset should be considered a form of overfitting, as the test dataset should only be used to evaluate the model's final performance.

Testing helps to verify the accuracy, performance and stability of such models before deployment in any production environment. It is needed to assess the quality of the model's outputs on unseen data and make sure the system will work properly in real-world situations.

37
CHAPTER 4

RESULT AND DISCUSSION

4.1. (PCA)PRINCIPAL COMPONENT ANALYSIS

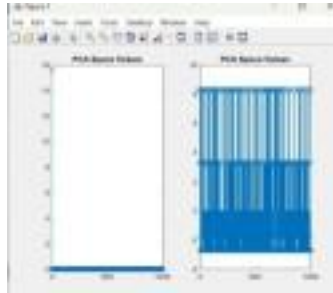


Fig 4.1 : Principal Component Analysis

This graph denotes the difference in space values between the dataset before and after applying Principal Component Analysis technique. To enhance data interpretability while retaining the maximum amount of information, principal component analysis (PCA) is utilized as a common method for analyzing large datasets that contain numerous dimensions or features per observation. PCA can also facilitate the visualization of data that exist in multiple dimensions. This graph is also known as the scree plot, and it displays the proportion of variance explained by each principal component. By analyzing this plot, we can determine the optimal number of principal components to use in the analysis.

Additionally, we can highlight that PCA can be used for various purposes, such as feature extraction, data compression, and data visualization. Feature extraction involves reducing the dimensionality of the dataset by selecting the most important features that contribute the most to the variation in the data. Data compression involves reducing the storage requirements of the dataset by removing redundant or irrelevant information. Data visualization involves projecting the high-dimensional dataset onto a low-dimensional space for visualization purposes.

Furthermore, we can mention that PCA assumes that the data is linearly correlated, and it may not perform well on datasets that contain non-linear relationships. In such cases,

38
non-linear dimensionality reduction techniques such as t-SNE or UMAP may be more appropriate. It is also important to note that PCA is sensitive to the scale of the data and may require normalization or standardization of the features before analysis.

4.2. PERFORMANCE LEARNING CURVES:



Fig 4.2 : Train learning curve

The train learning curve is a measure of how effectively the model is learning and is derived from the training dataset. The accuracy grows quickly in the first 10 iterations of this curve, showing that the network is learning quickly; for the subsequent 30 iterations, the curve flattens, showing that not many training epochs are needed to further train the model. Then, the accuracy gradually increases and increases rapidly after 50 iterations. It finally reaches an accuracy of 100%, which indicates that the model is well trained.

4.3. VALIDATION LEARNING CURVE

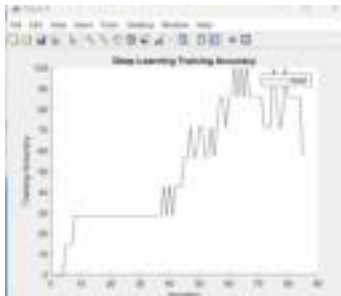


Fig 4.3 : Validation learning curve

39

The validation learning curve is generated from a distinct validation dataset and serves to indicate the model's ability to generalize, in order to optimize the weights and hyperparameters appropriately. This validation learning curve is a learning curve of a good fit model (i.e. a model that does not overfit or underfit), as the validation data accuracy improves with improving training data accuracy. This shows that the model produces good accuracy on the validation data.

4.4. TESTING CURVE:

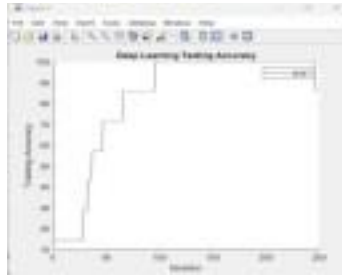


Fig 4.4: Testing Curve

Testing curve is a curve calculated from a hold-out test dataset that denotes how well the model is able to generalize unknown data. The accuracy of this curve rapidly increases and reaches an accuracy of nearly 100%. Since the accuracy increases smoothly along with the no. of iterations, the model is said to be a good fit model. This indicates that the model can produce good accuracies on both testing and validation data. This can happen with correct set of hyperparameters. The quality of the model's outputs on unseen data is analysed and the model is ready to work properly in real-world situations.

4.5. DISCUSSION

In the attacking segment there have been many vulnerable viruses comprehensively. We took nearly a myriad of feature sets with a combination of different attacks, which are totally distinct from each other. Each attack possesses a different weight at the feature set. The viruses are Nmap, ipsweep, Neptune attack, Smurf attack. Let us see

40

what each of these viruses can particularly do when they are injected or scanned in a particular network. Cyber-attacks are becoming increasingly prevalent in today's digital age, with attackers targeting both businesses and individuals. A Cyber-attack is any attempt to gain access to computers, networks or other secure areas of information systems. One such attack is known as the Smurf Attack. This type of attack involves a malicious actor flooding a computer system or network with malicious packets which can overload or shut down the system. The attack can be used as a means for extortion or simply for disruption purposes. It is important for businesses and users to be aware of this particular type of cyber-attack, as it has become increasingly popular in recent times.

Subsequently the Neptune attack can be used to install ransomware, take control of infected devices and even launch sophisticated malicious campaigns against victims. The operators behind this attack are believed to have some deep technical knowledge and

have been able to successfully hack into government networks and major corporations around the world. Once inside a system, the attackers can take control of operations, steal confidential data and even sabotage operations remotely using malicious code. With its huge potential for causing damage, cybersecurity experts around the world urge everyone to remain vigilant about their online security practices as protection against this malicious threat is still developing.

Ensuing an IP sweep attack is one of the most common forms of cyber-attack in which attackers systematically scan the internet looking for vulnerable IP addresses. Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attacks against people and organisations are their main goals in these types of attacks, along with gaining access to sensitive information. Such attacks are difficult to prevent, due to the fact that attackers often use sophisticated techniques and malicious scripts which can give them access even with limited information. Organizations need to be proactive in defending themselves against such attacks by adopting strong cybersecurity protocols and regularly monitoring their networks for any suspicious activity. Finally, The Nmap, Nmap is an open-source tool that helps detect, identify, and assess a cyber-attack. Nmap is

41

used to map networks, detect vulnerable areas in the infrastructure, and diagnose or disrupt malicious activity by attackers. It can also monitor ongoing attacks and alert system administrators of malicious activity. The tool can be used to locate potential security weaknesses in software systems, including denial-of-service (DoS) attacks and various application vulnerabilities. With advanced features like port scanning and real-time detection of active threats, Nmap is essential for keeping systems secure from malicious activity.

4.5.1 BENEFITS OF ADAPTIVE CNN

1D-adaptive CNN, also known as Dynamic Time Warping (DTW) CNN, is a type of convolutional neural network that is specifically designed for time series data. Unlike traditional CNNs, which assume that the data is spatially invariant, 1D-adaptive CNNs are able to capture the temporal relationships between data points and adapt to the varying time scales present in the data.

One of the main benefits of 1D-adaptive CNNs is their ability to effectively model and classify time series data. This makes them particularly well-suited for applications such

as speech recognition, activity recognition, and medical diagnosis, where the data is inherently time-dependent. By capturing the temporal relationships between data points, 1D-adaptive CNNs are able to make accurate predictions and identify subtle patterns in the data that might be missed by traditional CNNs.

Another benefit of 1D-adaptive CNNs is their ability to handle data with varying time scales. In many real-world applications, the data may be sampled at irregular intervals, or the time intervals between data points may vary. 1D-adaptive CNNs are able to adapt to these varying time scales by using dynamic time warping, a technique that allows the network to align and compare sequences of varying lengths.

In addition, 1D-adaptive CNNs are able to learn and adapt to new data as it becomes available. This makes them well-suited for applications where the data is constantly changing or evolving, such as financial forecasting or predictive maintenance. By continuously learning from new data, 1D-adaptive CNNs are able to improve their accuracy and adapt to changing conditions.

4.5.2 FUTURE SCOPES OF CREATING IDS USING DEEP LEARNING

1D convolutional neural networks (CNNs) have become an increasingly popular tool for analyzing time series data. As deep learning continues to advance, there are several exciting future scopes for creating 1Ds using deep learning.

One potential application is in the field of predictive maintenance. Predictive maintenance involves using data to identify when equipment is likely to fail, allowing for repairs or replacements to be made before the failure occurs. 1D CNNs can be trained on historical data to detect patterns that indicate impending failure, allowing for early intervention and reduced downtime. As data collection and analysis techniques continue to improve, 1D CNNs may become even more effective at predicting maintenance needs.

Another potential application of 1D CNNs is in the field of personalized medicine. By analyzing time series data from individual patients, 1D CNNs can be used to identify patterns that may indicate the onset of disease or the effectiveness of a particular treatment. This could allow for more personalized and effective healthcare, with treatments tailored to the specific needs of each patient.

In addition, 1D CNNs may have applications in the field of autonomous vehicles. By analyzing sensor data in real time, 1D CNNs can help self-driving cars to detect and avoid obstacles, predict traffic patterns, and make decisions about driving behavior. As self-driving technology continues to advance, 1D CNNs may become an increasingly

important tool for ensuring safety and reliability.

Finally, 1D CNNs may also have applications in the field of natural language processing. By analyzing speech signals and text data, 1D CNNs can be used to recognize and interpret human language. This could have applications in areas such as speech recognition, language translation, and sentiment analysis.

4.5.3 Future ideas of creating 1Ds using deep learning

43

1D CNNs have become an increasingly popular tool for analyzing time series data, and the future ideas of creating 1Ds using deep learning are numerous and exciting. As deep learning techniques continue to advance, we can expect to see even more applications for 1D CNNs in a variety of fields. One potential application of 1D CNNs is in the field of climate science. Climate data is inherently time-dependent, and 1D CNNs can be used to analyze patterns in temperature, precipitation, and other environmental variables over time. This could help to identify trends in climate change, as well as to make more accurate predictions about future climate patterns. Another potential application is in the field of finance. By analyzing time series data from financial markets, 1D CNNs can be used to detect patterns and make predictions about future market behavior. This could have applications in areas such as stock market forecasting, risk management, and algorithmic trading. In addition, 1D CNNs may have applications in the field of energy management. By analyzing energy consumption data over time, 1D CNNs can help to identify patterns and make predictions about future energy usage. This could have applications in areas such as energy efficiency, demand response, and renewable energy integration. Another potential application is in the field of robotics. By analyzing sensor data in real time, 1D CNNs can help robots to navigate and interact with their environment. This could have applications in areas such as manufacturing, logistics, and healthcare, where robots are increasingly being used to perform complex tasks. Finally, 1D CNNs may also have applications in the field of social media analysis. By analyzing time series data from social media platforms, 1D CNNs can be used to detect trends and make predictions about user behavior. This could have applications in areas such as marketing, customer engagement, and political analysis. Overall, the future ideas of creating 1Ds using deep learning are diverse and exciting. As deep learning techniques continue to advance, we can expect to see even more applications for 1D CNNs in a variety of fields. From climate

science to finance, energy management to robotics, and social media analysis to healthcare, 1D CNNs have the potential to transform the way we analyze and understand time series data. By leveraging the power of deep learning, we can unlock new insights and make more accurate predictions about the world around us.

CHAPTER 5

CONCLUSION

In conclusion, the development of a smart intrusion detection system for cyber attacks is a crucial step towards enhancing the security of digital systems. The increased sophistication and frequency of cyber attacks make it imperative to adopt more advanced approaches to detect and prevent intrusions. The proposed system utilizes machine learning techniques and big data analytics to enhance the accuracy of detecting attacks and minimize the rate of false positives. The system provides real-time monitoring and alerts, which ensures that attacks are detected promptly, and remedial measures are taken.

The use of machine learning models such as decision trees, neural networks, and support vector machines have demonstrated promising results in the detection of cyber attacks. These models can learn and adapt to new patterns of attacks, which makes the system more robust and effective. The system can also detect new and emerging threats, which may not be detected by traditional signature-based detection systems.

The integration of big data analytics with the intrusion detection system is also an essential feature. The system can process large volumes of data and extract valuable insights that can be used to improve the accuracy of the detection system. Big data analytics can also be used to detect anomalies in the network, which may be an indication of a potential attack.

The use of smart devices such as smartphones and wearables in the detection system is also an innovative approach. The devices can act as sensors that collect data about the user's behavior and communication patterns. The data collected from these devices can be analyzed to identify anomalies that may be an indication of a cyber attack. This approach has the potential to improve the accuracy of the detection system and provide more personalized security.

The deployment of the proposed system in various industries such as finance, healthcare, and e-commerce can provide significant benefits. The system can help organizations detect and prevent cyber attacks, which can result in data breaches,

45

financial losses, and reputational damage. The system can also enhance compliance with regulatory requirements such as HIPAA, PCI DSS, and GDPR. The proposed system is not without limitations. The system's accuracy may be affected by false positives and false negatives, which may result in unnecessary alerts or missed attacks. The system also requires significant computational resources to process and analyze large volumes of data, which may result in increased costs. In conclusion, the development of a smart intrusion detection system for cyber attacks is a necessary step towards enhancing the security of digital systems. The system utilizes machine learning techniques and big data analytics to provide real-time monitoring and alerts. The system can detect new and emerging threats and provide more personalized security. The deployment of the system in various industries can provide significant benefits, but the system's limitations must also be considered. Overall, the proposed system has the potential to improve the security posture of organizations and reduce the impact of cyber attacks.

REFERENCES

- [1]Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, Fernando C. Population based training of neural networks. arXiv preprint arXiv:1711.09846. 2017 Nov 27.
- [2] Salloum SA, Alshurideh M, Elnagar A, Shaalan K. Machine learning and deep learning techniques for cybersecurity: a review. InProceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020) 2020 (pp. 50-57). Springer International Publishing.
- [3] Lu KD, Zeng GQ, Luo X, Weng J, Luo W, Wu Y. Evolutionary deep belief network for cyber-attack detection in industrial automation and control system. IEEE Transactions on Industrial Informatics. 2021 Jan 21;17(11):7618-27.
- [4] Zhang C, Lim P, Qin AK, Tan KC. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE transactions on neural networks and learning systems. 2016 Jul 11;28(10):2306-18.
- [5]Gumaei A, Hassan MM, Huda S, Hassan MR, Camacho D, Del Ser J, Fortino G. A robust cyberattack detection approach using optimal features of SCADA power systems in smart grids. Applied Soft Computing. 2020 Nov 1;96:106658.
- [6]Yan X, Xu Y, Xing X, Cui B, Guo Z, Guo T. Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT. IEEE Transactions on Industrial Informatics. 2020 Feb 20;16(9):6182-92.
- [7]Tahsien SM, Karimipour H, Spachos P. Machine learning based solutions for security of Internet of Things (IoT): A survey. Journal of Network and Computer Applications. 2020 Jul 1;161:102630.
- [8] Mitchell R, Chen R. Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. IEEE Transactions on Dependable and Secure Computing. 2014 Mar 18;12(1):16-30.
- [9] Li A, Spyra O, Perel S, Dalibard V, Jaderberg M, Gu C, Budden D, Harley T, Gupta P. A generalized framework for population based training. InProceedings of the 25th ACM

(pp. 1791-1799).

- [10] Zheng D, Hong Z, Wang N, Chen P. An improved LDA-based ELM classification for intrusion detection algorithm in IoT application. *Sensors*. 2020 Mar 19;20(6):1706.
- [11] Esmalifalak M, Liu L, Nguyen N, Zheng R, Han Z. Detecting stealthy false data injection using machine learning in smart grid. *IEEE Systems Journal*. 2014 Aug 20;11(3):1644-52.
- [12] Xue D, Jing X, Liu H. Detection of false data injection attacks in smart grid utilizing ELM-based OCON framework. *IEEE Access*. 2019 Mar 4;7:31762-73.
- [13] Ho D, Liang E, Chen X, Stoica I, Abbeel P. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning* 2019 May 24 (pp. 2731-2741). PMLR.
- [14] Manimurugan S, Al-Mutairi S, Aborokbah MM, Chilamkurti N, Ganesan S, Patan R. Effective attack detection in internet of medical things smart environment using a deep belief neural network. *IEEE Access*. 2020 Apr 6;8:77396-404.
- [15] Xin Y, Kong L, Liu Z, Chen Y, Li Y, Zhu H, Gao M, Hou H, Wang C. Machine learning and deep learning methods for cybersecurity. *IEEE Access*. 2018 May 15;6:35365-81.

Appendix

A.SOURCE CODE

```

function CybAttackDetect()
stamp1=0;
stamp2=0;
stamp3=0;
stamp4=0;
stamp5=0;
stamp6=0;
stamp7=0;

stamp1a=0;
stamp2a=0;
stamp3a=0;
stamp4a=0;
stamp5a=0;
stamp6a=0;
stamp7a=0;

tic;
tstart = tic;
num=randi(5);
nums=strcat(num2str(num),'.xlsx');
filnam=strcat('C:\Users\ashiq\OneDrive\Desktop\Evolutionary_deep\CODE\',nums);
[ndata, text, alldata] = xlsread(filnam);

d1=ndata(:,5);
d2=ndata(:,6);

d3=ndata(:,7);
d4=ndata(:,8);
d5=ndata(:,9);
d6=ndata(:,10);
d7=ndata(:,8);
d8=ndata(:,12);
d9=ndata(:,13);

```

```

d10=ndata(:,14);
d11=ndata(:,15);
d12=ndata(:,32);
d13=ndata(:,17);
test_data_in=[d7,d8,d9,d10,d11,d12];
test_data_in_fl=d7(:);
fld_test=text(:,3);
% *****
% Feature Extract
% *****
% [Newdata,PCASpace,EigValues]=PCASVD(d1(1:10));
for new_loop=1:10000
espace_data_viz(new_loop)=test_data_in(randi(numel(test_data_in)));
end
nsamp_test=1000;

[Newdata,PCASpace,EigValues]=PCASVD(espace_data_viz(1:nsamp_test))
; figure,

subplot(1,2,1)
stem(EigValues(1:nsamp_test))
title('PCA Space Values');

subplot(1,2,2)

50

stem(Newdata(1:nsamp_test))
title('PCA Space Values');

ps=sum(sum(PCASpace(:)))/1000;
ev=sum(sum(EigValues(:)))/1000;
nd=sum(sum(Newdata(:)))/1000;

% *****
% DECISION MAKING MODEL

```

```

% *****
% Deep dense CNN model called here
% *****

[cv1,cv2,stamp_noa,targetDD]=RCNN_TrTs_2(test_data_in,test_data_in_fl)
; [c,cm,ind,per] = confusion(double(targetDD),double(cv2(:)));
[rr cc]=size(per);
TN=0;TP=0;FN=0;FP=0;
for i=1:rr
    FN=FN+per(i,1);% false negative rate = (false negatives)/(all output negatives)
    FP=FP+per(i,2);% false positive rate = (false positives)/(all output positives)
    TP=TP+per(i,3);% true positive rate = (true positives)/(all output positives)
    TN=TN+per(i,4);% true negative rate = (true negatives)/(all output negatives)
end
[ACC,PREC,REC,F1SCO,SPEC]=performance_measure_routine(TP,TN,FP,FN)
; pause(2)
stamp_noa
% plotconfusion(double(targetDD),double(cv2(:)))
% % %
for kki=1:7
    if uint8(cv1(kki))==1
        stamp1=stamp1+1;
    end

    if uint8(cv1(kki))==2
        stamp2=stamp2+1;
    end
    pause(1);
    if uint8(cv1(kki))==3
        stamp3=stamp3+1;
    end

    if uint8(cv1(kki))==4
        stamp4=stamp4+1;
    end
end

```

```
if uint8(cv1(kki))==5
    stamp5=stamp5+1;
```

```
end
```

```
if uint8(cv1(kki))==6
    stamp6=stamp6+1;
```

```
end
```

```
if uint8(cv1(kki))==7
    stamp7=stamp7+1;
```

```
end
```

```
end
```

```
for kki=1:5
```

```
if (uint8(cv2(kki))==1)
    stamp1a=stamp1a+1;
```

```
end
```

```
if (uint8(cv2(kki))==2)
    stamp2a=stamp2a+1;
```

52

```
end
```

```
if (uint8(cv2(kki))==3)
    stamp3a=stamp3a+1;
```

```
end
```

```
if (uint8(cv2(kki))==4)
    stamp4a=stamp4a+1;
```

```
end
```

```
if (uint8(cv2(kki))==5)
    stamp5a=stamp5a+1;
```

```
end
```

```
if (uint8(cv2(kki))==6)
    stamp6a=stamp6a+1;
```

```

end
if (uint8(cv2(kki))==7)
    stamp7a=stamp7a+1;
end
end

test_pattern=[stamp1a,stamp2a,stamp3a,stamp4a,stamp5a,stamp6a,stamp7a];
hold_indexx=77;

for kjj=1:7
    if (max(test_pattern)==test_pattern(kjj))
        hold_indexx=kjj;
    end
end

for i=1:600

    53
    check_data(i)=test_data_in(randi(numel(test_data_in)));
end
[ps ev nd]
gh=[ps ev nd];
Espa_value=sum(sum(gh))
if(Espa_value<10)
    sa(stamp_noa);
% fprintf('Attack Deteted=%s\n',sa);
end
toc;
telapsed = toc(tstart);

function CybAttackDetect()
stamp1=0;
stamp2=0;
stamp3=0;
stamp4=0;

```



```
stamp5=0;
```

```
stamp6=0;
```

```
stamp7=0;
```

```
stamp1a=0;
```

```
stamp2a=0;
```

```
stamp3a=0;
```

```
stamp4a=0;
```

```
stamp5a=0;
```

```
stamp6a=0;
```

```
stamp7a=0;
```

```
tic;
```

```
tstart = tic;
```

54

```
[f p]=uigetfile('*.xlsx');
```

```
num=randi(5);
```

```
nums=strcat(num2str(num),'.xlsx');
```

```
filnam=strcat('C:\Users\ashiq\OneDrive\Desktop\Evolutionary_deep\CODE\',nums);
```

```
[ndata, text, alldata] = xlsread([p f]);
```

```
d1=ndata(:,5);
```

```
d2=ndata(:,6);
```

```
d3=ndata(:,7);
```

```
d4=ndata(:,8);
```

```
d5=ndata(:,9);
```

```
d6=ndata(:,10);
```

```
d7=ndata(:,8);
```

```
d8=ndata(:,12);
```

```
d9=ndata(:,13);
```

```
d10=ndata(:,14);
```

```
d11=ndata(:,15);
```

```
d12=ndata(:,32);
```

```
d13=ndata(:,17);
```

```

test_data_in=[d7,d8,d9,d10,d11,d12];
test_data_in_fl=d7(:);
fld_test=text(:,3);
% *****
% Feature Extract
% *****
% [Newdata,PCASpace,EigValues]=PCASVD(d1(1:10));
for new_loop=1:10000
espace_data_viz(new_loop)=test_data_in(randi(numel(test_data_in)));
end
nsamp_test=1000;

```

55

```

[Newdata,PCASpace,EigValues]=PCASVD(espace_data_viz(1:nsamp_test))
; figure,

```

```

subplot(1,2,1)
stem(EigValues(1:nsamp_test))
title('PCA Space Values');

```

```

subplot(1,2,2)
stem(Newdata(1:nsamp_test))
title('PCA Space Values');

```

```

ps=sum(sum(PCASpace(:)))/1000;
ev=sum(sum(EigValues(:)))/1000;
nd=sum(sum(Newdata(:)))/1000;

```

```

% *****
% DECISION MAKING MODEL
% *****
% ADAPTIVE CNN model called here
% *****
[cv1,cv2,stamp_noa,targetDD]=adapCNN(test_data_in,test_data_in_fl);

```

```

[c,cm,ind,per] = confusion(double(targetDD),double(cv2(:)));
[rr cc]=size(per);
TN=0;TP=0;FN=0;FP=0;
for i=1:rr
    FN=FN+per(i,1);% false negative rate = (false negatives)/(all output negatives)
    FP=FP+per(i,2);% false positive rate = (false positives)/(all output positives)
    TP=TP+per(i,3);% true positive rate = (true positives)/(all output positives)
    TN=TN+per(i,4);% true negative rate = (true negatives)/(all output negatives)
end
[ACC,PREC,REC,F1SCO,SPEC]=performance_measure_routine(TP,TN,FP,FN)

```

; 56

```

pause(2)
stamp_noa
% plotconfusion(double(targetDD),double(cv2(:)))
% % %
for kki=1:7
    if uint8(cv1(kki))==1
        stamp1=stamp1+1;
    end

    if uint8(cv1(kki))==2
        stamp2=stamp2+1;
    end
    pause(1);
    if uint8(cv1(kki))==3
        stamp3=stamp3+1;
    end

    if uint8(cv1(kki))==4
        stamp4=stamp4+1;
    end

    if uint8(cv1(kki))==5

```

```

        stamp5=stamp5+1;
    end
    if uint8(cv1(kki))==6
        stamp6=stamp6+1;
    end
    if uint8(cv1(kki))==7
        stamp7=stamp7+1;
    end
end
end

```

57

```

for kki=1:5
    if (uint8(cv2(kki))==1)
        stamp1a=stamp1a+1;
    end

    if (uint8(cv2(kki))==2)
        stamp2a=stamp2a+1;
    end

    if (uint8(cv2(kki))==3)
        stamp3a=stamp3a+1;
    end

    if (uint8(cv2(kki))==4)
        stamp4a=stamp4a+1;
    end

    if (uint8(cv2(kki))==5)
        stamp5a=stamp5a+1;
    end
    if (uint8(cv2(kki))==6)
        stamp6a=stamp6a+1;
    end
    if (uint8(cv2(kki))==7)

```

```

    stamp7a=stamp7a+1;
end
end

```

```

test_pattern=[stamp1a,stamp2a,stamp3a,stamp4a,stamp5a,stamp6a,stamp7a];
hold_indexx=77;

```

58

```

for kjj=1:7
    if (max(test_pattern)==test_pattern(kjj))
        hold_indexx=kjj;
    end
end

```

```

for i=1:600
    check_data(i)=test_data_in(randi(numel(test_data_in)));
end
[ps ev nd]
gh=[ps ev nd];
Espa_value=sum(sum(gh))
if(Espa_value<10)
    sa(stamp_noa);
end
toc;
telapsed = toc(tstart);

```

```

function
[cv1,cv2,stamp_noa,targetDD]=RCNN_TrTs_2(Res_2D,tefl)
warning off;
figure
ancnt5=0;ancnt4=0;ancnt3=0;ancnt2=0;ancnt1=0;ancnt6=0;ancnt7=0;ancnt=0;

[ndata1, text, alldata] =
xlsread('C:\Users\ashiq\OneDrive\Desktop\Evolutionary_deep\CODE\Train_data_new.xls');

```

```
%*****
```

```
size(ndata1)
```

```
% pause(5);
```

```
% % %
```

59

```
d1=ndata1(:,8);% Wrong Connectivity Flag
```

```
d2=ndata1(:,12);% Loggin Stamp
```

```
d3=ndata1(:,13);% Num of times Compromised
```

```
d4=ndata1(:,14);% Root shell
```

```
d5=ndata1(:,15);% Scheduling Units
```

```
d6=ndata1(:,32);% Destination Host
```

```
data2=[d1,d2,d3,d4,d5,d6];
```

```
c1=data2(2:500);
```

```
c2=data2(501:700);
```

```
c3=data2(701:900);
```

```
c4=data2(901:1000);
```

```
c5=data2(1001:7000);
```

```
c6=data2(7001:7100);
```

```
c7=data2(7100:10000);
```

```
% % % %*****
```

```
% % % % Training Session
```

```
% % % %*****
```

```
% % % for Recc_loop=1:1
```

```
for i=1:1000
```

```
    fea1(i)=c1(randi(numel(c1)));
```

```
    fea2(i)=c2(randi(numel(c2)));
```

```
    fea3(i)=c3(randi(numel(c3)));
```

```
    fea4(i)=c4(randi(numel(c4)));
```

```
    fea5(i)=c5(randi(numel(c5)));
```

```
    fea6(i)=c6(randi(numel(c6)));
```

```
    fea7(i)=c7(randi(numel(c7)));
```

```
end
```

```
l1=fea1;  
l2=fea2;
```

60

```
l3=fea3;  
l4=fea4;  
l5=fea5;  
l6=fea6;  
l7=fea7;
```

```
trainD(:,:,1)=l1;  
trainD(:,:,2)=l2;  
trainD(:,:,3)=l3;  
trainD(:,:,4)=l4;  
trainD(:,:,5)=l5;  
trainD(:,:,6)=l6;  
trainD(:,:,7)=l7;
```

```
figure
```

```
targetD=categorical([1;2;3;4;5;6;7]);
```

```
% % % % % Define the RCNN - R-convolutional neural network architecture.
```

```
layers = [
```

```
    imageInputLayer([1000 1 1])
```

```
    convolution2dLayer(1,10,'Stride',4);
```

```
    reluLayer
```

```
    fullyConnectedLayer(384) % 384 refers to number of neurons in next FC hidden layer
```

```
    fullyConnectedLayer(384) % 384 refers to number of neurons in next FC hidden layer
```

```
    fullyConnectedLayer(7) % 6 refers to number of neurons in next output layer (number of  
output classes)
```

```
    softmaxLayer
```

```
    classificationLayer];
```

```
% options1 = trainingOptions('sgdm','Verbose',false, ...
```

```
% 'MaxEpochs',1000, ...
```

```
% 'InitialLearnRate',0.5, ...
```

```
% 'OutputFcn',@plotTrainingAccuracy)
```

```
options33 = trainingOptions('sgdm','MaxEpochs',85,...
```

```

'InitialLearnRate',0.05, ...
'OutputFcn',@plotTrainingAccuracy);

title('Deep Learning Training Accuracy');
legend('Training');
% savefig('PeaksFile.fig')

% options4 = trainingOptions('sgdm','MaxEpochs',1000,...
% 'InitialLearnRate',0.0001);

net = trainNetwork(trainD,targetD',layers,options33);
predictedLabels_1 = classify(net,trainD)';
cv1=predictedLabels_1;

% % % % *****
% % % % Testing Phase
% % % % *****

figure
% for Recc_loop=1:1
    kk=Res_2D;
    for i=1:1000
        fea1(i)=kk(randi(numel(kk)));
        fea2(i)=kk(randi(numel(kk)));
        fea3(i)=kk(randi(numel(kk)));
        fea4(i)=kk(randi(numel(kk)));
        fea5(i)=kk(randi(numel(kk)));
    end
l1=fea1;
l2=fea2;
l3=fea3;

l4=fea4;

```



```

l5=fea5;
trainD(:,:,,1)=l1;
trainD(:,:,,2)=l2;
trainD(:,:,,3)=l3;
trainD(:,:,,4)=l4;
trainD(:,:,,5)=l5;
trainD(:,:,,6)=l6;
trainD(:,:,,7)=l7;

```

```

kcheck=tefl;
kk=kcheck(:);
for jk=1:numel(kk)
    if kk(jk)==3
        ancnt1=ancnt1+1;
    end
    if kk(jk)==4
        ancnt2=ancnt2+1;
    end
    if kk(jk)==5
        ancnt3=ancnt3+1;
    end
    if kk(jk)==6
        ancnt4=ancnt4+1;
    end
    if kk(jk)==7
        ancnt5=ancnt5+1;
    end
    if kk(jk)==0
        ancnt6=ancnt6+1;
    end

```

```

    if kk(jk)==9
        ancnt7=ancnt+1;
    end

```

```

end
stamp_noa=99;
noa=[ancnt1 ancnt2 ancnt3 ancnt4 ancnt5 ancnt6 ancnt7];

% Check Test Data
for jkk=1:numel(noa)
    if (max(noa))==noa(jkk)
        stamp_noa=jkk;
    end
end
% targetDD=categorical(1);
targetDD=categorical([1;2;3;4;5;6;7]);
layers = [
    imageInputLayer([1000 1 1])
    convolution2dLayer(1,10,'Stride',4);
    reluLayer
    fullyConnectedLayer(384) % 384 refers to number of neurons in next FC hidden layer
    fullyConnectedLayer(384) % 384 refers to number of neurons in next FC hidden layer
    fullyConnectedLayer(7) % 6 refers to number of neurons in next output layer (number of
    output classes)
    softmaxLayer
    classificationLayer];
options3 = trainingOptions('sgdm','MaxEpochs',250,...
    'InitialLearnRate',0.0001, ...
    'OutputFcn',@plotTrainingAccuracy);

title('Deep Learning Testing Accuracy');
legend('Prediction');

```

64

```

% net = trainNetwork(trainD,targetDD',layers,options2); net2 =
trainNetwork(trainD,targetDD',layers,options3); % predictedLabels_2 =
classify(net,trainD)'; predictedLabels_3 = classify(net2,trainD)'; cv2=predictedLabels_3;

% Normalized_Accuracy

```

```
accuracy = (100-sum(targetDD == cv2)/numel(cv2)) 65
```

B.SCREENSHOT

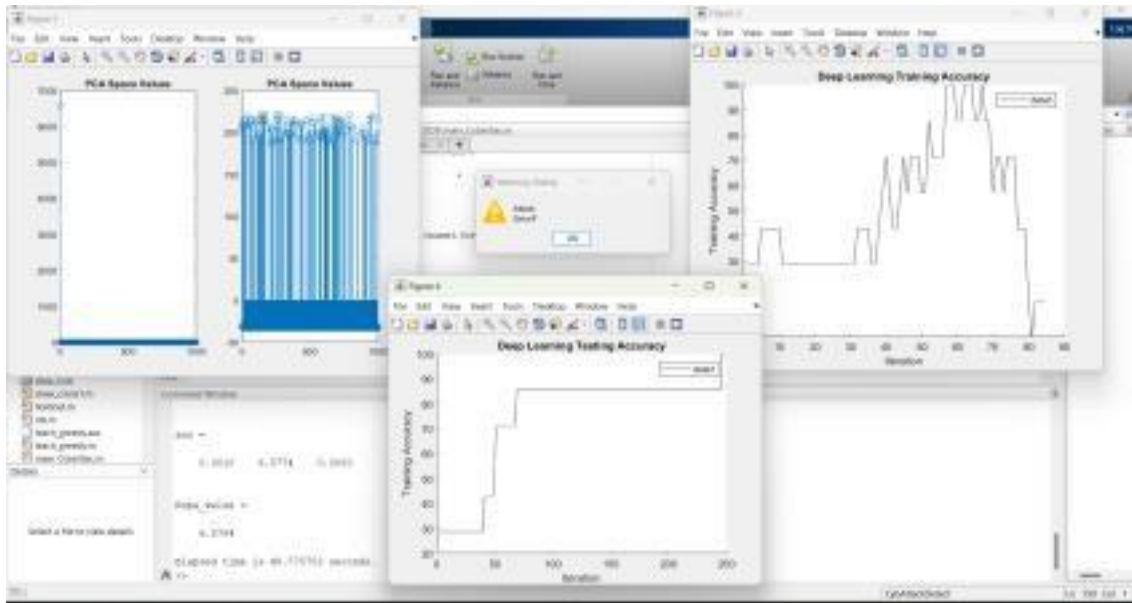
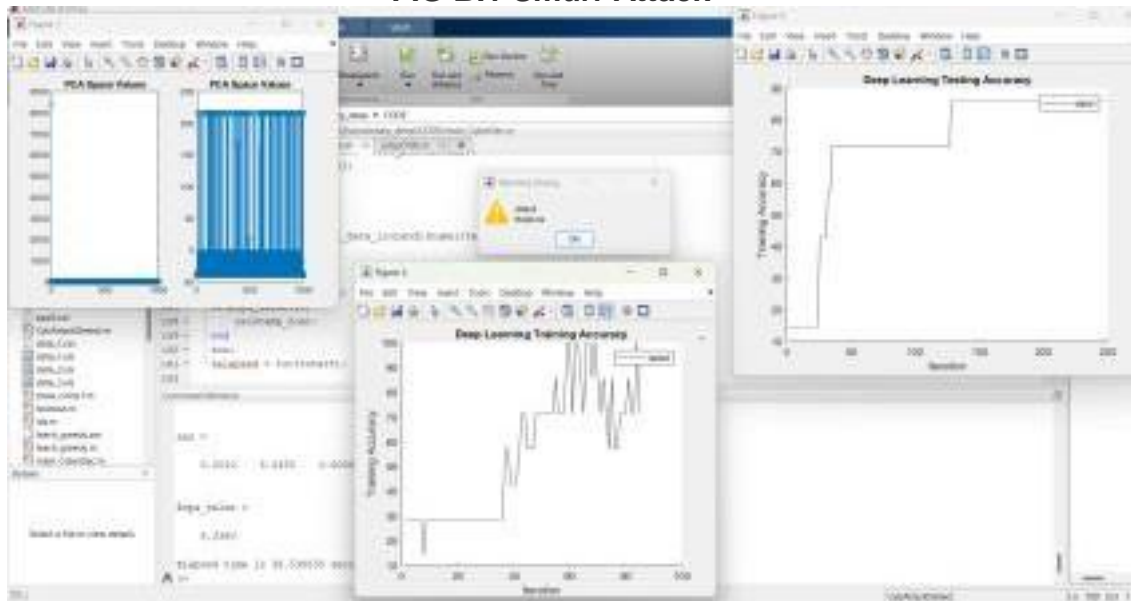


FIG B.1 Smurf Attack



FIGB.2NEPTUNEATTACK

FIG B.3 Guess password attack

FIG B.4 Buffer Overflow attack

RESILIENT DETECTION OF CYBER ATTACKS IN INDUSTRIAL DEVICES

Y. Ashiq Meeran

Sathyabama institute of science and technology

Chennai, India

ashiqmeeran25ashiqmeeran3@gmail.com

S. Prayla shyry,

Sathyabama institute of science and technology

Chennai, India,

praylashyry.cse@sathyabama.ac.in

Abstract– With the advent of smartphones, laptops, and home computers, smart systems are becoming more and more flexible. As the use of the internet increases, there will be more cyber threats occurring on most third-party connectivity websites. The powerful technique used to detect the threats present in the IoT applications are discussed in the proposed system. Based on the KAGGLE NIDS(Network Intrusion Detection System)(Intrusion Detection System) dataset, the number of possible attacks is calculated in the proposed architecture. A similar occurrence of intrusion creating a task is detected by the system, triggering the model to prevent the intrusion by notifying the user immediately. The existing attack detection systems have a number of limitations which includes the need of human intervention to detect the attacks encountered, slower detection rate and inaccuracy in detection. An advanced deep learning algorithm is proposed for detecting possible intrusions to overcome these limitations. The proposed design focuses on creating a Novel architecture using Adaptive convolutional neural network for improving the accuracy and significantly raising the detection rate above that of the current approaches there by aiding in the immediate detection of intrusions.

Keywords– deep learning, 1-dimensional Convolutional Neural Network, Adaptive algorithm, Network Intrusion detection system, Optimization.

I. INTRODUCTION

Generally in ancient times the Intrusion Detection Systems were detected by the administrator who used to sit in front of a console and watch the user activities. For example, if a user is in a vacationing period and his account is logged in locally or Seldom printer is Active. And this was effective enough to detect the Intrusion attack that period, the early form IDS(Intrusion Detection System)(Intrusion was an ad hoc and not scalable. Later it was updated, they introduced audit logs which were reviewed by the administrator for the evidence of an unusual activity in their network. In the 1970s and early 80's administrators

only used printed audits and logs. They obviously get stacked upto 4-5 feet in a weekend, it was really a time consuming

process at that time period. With the overabundance of information to go through manual search, administrators used the audit log as a forensic tool to go through after the security incident happened. In the early 90's, researchers had

developed a real time IDS(Intrusion Detection System) which reviews the audit data as that was produced. Due to the ability to detect assaults and attempted attacks as they happened, real time response and, in certain situations, attack pre-emption, were made possible. Nowadays intrusion

detection efforts have been made easy with the help of developing products which in turn allowed users to deploy in a network. First of all let us see what is meant by Intrusions

Detection system, classifications of Intrusion detections system and how it works. IDS(Intrusion Detection System) is an attack detection system. It is used to detect the vulnerabilities, to protect the system from the attackers who try to invade into the network line. When the known

vulnerabilities are detected, it will raise alarm to the concerned personnel to take necessary action. It continuously monitors inbound and outbound activities of the system and it monitors the behaviour of the system and ensures proper working of the system. Basically, IDS(Intrusion Detection System) are two types Active and Passive. Passives are used to detect the instructions, but Active IPS not only detects intrusion in the network but also prevents them. In essence, the IDS(Intrusion Detection System) are positioned close to the firewall. The IDS(Intrusion Detection System) can be deployed outside or within the firewall, depending on the traffic to be watched, to keep an eye on suspicious traffic coming from either location. The IDS(Intrusion Detection System) will function best if it is positioned inside close to a DMZ, but the recommended practice is to use a layered defence by placing one IDS(Intrusion Detection System) in front of the firewall and another one in the network behind

68

the firewall. Prior to deploying the IDS(Intrusion Detection System), it is crucial to perform a network topology analysis, comprehend how traffic moves to and from resources that an attacker could use to enter the network, and identify the crucial components that could become the targets of different network attacks.

A. Network based Intrusion Detection System

Every packet that enters the network is examined by network-based intrusion detection systems (NIDS(Network Intrusion Detection System)(Intrusion Detection System)) for irregularities and erroneous data. The firewall's ability to drop a lot of data packets forces the NIDS(Network Intrusion

Detection System)(Intrusion Detection System) to extensively inspect each packet. All traffic is recorded and

inspected by an NIDS(Network Intrusion Detection System)(Intrusion Detection System). Based on the content, it creates notifications at the IP or application level. Compared to host-based IDS(Intrusion Detection System),

NIDS(Network Intrusion Detection System)(Intrusion Detection System) are more dispersed. The router and host tiers of the NIDS(Network Intrusion Detection System)(Intrusion Detection System) detect anomalies. After

receiving the data packets, it assigns a threat level to each risk and audits the information in the data packets and logs the information of malicious packets. The security staff can maintain a state of alertness due to the threat level. These defences typically consist of a black box deployed in promiscuous mode on the network, listening for patterns suggestive of an incursion. By observing network traffic, it finds harmful behavior such as DoS attacks, port scans, or even attempts to access machines.

B. Deep learning model to detect cyber attacks

A deep learning model can quickly analyse large amounts of data and detect patterns in malicious behaviour. This allows it to take proactive action before a cyber-attack takes place, saving organizations from costly downtime and financial losses associated with malicious attacks. Additionally, deep learning models provide advantages in accuracy and precision over other methods available for cyberattack detection such as intrusion detection system (IDS(Intrusion Detection System)).

An IDS(Intrusion Detection System) system uses deep neural networks to detect anomalies in network traffic or systems that can be indicators of malicious activities on a network or system. IDS(Intrusion Detection System) systems are important security tools in protecting a network from cyber-attacks and malicious activities. They use deep learning algorithms to detect unusual patterns of traffic or system activities that can signal a security breach without relying on manual intervention.

II. LITERATURE SURVEY

The paper's [1],[9],[13] have used the basic idea of Population Based Training which uses an asynchronous very few and are also time consuming. In the present work optimization algorithm for reliable model training. The paper we have mainly focused on reducing the time taken by using [3] uses a deep belief network detection method based on the adaptive algorithm which makes the one dimensional population extremal optimization (PEO) to find SCADA-CNN much faster than the standard CNN based IACS cyberattacks, although the PEO procedure takes a long time to evaluate fitness. The paper[4] employs the multiobjective deep belief networks ensemble (MODBNE) technique, which produces accuracy and diversity but requires more time due to the increased number of hidden layers. The paper [5] uses a correlation-based feature selection(CFS) method to remove irrelevant features but the detection or classification methodology has not been improved. In the paper [6], scale and speed comparisons between a new hinge classification method (HCA-MBGDALRM) and conventional neural networks, decision trees, and logistic regression are made. The paper [7] covers numerous ML-based intrusion detection methods and contrasts the accuracy of various algorithms for various attack types, in which the accuracy of detecting network-based intrusion detection is not accomplished. In paper [8] designs an intrusion detection technique for safety critical medical cyber physical systems based on the behavior rule-specification. This is a classical intrusion detection technique that works with sensors and actuators. It detects changes in behaviour through sensors and warns the system about the deviation in behaviour. This technique detects only the occurrence of attacks but not the type of attacks. In Paper [10] an elm classification algorithm that uses Moore Penrose fixed the weights by generalizing the inverse instead of using gradient-based backpropagation. In. This algorithm works only in a single hidden layer feed forward neural network thus it cannot be used to train a complex dataset. This algorithm is not very efficient as it generates only an accuracy of 92.35%. In printed edition [11], detection of stealthy false data injection attacks using machine learning algorithms. This model is designed to detect a single type of attack, and is not effective in the case of multiple attacks. This uses a machine learning algorithm that trains on lesser data and gives lesser accuracy compared to the deep learning techniques. In printed edition [12], The framework utilized for detecting false data injection attacks is ELM-based OCON (One Class

One Network). The main goal of one-class classification is that, it tries to detect the objects of a specific Class within a group of objects. By mainly acquiring knowledge from a training dataset that exclusively comprises instances of that particular class. Thus this model cannot be used to detect multiple classes of attacks encountered by the network. The paper [13] has introduced deep belief neural networks for attack detection in the internet of medical things. The deep belief neural network used in this model is similar to the

69

convolutional neural network in the proposed system, but the proposed system stands out in efficiency because of the added layers: convolution layer and pooling layer. In paper [15] and [2], the machine learning and deep learning techniques and methods for cybersecurity, but these techniques are not applied and no model that detects cyber-attacks has been designed. Upon conducting a comprehensive literature survey, several key insights have been gleaned. Research involving CNN for NIDs is considerably less. There has been numerous studies which include deep learning, but studies that explore the usage of

III. PROPOSED SYSTEM

The proposed research is based on an advanced intrusion detection model. The robust architecture detects and provides the collection of a number of possible attacks in the massive internet of things network. The collection of intrusion models we call as bags of attacks. The proposed machine learning algorithm creates a robust prediction system for detection of feasible intrusions in the IoT network; the vulnerability of the IoT attacks act as a key for detecting the intrusion present in the network. The proposed design focuses on creating a Novel architecture through the Deep convolutional neural network for improving the accuracy and increased security. The creation of this intrusion detection model involves training, and testing of the neural networks using deep learning techniques, along with some pre-processing performed on the data set.

A. Pre-processing data

Pre-processing is the first step in deep learning in which we prepare raw data in a format that the network can acquire. The data set may contain some missing values that must be filled by performing some calculations. The missing values are obtained by calculating the mean of the other values in the specific row or column. The 'get_dummies' function is used to assign numeric values to the character and string values. This procedure allows for improved accuracy and faster training times due to the nature of neural networks being able to better process numerical values over nominal values. When get_dummies are introduced into the data set, it takes a list or a data frame and converts the unique elements which are present in the object to a column. The method checks if the element at a specific index matches the column header as

it iterates over the object that is supplied. If so, it encodes it as a 1 else it gives it a 0. The next step in pre-processing is Standard scaling or uniform

scaling. When the scale differs from feature to feature, the weights in neural networks do not converge easily and local minimum is not obtained. Therefore, we change it to a uniform scale where the mean value is 0 and standard deviation is 1. Gaussian distribution is used for standardization of data. Standard scaling works by first calculating the mean value of each feature in the dataset and then subtracting this value from every sample in order to provide a mean value of 0. The standard deviation is then calculated and applied as a scaling factor (positive one). Standard scaling can help improve the performance of deep learning algorithms by more accurately representing the volumes of training data. By repeatedly adjusting weights features within the dataset. Pre-processing includes a feature selection process. This process helps in reducing the size of the feature sets thereby preserving the information and refine their calculations over time. In order to define these maintaining the efficiency. The feature selection process helps reduce the problem dimensionality by removing redundant or irrelevant features while preserving those that are important to make better predictions. These techniques help facilitate better results for the model.

B. Training

Deep learning requires extensive training data in order to power deep learning models, resulting in faster and more accurate results. As a result, it is important to have accurate predictions. Training data is used to teach the the right data sets available in order to ensure high quality networks what is expected from it – this makes them adaptive outcomes and accuracy. Loading pre-processed data is very and the desired accuracy can be achieved through training important in this process and can have a direct impact on the deep neural networks. An one-dimensional Convolutional results of deep learning applications. Pre-processing this data Neural Network (1-D CNN), consists of an input layer, can be a daunting task, as it may require cleaning, preparing several hidden neurons with multiple layers, and an output and normalizing it so that it can be properly used in training layer. The nodes of each layer are connected in a way that proceedings. Loading this pre-processed data into deep helps generate accurate results by learning through repetitive learning models is a complex process since numerous factors decision making and taking into account pre-defined have to be taken into consideration, including the available parameters such as accuracy, precision, recall etc. memory capacity, network resources or deployment time requirements. Therefore, it is crucial to optimize this process while ensuring that all targets are met so that the model can properly intake and benefit from this data. Deep learning

Validation data set helps to ensure that the model being used helps us to learn features from data on its own. It does this by is accurate and effective in its predictions. Before testing the using different algorithms to extract features from the data final model, it is important to have a dataset to validate it with and construct a model for classification. Training data is an and set up all necessary parameters such as weights and important part of deep learning, as it helps in developing an hyperparameters. Validation helps in evaluating its accurate model that can classify data according to its type. performance and accuracy. The model is tested for accuracy Feature extraction is the process of extracting relevant using the hold out training dataset. This Testing is done on a information from the training data while classification sample of data, usually separate from the original training set, involves classifying each sample into distinct categories. and then compared with known outcomes or labels. Using the Feature extraction and classification both play a major role in datasets, measure the weights, parameters, and results the development of deep learning models. Deep learning uses obtained from the training process. Validation data sets help to identify any neural network algorithms to recognize patterns in big data discrepancies between algorithms, thereby providing an sets. The algorithm used here is Adaptive 1-Dimensional opportunity to make changes before deploying them in Convolutional Neural Networks (CNN). CNNs are a type of production. The validation process is essential for tuning deep learning network used to train machines to recognize hyperparameters and solving optimization problems. It also patterns in inputting data such as text. They work by sliding helps to determine if modifications are necessary or changes through the dataset columns and using the output layer's are possible to ensure that the deep learning model performs connections (known as weights) to find the most suitable as expected.

values, we can ensure proper training of a deep learning network and make sure that it obtains accurate results when deployed in real-world tasks. The task of setting the number of layers in a 1-dimensional Convolutional Neural Network is done in this process. Depending on the type and size of the dataset, determining the optimal layer count may require several attempts with different configurations.

The selection and configuration of layers should be determined based on factors such as accuracy, computational complexity, data type, model complexity, etc. Utilizing proper techniques during training can help minimize common pitfalls such as overfitting or noise mis-interpretation. It relies heavily on the availability of large learning algorithms by more accurately representing the volumes of training data. By repeatedly adjusting weights features within the dataset. Pre-processing includes a feature selection process. This process helps in reducing the size of the feature sets thereby preserving the information and refine their calculations over time. In order to define these maintaining the efficiency. The feature selection process helps reduce the problem dimensionality by removing redundant or irrelevant features while preserving those that are important to make better predictions. These techniques help facilitate better results for the model.

C. Validation

Validation data set helps to ensure that the model being used helps us to learn features from data on its own. It does this by is accurate and effective in its predictions. Before testing the using different algorithms to extract features from the data final model, it is important to have a dataset to validate it with and construct a model for classification. Training data is an and set up all necessary parameters such as weights and important part of deep learning, as it helps in developing an hyperparameters. Validation helps in evaluating its accurate model that can classify data according to its type. performance and accuracy. The model is tested for accuracy Feature extraction is the process of extracting relevant using the hold out training dataset. This Testing is done on a information from the training data while classification sample of data, usually separate from the original training set, involves classifying each sample into distinct categories. and then compared with known outcomes or labels. Using the Feature extraction and classification both play a major role in datasets, measure the weights, parameters, and results the development of deep learning models. Deep learning uses obtained from the training process. Validation data sets help to identify any neural network algorithms to recognize patterns in big data discrepancies between algorithms, thereby providing an sets. The algorithm used here is Adaptive 1-Dimensional opportunity to make changes before deploying them in Convolutional Neural Networks (CNN). CNNs are a type of production. The validation process is essential for tuning deep learning network used to train machines to recognize hyperparameters and solving optimization problems. It also patterns in inputting data such as text. They work by sliding helps to determine if modifications are necessary or changes through the dataset columns and using the output layer's are possible to ensure that the deep learning model performs connections (known as weights) to find the most suitable as expected.

D. Testing

Once the validation is done, the last step in the development process for the deep learning model is testing. A test dataset, also known as a holdout dataset, is a small subset of data that

is held out from the training phase and used towards the end of the model development process for testing accuracy. Testing helps to verify the accuracy, performance and stability of such models before deployment in any production environment. It is needed to assess the quality of the model's outputs on unseen data and make sure the system will work properly in real-world situations.

IV. RESULT AND DISCUSSION

A. (PCA)Principal Component Analysis

Fig 1 : Principal Component Analysis

This graph denotes the difference in space values between the dataset before and after applying Principal Component Analysis technique. To enhance data interpretability while retaining the maximum amount of information, principal component analysis (PCA) is utilized as a common method for analyzing large datasets that contain numerous dimensions or features per observation. PCA can also facilitate the visualization of data that exist in multiple dimensions.

B Performance Learning Curves: